NUMERICAL METHODS FOR COMPUTING TURBULENT FLOWS

Milovan Perić
Computational Dynamics Ltd
Dürrenhofstrasse 4, 90402 Nürnberg, Germany

1 INTRODUCTION

Numerical methods for computing turbulent flows are derived from methods developed for laminar, viscous flows. Although most of the methods developed for computing inviscid flows are of the same type and can be extended to turbulent flows, there are some special methods which can not be extended in a straightforward manner; the viscous part of the Navier-Stokes equations is essential in turbulent flows and any method to be used for computing such flows must deal efficiently with the elliptic equations.

There are several features by which the numerical methods for computing turbulent flows can be classified. The approach used here is not the only possibility, but it is appropriate enough; the classification is according to:

- the discretization method used to approximate the differential or integral conservation equations by a system of algebraic equations which can be solved on a computer;
- the time-integration method;
- the method of pressure-velocity-density coupling.

The most often used discretization methods are finite-difference (FD), spectral (S), finite-element (FE), and finite-volume (FM) methods. There are of course also methods of mixed type, like the spectral-element methods (see e.g. Henderson and Karniadakis, 1995) or control-volume finite-element methods (see e.g. Baliga, 1997). Not all of these methods will be covered here; FD and FV methods, being the most widely spread, will be described in more detail, while only the basic information on the others will be provided.

The time-integration methods are either explicit or implicit; there are many variants of each group and again only some of them will be described in detail. The choice of the time-integration method is essential for both the accuracy and computational efficiency reasons. Special attention will be given to the computation of steady flows, which are representative of engineering applications in which only the mean features of the flow are of interest.

The method of pressure-velocity-density coupling is important in both incompressible and compressible flows, and especially if both flow types are to be covered. Again, many approaches are possible but only the most important ones will be described.

Special attention will be given to error estimation and computational efficiency, measured by the effort required to achieve a solution of given accuracy. Some of the methods – like the

use of local grid refinement – are applicable only to some types of discretization methods, while others – like multigrid acceleration and parallel computing – can always be used.

The methods used to solve the Reynolds-averaged Navier-Stokes equations also differ from those used to perform direct numerical simulations (DNS) and large-eddy simulations (LES) of turbulent flows, although the gap appears to be narrowing lately. In the following sections the emphasis will be given to second-order methods which are widely used to solve RANS-equations, both in commercial codes and in academic research. These kinds of methods are nowadays also being used in LES and DNS, especially those which offer local grid refinement techniques.

Section 2 describes the discretization methods. It is followed by a section on time integration techniques. Section 4 deals with the methods for solving linear equation systems. The methods of computing incompressible flows are covered in the following section. Section 6 is devoted to flows in complex geometries and the aspects of grid adaptation. In section 7, the methods of computing compressible flows are discussed. In section 8, an approach to moving grids is described. Finally, some concluding remarks are given.

The following descriptions of the various methods are kept concise; for a more detailed analysis see books by Anderson et al. (1984), Hirsch (1992), Fletcher (1991), and Ferziger and Perić (2002), among others.

2 DISCRETIZATION METHODS

In order to describe some of the discretization methods, a generic conservation equation for quantity ϕ will be considered. It will be shown later that all conservation equations have the same form as this generic equation; actually, by proper substitutions for ϕ, its diffusivity Γ, and the generic source term q_ϕ, all equations can be cast in this form:

$$\frac{\partial(\rho\phi)}{\partial t} + \boldsymbol{\nabla}\cdot(\rho\phi\mathbf{v}) = \boldsymbol{\nabla}\cdot(\Gamma\boldsymbol{\nabla}\phi) + q_\phi \ . \tag{1}$$

Depending on the coordinate system used, one has to provide the appropriate form of the divergence and gradient operator.

Some methods use the integral form of the conservation equation as the starting point; it reads:

$$\frac{\partial}{\partial t}\int_V \rho\phi \,\mathrm{d}V + \int_S \rho\phi\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = \int_S \Gamma\boldsymbol{\nabla}\phi\cdot\mathbf{n}\,\mathrm{d}S + \int_V q_\phi\,\mathrm{d}V \ , \tag{2}$$

where V is the volume of a control volume bounded by a closed surface S, $\mathbf{v}$ is the fluid velocity vector, and $\mathbf{n}$ is the outward-pointing unit vector normal to the surface S.

In the descriptions which follow it will be assumed that the coordinate system is arbitrary non-orthogonal, and that the control volume can have any polyhedral shape. The special forms of the expressions that result when Cartesian or curvilinear orthogonal coordinates are used can be obtained by appropriate simplifications of the equations. Also, vectors and tensors are expressed through their Cartesian components. This ensures that the strong conservation form

of the equations is preserved, since only the coordinates may need to be transformed, but the Cartesian base vectors are retained.

The discretization methods shall be described under the assumption that an implicit time-integration method is used, i.e. that an algebraic equation system needs to be solved. In the case of explicit methods, the same expressions remain valid but the fluxes and source terms can be directly computed using solution from previous time levels.

2.1 Finite-Difference Methods

The finite-difference (FD) methods are used in conjunction with structured grids, where each set of grid lines is considered to represent lines along which two of the three independent co-ordinates are constant (one of two in two-dimensional – 2D – problems). An example of a 2D, structured, non-orthogonal grid is shown in Fig. 1.

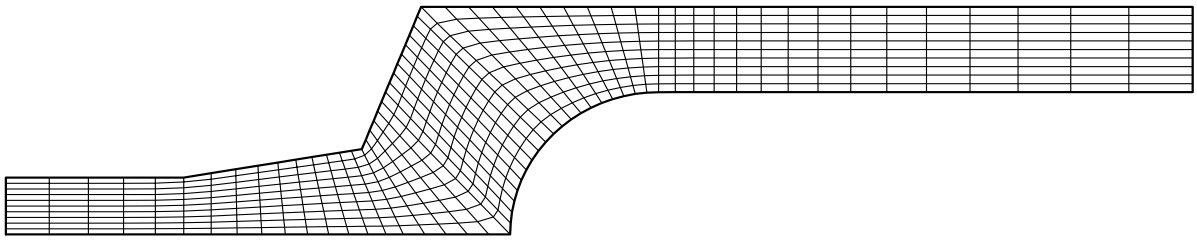

Fig. 1: An example of a 2D, structured, non-orthogonal grid

The generic conservation equation, which for Cartesian coordinates x_i and tensor notation reads:

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial(\rho u_j \phi)}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial \phi}{\partial x_j}\right) + q_\phi \ , \tag{3}$$

needs to be transformed when non-orthogonal coordinates ξ_j are used.:

$$J\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial(\rho U_j \phi)}{\partial \xi_j} = \frac{\partial}{\partial \xi_j}\left[\frac{\Gamma}{J}\left(\frac{\partial \phi}{\partial \xi_m} B^{mj}\right)\right] + J q_\phi \ , \tag{4}$$

where

$$U_j = u_k \beta^{kj} = u_1\beta^{1j} + u_2\beta^{2j} + u_3\beta^{3j} \tag{5}$$

is proportional to the velocity component normal to the coordinate surface ξ_j =const. The coefficients B^{mj} are defined as:

$$B^{mj} = \beta^{kj}\beta^{km} = \beta^{1j}\beta^{1m} + \beta^{2j}\beta^{2m} + \beta^{3j}\beta^{3m} \ . \tag{6}$$

where β^{ij} represents the co-factor of $\partial x_i/\partial \xi_j$ in the Jacobian J of the coordinate transformation

$x_i = x_i(\xi_j)$, $j = 1, 2, 3$:

$$J = \det\left(\frac{\partial x_i}{\partial \xi_j}\right) = \begin{vmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2} & \frac{\partial x_1}{\partial \xi_3} \\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2} & \frac{\partial x_2}{\partial \xi_3} \\ \frac{\partial x_3}{\partial \xi_1} & \frac{\partial x_3}{\partial \xi_2} & \frac{\partial x_3}{\partial \xi_3} \end{vmatrix} . \tag{7}$$

Equation (4) differs from Eq. (3) in that the velocities U_j, which are a linear combination of the Cartesian velocity components u_j, replace the latter in the convective term, and that in the diffusive term, mixed derivatives appear. The mixed derivatives disappear when the grid becomes orthogonal, even if it is curvilinear; the coefficients B^{mj} with unequal indices become than zero. When the grid is only mildly non-orthogonal, the terms involving mixed (or "cross") derivatives are smaller than the terms involving "normal" derivatives. However, they may become dominant if the grid non-orthogonality is high and the aspect ratio of the grid is large (i.e. if $\Delta\xi_1 > \Delta\xi_2$ by a factor of two or more). This affects both the computational effort and the accuracy of the solution, as will be discussed below.

In spite of the above differences, the discretization techniques are basically the same for both non-orthogonal and orthogonal grids. We shall consider here the general case of the diffusion coefficient being a derived variable. This is the case when turbulent flows are calculated using some kind of an eddy-viscosity model; the turbulent diffusion coefficient may than vary in space by as much as three orders of magnitude. Thus, what would be a second derivative in the case of constant Γ appears as an "outer" first derivative (which stems from the divergence operator) applied to an "inner" first derivative (which stems from the gradient operator) multiplied by the diffusion coefficient. This leads naturally to an appropriate discretization approach.

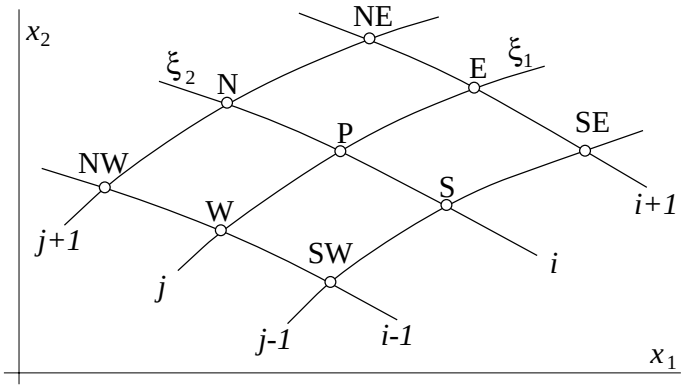

Fig. 2: An example of a non-orthogonal FD–grid and the notation used

In FD methods, the grid lines are associated with coordinate lines, as mentioned above. We shall consider a 2D grid and the notation shown in Fig. 2; the extension to 3D is straightforward. The Eq. (4) is approximated at each grid point by replacing the derivatives with appropriate finite approximations. These approximations should be consistent (i.e. in the limit of the mesh spacing becoming zero, the approximations should become exact) and lead to a numerical method that converges towards the exact solution of the differential equation as the mesh

size reduces. More details on the choice of approximations and their properties can be found in standard textbooks (Anderson et al., 1984; Hirsch, 1992; Fletcher, 1991; Ferziger and Perić, 2002); here only some of the most often used approximations are given.

One possibility of approximating the derivative of ϕ at a grid point is to fit a "shape function" (usually a polynomial) through that grid point and a number of neighbors, and differentiate that function. One usually uses one-dimensional polynomials along each grid line to approximate the derivative in the particular direction. Here ξ is used to denote any of the grid lines ξ_1, ξ_2, or ξ_3.

The simplest shape function is piece-wise linear, assuming that ϕ varies linearly from node to node. The problem is that at any grid point the slope is different on either side; thus, at the grid point identified by an index i, see Fig. 2, one can have:

$$\left(\frac{\partial\phi}{\partial\xi}\right)_i \approx \frac{\phi_i - \phi_{i-1}}{\xi_i - \xi_{i-1}} \ , \tag{8}$$

which is called *backward differencing scheme* (BDS), or:

$$\left(\frac{\partial\phi}{\partial\xi}\right)_i \approx \frac{\phi_{i+1} - \phi_i}{\xi_{i+1} - \xi_i} \ , \tag{9}$$

which is called *forward differencing scheme* (FDS). Obviously, for any non-linear variation of ϕ, both of these approximations will be rather inaccurate. However, these approximations are still used (in some commercial codes as well), because they are numerically stable. Which of the two variants is selected, depends usually on the flow direction: the point on the upwind-side is used in addition to point at ξ_i. This choice is based on the physical argument (when the convective term is considered) that the state at any point is by virtue of convective transport influenced only by what happens upstream of it. These approximations are not recommended – except for local blending with higher-order schemes in order to avoid oscillations in the solution near discontinuities, e.g. near shocks – because they introduce an error known as *numerical* or *false diffusion*. This feature will be discussed further below.

Assuming a parabolic profile passed through three points, one obtains the following approximation on grids with a uniform spacing:

$$\left(\frac{\partial\phi}{\partial\xi_1}\right)_i \approx \frac{\phi_{i+1} - \phi_{i-1}}{\xi_{i+1} - \xi_{i-1}} \ , \tag{10}$$

which is called *central differencing scheme* (CDS). When the grid is non-uniform, the approximation involves the ratio of mesh spacing and also the value ϕ_i; see Ferziger and Perić (2002) for details. This approximation is much more accurate than the above two; it is exact if the variable ϕ varies linearly or quadratically.

By using more points and polynomials of higher degree, more accurate approximations can be obtained. Some will be presented below.

Another possibility for developing approximations to derivatives is to use Taylor series expansion around ξ_i:

$$\begin{aligned} \phi(\xi) \ &= \ \phi(\xi_i) + (\xi - \xi_i)\left(\frac{\partial\phi}{\partial\xi}\right)_i + \frac{(\xi - \xi_i)^2}{2!}\left(\frac{\partial^2\phi}{\partial\xi^2}\right)_i + \\ &\quad \frac{(\xi - \xi_i)^3}{3!}\left(\frac{\partial^3\phi}{\partial\xi^3}\right)_i + \cdots + \frac{(\xi - \xi_i)^n}{n!}\left(\frac{\partial^n\phi}{\partial\xi^n}\right)_i + H \ , \end{aligned} \tag{11}$$

where H means "higher order terms". By replacing ξ by ξ_{i+1}, ξ_{i-1}, etc., one can express the variable values at these points in terms of the variable and its derivatives at ξ_i. From these expansions, one can obtain approximations for the first and higher derivatives at point ξ_i in terms of the function values at neighboring points. For example, BDS is obtained by using the expression (11) for ξ_{i-1}, and FDS when ξ_{i+1} is used instead; most of the terms in the series are neglected, though. CDS is obtained when the expression for ξ_{i-1} is subtracted from the series for ξ_{i+1}; the result is:

$$\left(\frac{\partial\phi}{\partial\xi}\right)_i = \frac{\phi_{i+1}-\phi_{i-1}}{\xi_{i+1}-\xi_{i-1}} - \frac{(\xi_{i+1}-\xi_i)^2-(\xi_i-\xi_{i-1})^2}{2\,(\xi_{i+1}-\xi_{i-1})}\left(\frac{\partial^2\phi}{\partial\xi^2}\right)_i - \frac{(\xi_{i+1}-\xi_i)^3+(\xi_i-\xi_{i-1})^3}{6\,(\xi_{i+1}-\xi_{i-1})}\left(\frac{\partial^3\phi}{\partial\xi^3}\right)_i + H\ . \tag{12}$$

The CDS approximation uses only the first term on the right-hand side; the remainder represents the *truncation error;* it is a measure of the accuracy of approximation and determines the rate at which the error decreases when the mesh spacing is reduced. The truncation error is proportional to the product of the mesh spacing to the power m, $m+1$ etc., and derivatives of ϕ higher than the one being approximated. Usually, the term involving the smallest exponent m is the dominant one, and one can say that the approximation is of mth order.

Although the order of an approximation is an important measure of its accuracy, one has to be careful: two methods may be of the same order but the errors on a given grid may differ by as much as an order of magnitude! Also, the order tells us how fast the error reduces when the grid is refined, and not how big the error is on a given grid. See Roache (1994) or Ferziger and Perić (1996) about how the order can be determined and used to estimate the discretization errors.

In some cases, as in Eq. (12), the leading truncation error term becomes zero when the grid is uniform. However, the error is primarily a function of the mesh spacing and the spatial variation of ϕ, and then of the expansion (non-uniformity) factor. The purpose of using non-uniform grids is to reduce the mesh spacing where the variable ϕ varies most, while leaving large spacing in regions of small variation. Thus, if the same number of uniformly spaced grid points were used in the solution domain, the errors would be much larger since the mesh spacing would then be increased in regions of strong variable variation and decreased in regions where errors are small anyway. It can be shown (see Ferziger and Perić, 2002, for a demonstration) that, when the grid is systematically refined, the error reduces at the same rate on a non-uniform and on a uniform grid when CDS approximation is used.

Taylor series is used to determine the truncation error even when the approximations are derived from polynomial fitting. Below are given two third-order approximations obtained by fitting a cubic polynomial to four points and a fourth-order approximation obtained by fitting a polynomial of degree four to five points on a uniform grid:

$$\left(\frac{\partial\phi}{\partial\xi}\right)_i = \frac{2\,\phi_{i+1}+3\,\phi_i-6\,\phi_{i-1}+\phi_{i-2}}{6\,\Delta\xi} + \mathcal{O}((\Delta\xi)^3)\ ; \tag{13}$$

$$\left(\frac{\partial\phi}{\partial\xi}\right)_i = \frac{-\phi_{i+2}+6\,\phi_{i+1}-3\,\phi_i-2\,\phi_{i-1}}{6\,\Delta\xi} + \mathcal{O}((\Delta\xi)^3)\ ; \tag{14}$$

$$\left(\frac{\partial\phi}{\partial\xi}\right)_i = \frac{-\phi_{i+2} + 8\,\phi_{i+1} - 8\,\phi_{i-1} + \phi_{i-2}}{12\,\Delta\xi} + \mathcal{O}((\Delta\xi)^4)\ . \tag{15}$$

These approximations are called third-order BDS, third-order FDS, and fourth-order CDS, respectively. When the grid is non-uniform, the coefficients in the above expressions become functions of grid expansion ratios. The two third-order schemes are also called *third-order upwind schemes;* the switch from one to the other is dependent on the flow direction – two points are always used on the upstream and one point on the downstream side of the point at ξ_i.

Many more approximations can be developed by using more points or different polynomials or shape functions (even multi-dimensionally). Especially for uniformly spaced grid points, many special (compact) schemes can be derived; here only Padé-schemes will be briefly described.

A family of compact centered approximations can be defined as follows:

$$\alpha\left(\frac{\partial\phi}{\partial x}\right)_{i+1} + \left(\frac{\partial\phi}{\partial x}\right)_i + \alpha\left(\frac{\partial\phi}{\partial x}\right)_{i-1} = a_1\frac{\phi_{i+1} - \phi_{i-1}}{2\,\Delta x} + a_2\frac{\phi_{i+2} - \phi_{i-2}}{4\,\Delta x}\ . \tag{16}$$

Depending on the choice of parameters α, a_1, and a_2, the second-order CDS, the fourth-order CDS, the fourth-order Padé or the sixth-order Padé scheme is obtained; the parameters and the corresponding truncation errors are listed in Table 1.

Table 1: Compact schemes: the parameters and truncation errors

Scheme	Truncation error	α	a_1	a_2
CDS-2	$\frac{(\Delta x)^2}{3!}\frac{\partial^3\phi}{\partial x^3}$	0	1	0
CDS-4	$\frac{13(\Delta x)^4}{3\cdot 3!}\frac{\partial^5\phi}{\partial x^5}$	0	$\frac{4}{3}$	$-\frac{1}{3}$
Padé-4	$\frac{(\Delta x)^4}{5!}\frac{\partial^5\phi}{\partial x^5}$	$\frac{1}{4}$	$\frac{3}{2}$	0
Padé-6	$\frac{4(\Delta x)^6}{7!}\frac{\partial^7\phi}{\partial x^7}$	$\frac{1}{3}$	$\frac{14}{9}$	$\frac{1}{9}$

The standard central-difference schemes use variable values at two or four neighbor points to compute the derivative at a point. In Padé schemes, the order of the approximation is increased by two while keeping the same computational molecule compared to CDS-schemes. This is achieved by using the derivatives at near neighbors instead of variable values at more distant nodes; the computational molecule is thus smaller, but one has to solve an equation system to compute derivatives at grid nodes using known values of the variable ϕ. This is not a problem when using explicit time-integration methods; however, in implicit methods some special treatment is required if Padé schemes are to be used.

One approach is to use the so-called *deferred correction* (which shall be referred to very often further below):

$$\left(\frac{\partial\phi}{\partial x}\right)_i = a_1\frac{\phi_{i+1}-\phi_{i-1}}{2\,\Delta x} + a_2\frac{\phi_{i+2}-\phi_{i-2}}{4\,\Delta x} - \alpha\left(\frac{\partial\phi}{\partial x}\right)_{i+1}^{\text{old}} - \alpha\left(\frac{\partial\phi}{\partial x}\right)_{i-1}^{\text{old}} . \tag{17}$$

Here, only the derivative at point i is expressed through the unknown variable values, while the derivatives at neighbor nodes are explicitly computed using values from the previous iteration. When the iterations converge, the old values will be equal to the current ones and the correct expression corresponding to Eq. (16) will be obtained. However, this approach may affect the convergence rate, since the expression is not quite balanced: the "implicit" part on the right-hand side represents more than the derivative on the left-hand side. The following kind of deferred correction may be more efficient:

$$\left(\frac{\partial\phi}{\partial x}\right)_i = \frac{\phi_{i+1}-\phi_{i-1}}{2\,\Delta x} + \left[\left(\frac{\partial\phi}{\partial x}\right)_i^{\text{Padé}} - \frac{\phi_{i+1}-\phi_{i-1}}{2\,\Delta x}\right]^{\text{old}} . \tag{18}$$

Here, the second-order CDS is used as an implicit approximation, thus limiting the computation molecule to the nearest neighbors only (meaning that less storage space for the matrix coefficients and less computing time for solving the linear algebraic equation systems is needed). On the right-hand side we have the explicitly computed derivative from the Padé scheme (which is what we want to have at the end) and from it is subtracted the explicitly computed lower-order CDS-approximation of the derivative (which we used in the implicit approximation). This gives a more balance expression, since in regions where the second-order CDS is already accurate, the term in square brackets will be negligible.

These compact schemes are limited to regular grids and can not be used in general-purpose codes. Actually, most general-purpose codes use second-order methods, which appear to offer the best compromise between accuracy, complexity of coding, and efficiency when used for computing practical flows. Higher-order methods are in place when very low discretization errors are required (e.g. in DNS). When turbulence models are used in conjunction with RANS equations, the errors due to turbulence model are usually of the order of few per cent, so using methods of high order and reducing the discretization errors much below 1 % is not practical, except when turbulence models are tested.

The diffusive term requires that the differentiation is performed twice. When the diffusion coefficient is not constant, as is often the case, the first derivative is multiplied by it before the product is differentiated again. Any of the approximations shown above – and many others – can be used for each of the differentiation steps; they even don't have to be necessarily the same for the two steps. It is also possible to calculate first derivatives at imaginary nodes placed between grid points (identified here by indices $i+\frac{1}{2}$ and $i-\frac{1}{2}$), and then evaluate the second derivative using these intermediate values; for example, CDS leads to:

$$\left[\frac{\partial}{\partial\xi}\left(\Gamma\frac{\partial\phi}{\partial\xi}\right)\right]_i \approx \frac{\left(\Gamma\dfrac{\partial\phi}{\partial\xi}\right)_{i+\frac{1}{2}} - \left(\Gamma\dfrac{\partial\phi}{\partial\xi}\right)_{i-\frac{1}{2}}}{\dfrac{1}{2}(\xi_{i+1}-\xi_{i-1})} = \frac{\Gamma_{i+\frac{1}{2}}\dfrac{\phi_{i+1}-\phi_i}{\xi_{i+1}-\xi_i} - \Gamma_{i-\frac{1}{2}}\dfrac{\phi_i-\phi_{i-1}}{\xi_i-\xi_{i-1}}}{\dfrac{1}{2}(\xi_{i+1}-\xi_{i-1})} . \tag{19}$$

Mixed derivatives occur in conservation equations only when they are transformed into non-orthogonal coordinate systems, see Eq. (4). For example, one needs to compute:

$$\frac{\partial}{\partial \xi_1}\left(\frac{\Gamma}{J}\frac{\partial \phi}{\partial \xi_2}B^{21}\right) \ . \tag{20}$$

If CDS is used, one would first express the first derivative with respect to ξ_2 at nodes E and W as (see Fig. 2):

$$\left(\frac{\partial \phi}{\partial \xi_2}\right)_{\rm E} \approx \frac{\phi_{\rm NE}-\phi_{\rm SE}}{\xi_{2,\rm NE}-\xi_{2,\rm SE}} \quad \text{and} \quad \left(\frac{\partial \phi}{\partial \xi_2}\right)_{\rm W} \approx \frac{\phi_{\rm NW}-\phi_{\rm SW}}{\xi_{2,\rm NW}-\xi_{2,\rm SW}} \ . \tag{21}$$

For the second differentiation, one multiplies the above approximations by $\Gamma B^{21}/J$ evaluated at nodes E and W, respectively, takes the difference and divides by $\xi_{1,\rm E}-\xi_{1,\rm W}$.

The approximation of mixed derivatives using standard techniques is easy, but when the grid is severely non-orthogonal (angle between grid lines smaller than 45°) and the mesh spacing in one direction is much larger than in the other direction (say more than a factor of four), the approximations like the one described above can lead to unphysical, oscillatory solutions (with undershoots and overshoots). Some authors have suggested modifications which partly overcome these problems; avoiding the use of nodes in "sharp corners" of the computational molecule like the one shown in Fig. 2 helps (see Demirdžić, 1987). For example, in a situation like that of Fig. 2, a more robust approximation of the expression (20) is obtained by replacing the first derivatives in Eqs. (21) by:

$$\left(\frac{\partial \phi}{\partial \xi_2}\right)_{\rm E} \approx \frac{\phi_{\rm NE}-\phi_{\rm E}}{\xi_{2,\rm NE}-\xi_{2,\rm E}} \quad \text{and} \quad \left(\frac{\partial \phi}{\partial \xi_2}\right)_{\rm W} \approx \frac{\phi_{\rm W}-\phi_{\rm SW}}{\xi_{2,\rm W}-\xi_{2,\rm SW}} \ . \tag{22}$$

After CDS is applied in the second step, the final expression contains only the values at nodes NE, E, W, and SW – the nodes SE and NW are not used. However, the approximations of the first derivative at E and W are first-order FDS and BDS, respectively, so the numerical stability is improved at the expense of accuracy.

Special attention is needed at nodes next to solution domain boundaries if more than nearest neighbor nodes are used in the approximations. In such a case one-sided approximations of higher-order have to be used. By fitting a fourth-order polynomial through the boundary and four inner points, ϕ_1 to ϕ_5, the following one-sided fourth-order approximation for the first derivative at the first interior point results:

$$\left(\frac{\partial \phi}{\partial \xi}\right)_2 = \frac{-\phi_5+6\,\phi_4+18\,\phi_3+10\,\phi_2-33\,\phi_1}{60\,\Delta\xi}+\mathcal{O}((\Delta\xi)^4) \ . \tag{23}$$

The boundary conditions are either of Dirichlet or Neumann type, i.e. either the variable value or its derivative (usually in the direction normal to boundary) is specified at boundary points. If the boundary values are known, they are used as such in the approximations and contribute to the right-hand side of the equation system to be solved. If the gradient is prescribed, it is used to eliminate the boundary value as unknown in the approximations at near-boundary grid points. For example, using polynomials of degree two or three, passed through the boundary and two or three interior nodes, the following second and third order approximations for the derivative at the boundary are obtained:

$$\left(\frac{\partial \phi}{\partial \xi}\right)_1 \approx \frac{-\phi_3+4\,\phi_2-3\,\phi_1}{2\,\Delta\xi}+\mathcal{O}((\Delta\xi)^2) \ ; \tag{24}$$

$$\left(\frac{\partial\phi}{\partial\xi}\right)_1 \approx \frac{2\,\phi_4 - 9\,\phi_3 + 18\,\phi_2 - 11\,\phi_1}{6\,\Delta\xi} + \mathcal{O}((\Delta\xi)^3)\;; \tag{25}$$

From these approximations one can express the boundary value ϕ_1 through the interior points as follows:

$$\phi_1 = \frac{4}{3}\phi_2 - \frac{1}{3}\phi_3 - \frac{2}{3}\left(\frac{\partial\phi}{\partial\xi}\right)_1 \Delta\xi\;, \tag{26}$$

$$\phi_1 = \frac{2}{11}\phi_2 - \frac{9}{11}\phi_3 + \frac{18}{11}\phi_2 - \frac{6}{11}\left(\frac{\partial\phi}{\partial\xi}\right)_1 \Delta\xi\;, \tag{27}$$

In the approximations at inner grid nodes, when a reference to ϕ_1 is made, the above replacement formulae are used; the boundary values are thus eliminated as unknowns.

The source term is evaluated at the point P. If it depends on the variable ϕ, it may be expressed through the unknown solution at that point. If the dependence is non-linear, some kind of linearization is necessary. Since the solution method is inevitably of iterative nature, either Picard or Newton iteration on the non-linear source term can be used; see Ferziger and Perić (2002) for more details.

The conservation equations are approximated at interior grid points only; boundary nodes serve for the specification of the boundary conditions, which make the solution unique. There must be no more unknowns than equations, i.e. only the variable values at the interior nodes may appear in the approximations as the unknowns. When appropriate approximations are applied to all terms of the conservation equation at a given point P, and when all non-linear terms are linearized in a suitable way, an algebraic (linear) equation of the following form results:

$$A_\mathrm{P}\phi_\mathrm{P} + \sum_l A_l\phi_l = Q_\mathrm{P}\;, \tag{28}$$

where index l runs over the neighbor nodes of node P represented in the computational molecule. Note that more nodes may be used in the individual approximations, but if the above described deferred-correction approach is used, some contributions may be treated explicitly so that they enter the source term Q_P. The coefficients A_P and A_l depend on the approximations employed, mesh spacing, fluid velocity, and fluid properties.

The algebraic equations for all interior nodes can be written in matrix form as:

$$A\boldsymbol{\phi} = \mathbf{Q}\;. \tag{29}$$

Here, A is the square $N \times N$ matrix, where N is the number of unknowns, while $\boldsymbol{\phi}$ and $\mathbf{Q}$ represent the vectors of unknowns and source terms, respectively. When the grid is structured, as is usually the case in FD methods, the matrix A has diagonal structure: all non-zero elements are located on the main diagonal and a number of other diagonals, depending on the computational molecule of the discretization scheme. This special feature can be exploited to construct some very efficient solution algorithms. Some methods which can be used to solve this equation system will be introduced later.

A few more remarks on FD methods are in place. Firstly, the coordinate transformation can be completely hidden (i.e. one does not have to assign any values to the coordinates ξ_i, but

can use the Cartesian coordinates of grid points instead). All one needs to do is to construct non-overlapping control volumes around each grid node and calculate their volume ΔV. Since $\mathrm{d}V = \mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = J\,\mathrm{d}\xi_1\,\mathrm{d}\xi_2\,\mathrm{d}\xi_3$, it follows that

$$\Delta V = J\Delta\xi_1\Delta\xi_2\Delta\xi_3 \ . \tag{30}$$

If one now multiplies the whole equation by $\Delta\xi_1\Delta\xi_2\Delta\xi_3$, the mesh spacing will disappear in all terms except those containing the Jacobian J, but due to the above expression, one can replace the resulting product by ΔV. The terms involving coordinate derivatives (βs and Bs) loose the mesh spacing in the transformed space and only the differences in Cartesian coordinates remain, e.g. in 2D:

$$\beta_{\mathrm{P}}^{11} = \left(\frac{\partial x_2}{\partial \xi_2}\right)_{\mathrm{P}} = \frac{y_{\mathrm{N}} - y_{\mathrm{S}}}{2\,\Delta\xi_2} \ , \quad \beta_{\mathrm{P}}^{12} = \left(\frac{\partial x_1}{\partial \xi_2}\right)_{\mathrm{P}} = \frac{x_{\mathrm{N}} - x_{\mathrm{S}}}{2\,\Delta\xi_2} \ . \tag{31}$$

The mesh spacing $\Delta\xi_2$ cancels out upon multiplication of the equation by $\Delta\xi_1\Delta\xi_2\Delta\xi_3$, so one only needs the Cartesian coordinates of the grid points.

Finally, FD can in principle be applied to unstructured grids as well. Actually, one does not have to define any grid at all – all one needs are suitably distributed points in space. One can then, in a pre-processing step, define computational molecules or clouds by assigning a certain number of neighbor points to each interior point. By fitting a multi-dimensional shape function to variable values at all points from the molecule, one can obtain approximations to the derivatives with respect to the Cartesian coordinates at the central point, without the need to perform coordinate transformations. No such methods seem to have been developed so far for computing turbulent flows, but similar methods have been used for some special applications. It appears that it would be easier to construct a high-order FD-method than a FV-method of the same order when unstructured, arbitrary grids are used; the reason will become obvious soon.

2.2 Finite-Volume Methods

The finite-volume (FV) methods use the conservation equation in integral form as the starting point, see Eq. (2). The solution domain needs to be subdivided into a finite number of non-overlapping control volumes (CVs). The edges of CVs are usually defined by a grid which may be either structured, block-structured, or unstructured. Figure 3 shows an example of a 2D block-structured grid, which may also be seen as an unstructured grid generated block-wise using a method for the generation of structured grids. The variables are usually defined at the centers of the CVs, except at boundaries where they are defined at the centers of the CV faces. The integral conservation equation is approximated on each CV, resulting in one algebraic equation per CV.

The grid lines and surfaces defining the CVs may be curved. However, when Cartesian vector and tensor components are used, the equations contain no curvature terms and the curvature of grid lines (i.e. CV-edges) is not important; they are usually assumed to be straight. The grid lines in a FV method are not associated with any coordinate directions and therefore any number of grid lines may cross at any CV corner. The CVs can also have an arbitrary shape, although in practice mostly CVs with up to eight corners are used, i.e. tetrahedra, pyramids, prisms and

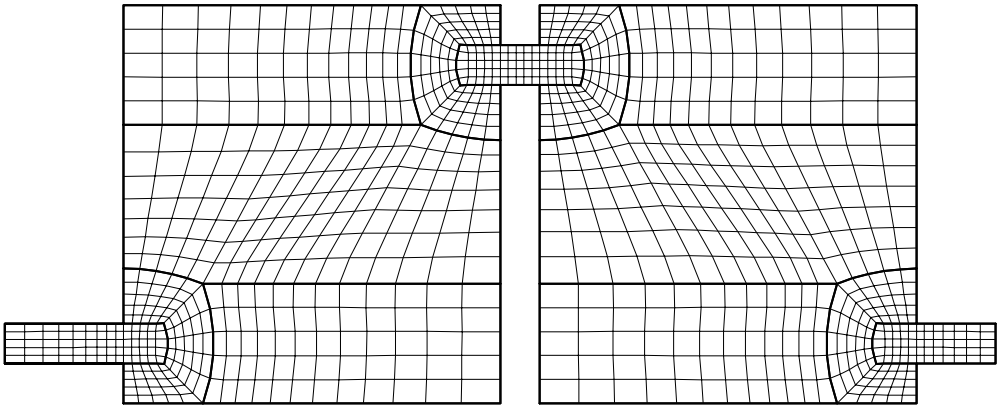

Fig. 3: An example of a 2D block-structured grid, consisting of 13 blocks.

hexahedra. Actually, for the sake of compatibility with CAD tools and commercial software for grid generation, it is a common convention to define CVs always by a list of eight vertices ordered in a certain way. If the CV is not a hexahedron, then some of the vertices collapse, as shown in Fig. 4.

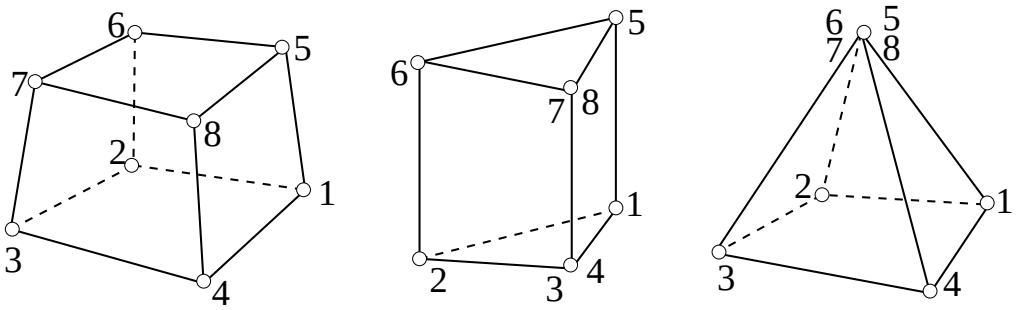

Fig. 4: On the definition of CVs by a list of eight vertices.

In FV methods, two levels of approximation are necessary:

- The integrals over surface and volume of a CV need to be evaluated by a suitable numerical approximation, which uses the value of the integrand at one or more locations within the integration domain;
- Since the variable values are calculated at CV center only, values at other locations which are required for the evaluation of integrals have to be obtained by interpolation; also, derivatives of certain quantities may be required, which makes numerical differentiation also necessary.

Some of the most frequently used approximations for each step are described below.

Approximation of Surface Integrals

The integration must be performed over a closed surface enclosing the CV. The surface of any kind of CV defined by a certain number of vertices which are joined by straight lines can be

decomposed into a certain number of sub-surfaces, enclosed by a polygon of straight lines connecting the vertices at corners. The number of sub-surfaces or cell-faces can range from four (in the case of tetrahedron-CVs) to any greater number; however, usually up to six faces (for hexahedron-CVs) are involved. Each CV face is common to two CVs and is thus uniquely defined for both; only the normal unit vector directed outwards changes sign when the cell face is viewed from either CV. It is therefore sufficient to describe the approximation of the surface integral over one such face; the integral over the whole surface is equal to the sum of integrals over all faces, and the same approximation can be applied to all faces if appropriate substitutions of indices are made, i.e.:

$$F = \oint_S \mathbf{f} \cdot \mathbf{n} \, \mathrm{d}S = \sum_k \int_{S_k} \mathbf{f} \cdot \mathbf{n} \, \mathrm{d}S \ , \tag{32}$$

where index k runs over all faces of one CV and **f** stands for the convective or diffusive flux vector, $\rho\phi\mathbf{v}$ or $\Gamma\boldsymbol{\nabla}\phi$, respectively.

The simplest approximation of the surface integral is provided by the midpoint rule:

$$F_k = \int_{S_k} \mathbf{f} \cdot \mathbf{n} \, \mathrm{d}S \approx (\mathbf{f} \cdot \mathbf{n})_k S_k = \sum_i f_k^i S_k^i \ , \tag{33}$$

where S_k^i is the area of the projection of cell face k onto the Cartesian coordinate surface $x_i =$ const. (or, in other words, the ith Cartesian component of the surface vector $S\mathbf{n}$), and f_k^i is the ith Cartesian component of the vector **f** at the center of the cell face. Thus, the values of f_k^i are taken to represent the mean value over the whole cell face, an approximation which is exact if the variation of the function is linear. The truncation error of this integral approximation (i.e. when the *exact* values of f_k^i are used) is proportional to the square of the mesh spacing, i.e. it is a *second-order approximation*. This is the most widely used approximation; its second order makes it accurate enough for most engineering applications, and this fact with its simplicity is the best argument for adopting it.

One of the nice features of the midpoint-rule approximation is that it is applicable to cell faces of any shape; one only needs to evaluate the integrand at one location, and even if the location at which f_k^i are evaluated does not fall exactly at the cell-face center, it will still be nearly second-order. Therefore, for unstructured grids as well as for structured grids, the above approximation is the best choice if second-order accuracy is acceptable.

Higher-order integral approximations are not so easy to develop for an arbitrary CV shape. For regular structured grids (quadrilaterals in 2D and hexahedra in 3D), one can come up with efficient methods of higher order. Especially in 2D is this task relatively easy: if one uses the Simpson rule, it is necessary to evaluate the integrand at three locations per cell face, i.e. (see Fig. 5):

$$F_\mathrm{e} = \int_{S_\mathrm{e}} \mathbf{f} \cdot \mathbf{n} \, \mathrm{d}S \approx \frac{S_\mathrm{e}}{6} \left[(\mathbf{f} \cdot \mathbf{n})_\mathrm{ne} + 4 \, (\mathbf{f} \cdot \mathbf{n})_\mathrm{e} + (\mathbf{f} \cdot \mathbf{n})_\mathrm{se} \right] \ . \tag{34}$$

Since the corners are common to two CVs, there are actually two evaluations of $(\mathbf{f} \cdot \mathbf{n})$ per cell face: its center and one corner. This approximation is *forth order* accurate; it is very simple, but in order to retain the fourth-order accuracy, the values of the integrand need to be accurately interpolated from the nodal values, which is not so simple, as will be shown

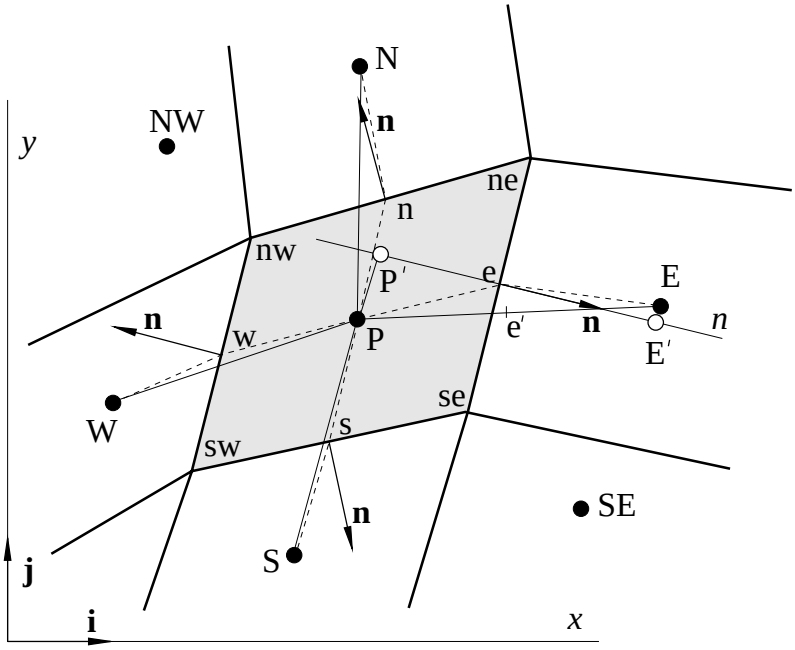

Fig. 5: A typical 2D CV and the notation used.

below. Lilek and Perić (1995) analyzed this approach and found that it is efficient if very high accuracy is required. As with all high-order methods, oscillatory solutions are obtained (and the convergence of the iterative solution procedure may be difficult) if the grid is too coarse. Thus, for a moderate accuracy a second-order method as the midpoint rule may turn out to be a better choice.

In three dimensions, higher-order methods require evaluation of the integrand at many more locations within the face; for example, a fourth-order approximation for a face of a hexahedron-CV requires evaluation of $(\mathbf{f} \cdot \mathbf{n})$ at nine locations, which increases the complexity of the discretization substantially. Especially since – in addition to a more complicated integral approximation – also more complicated interpolation and differentiation are required, the development of high-order FV methods is more elaborate than the development of FD methods of the same order; the latter require only high-order approximations of the derivatives at grid points. Especially if arbitrary polyhedral CVs are considered, methods of high order (higher than second) are difficult to construct.

The accuracy of integral approximation can be increased by subdividing the cell face in smaller pieces and applying midpoint rule at the center of each sub-face. This approach is simple and only requires a more sophisticated interpolation; with the help of the earlier described deferred-correction approach it can be easily implemented by computing the integrals over sub-faces explicitly and using a simple midpoint-rule approximation for the whole face to build the coefficient matrix. This approach could be applied to arbitrary polyhedral CVs, whose polygonal faces could be easily decomposed in a number of triangles.

Approximation of Volume Integrals

The simplest method to approximate a volume integral is again the midpoint rule:

$$\int_V q_\phi \, \mathrm{d}V \approx (q_\phi)_\mathrm{P} \Delta V \ , \tag{35}$$

where $(q_\phi)_\mathrm{P}$ stands for the value of the specific source term at the CV center, i.e. at the computational node P. Thus, no interpolation is necessary to evaluate the integrand (the differentiation may, however, be necessary, depending on the particular expression for q_ϕ, which often does involve gradients of some quantities; for example, source terms in equations for most variables describing turbulence, like k, ϵ, ω etc., involve velocity gradients).

This approximation is also of second order if the point P lies in the center of the volume ΔV, which most often is the case (there are methods which define the computational nodes at the crossings of grid lines, and then construct control volumes around each node; the node may then not lie in the center of the CV, but the cell faces lie midway between the neighbor nodes). Higher-order approximations are, as outlined above, more difficult to derive and are seldom used.

Interpolation Schemes

There are many methods to choose from when interpolation is considered. The crudest approximation is to assume that the cell-face value is equal to the cell-center value on one side, depending on the flow direction. This step-wise interpolation is called *upwind differencing scheme* (UDS), by analogy to the upwind-differencing of convective term in FD. It is first-order accurate, introduces excessive numerical diffusion error, and should therefore be avoided.

Another simple and popular approach is linear interpolation. If the grid is structured, one usually interpolates in each direction independently and calls the interpolation bi-linear (in 2D) or tri-linear (in 3D). When the grid is unstructured, one either uses some kind of linear shape functions, or interpolates by using the gradient vector evaluated at the CV center:

$$\phi_r = \phi_\mathrm{P} + (\boldsymbol{\nabla}\phi)_\mathrm{P} \cdot (\mathbf{r}_r - \mathbf{r}_\mathrm{P}) \ , \tag{36}$$

where ϕ_r is the interpolated value of ϕ at a location defined by the position vector $\mathbf{r}_r$. This approximation is based on the assumption that the gradient of ϕ is constant within the CV, which corresponds to the assumption of a linear variation of ϕ and is therefore second-order accurate.

In order to calculate the variable values at cell-face centers, one only needs to interpolate between the two nearest neighbors if linear interpolation is used, e.g.:

$$\phi_\mathrm{e} = \frac{|\mathbf{r}_\mathrm{e} - \mathbf{r}_\mathrm{P}|}{|\mathbf{r}_\mathrm{E} - \mathbf{r}_\mathrm{P}|}\phi_\mathrm{E} + \frac{|\mathbf{r}_\mathrm{E} - \mathbf{r}_\mathrm{e}|}{|\mathbf{r}_\mathrm{E} - \mathbf{r}_\mathrm{P}|}\phi_\mathrm{P} \ . \tag{37}$$

However, if the grid lines change their direction at CV corners (which is always the case on unstructured grids made of tetrahedra), linear interpolation according to this expression is second-order accurate at the location denoted by ‘e′’ in Fig. 5, which lies on the straight line connecting nodes P and E, and not at the cell-face center ‘e’. Thus, if ‘e′’ is substantially far away from ‘e’

(relative to the cell-face size), the accuracy of the integration is reduced. If, for example, the location 'e′' falls close to the cell corner 'se', the integral approximation will be only first-order accurate since the value used as an approximation of the mean value in the integration range is actually corresponding to the value at the boundary of the range and not at its center, as required for the second-order accuracy of midpoint rule. If the gradient of ϕ is known at cell centers, one can restore the second-order accuracy by applying a correction to the value resulting from the above simple linear interpolation:

$$\phi_{\mathrm{e}} \approx \phi_{\mathrm{e}'} + \overline{(\boldsymbol{\nabla}\phi)}_{\mathrm{e}'} \cdot (\mathbf{r}_{\mathrm{e}} - \mathbf{r}_{\mathrm{e}'}) \ . \tag{38}$$

The over-bar denotes here interpolated value.

Linear interpolation leads to a second-order approximation of the cell-face center value (for a formal proof, see Ferziger and Perić, 2002). It is equivalent to the *central differencing scheme* (CDS) in FD and is usually referred to under this name. CDS was long believed to be impractical for convection-dominated flows, since it – as any other scheme of the order higher than first – may lead to oscillatory solutions. No oscillations occur when the so called *cell Peclet number*, Pe = $\rho v_n \Delta/\Gamma < 2$, where v_n is the velocity component normal to the cell face and Δ is the distance between the two cell centers on either side of the face. However, this is only a *sufficient*, but not always *necessary* condition for non-oscillatory solutions. Only when the Peclet number is large in a region of strong variable change (high second derivative) is CDS prone to oscillations. Local grid refinement is the cure, not switching to UDS, since the oscillations carry a message: the grid is much too coarse where it should be fine (see Ferziger and Perić, 2002, for some examples).

CDS is suitable for use in conjunction with midpoint rule integral approximation. However, the accuracy is somewhat increased – even though the order of the overall approximation can not be increased above second – when a more accurate interpolation is used. Very popular is the quadratic interpolation on structured grids, which even has its own name: QUICK (Leonard, 1979). Two nodes are used on the upwind side of the cell face and one node on the downstream side, and a parabola is fitted to the nodal values. The resulting value at the cell face is, on a uniformly spaced grid and with the flow directed from P to E:

$$\phi_{\mathrm{e}} = \frac{3}{8}\phi_{\mathrm{E}} + \frac{6}{8}\phi_{\mathrm{P}} - \frac{1}{8}\phi_{\mathrm{W}} \ . \tag{39}$$

When the grid is non-uniform, the coefficients in the above expression become functions of the mesh spacing. When a grid made of triangles or tetrahedra is used, the second upstream node may not be readily available; instead of the variable value at the second upstream point one can use the gradient at the upstream cell center to obtain the third coefficient in the parabola, see below. This interpolation scheme is *third-order accurate*; however, the overall method still remains of *second order* if the interpolated value is used in a midpoint rule approximation of the integral, i.e. the error will asymptotically be reducing by a factor of four when the grid is refined. On a particular grid, the result might be though significantly more accurate than when linear interpolation is used instead (except on coarse grids).

For higher-order integral approximations, like the Simpson rule in 2D, one has to use higher-order interpolation in order to preserve the accuracy of the integral approximation. Thus, one

can retain the fourth-order of the Simpson rule if the interpolation is also at least fourth-order accurate. A suitable approximation results from fitting a polynomial of degree three through four nodes, two on either side of the cell face; on a uniform grid, the result is:

$$\phi_{\mathrm{e}} = \frac{27\,\phi_{\mathrm{P}} + 27\,\phi_{\mathrm{E}} - 3\,\phi_{\mathrm{W}} - 3\,\phi_{\mathrm{EE}}}{48} \ . \tag{40}$$

One can first calculate the values at all cell-face centers using the above kind of interpolation. One can then apply the same interpolation along the cell faces and obtain values at cell corners as a function of cell-face center values, e.g. ϕ_{ne} and ϕ_{se}. The integral approximation according to Eq. (34) can now be performed; the flux F_{e} is then a function of 15 nodal values.

When the grid is unstructured and made of arbitrary CVs, one would have to use more complicated interpolation polynomials. If the gradient-vector components are available at each CV-center (see below), one can construct a polynomial shape function up to degree three by using the variable values and gradient vectors at the two cell-centers on either side of the cell face, i.e. the coefficients a_i in the polynomial

$$\phi = a_0 + a_1\xi + a_2\xi^2 + a_3\xi^3$$

can then be computed. For example, if ξ described the coordinate along the line connecting the nodes P and E, see Fig. 5, we can obtain the four coefficients by fitting the polynomial to ϕ_{P} and ϕ_{E} as well as setting:

$$\left(\frac{\partial\phi}{\partial\xi}\right)_{\mathrm{P}} = a_1 = (\boldsymbol{\nabla}\phi)_{\mathrm{P}} \cdot \mathbf{i}_{\xi} \ ,$$

where $\mathbf{i}_{\xi}$ is the unit vector in the direction of ξ, and using an analogous expression at node E.

Since the convective flux through each cell face depends on 15 nodal values when the above approximation is used (on structured grids), the implicit treatment of all terms would result in an algebraic equation at each CV with 25 unknowns (in 2D). The solution of the resulting equation system for the whole solution domain would thus become prohibitively expensive. One can, however, argue that the flux calculated using the midpoint rule and an interpolation which uses the nearest neighbors only (from CVs which have common faces with the CV around node P) will not be much different from the flux calculated using a higher-order method, unless the variation of the variables is highly non-linear. Also, if one is solving a non-linear equation or a coupled equation system (which the Navier-Stokes equations are), one is forced to use an iterative solution strategy (see below). It then turns out to be both simpler and numerically more efficient to calculate the higher-order flux approximation explicitly (using values from previous iteration) and combine it with an implicit approximation which is numerically stable and uses only nearest neighbors. This leads to another version of the earlier described *deferred-correction* approach (first suggested for this purpose by Khosla and Rubin, 1972):

$$F_{\mathrm{e}} = F_{\mathrm{e}}^{\mathrm{L}} + \left(F_{\mathrm{e}}^{\mathrm{H}} - F_{\mathrm{e}}^{\mathrm{L}}\right)^{\mathrm{old}} \ . \tag{41}$$

Here, $F_{\mathrm{e}}^{\mathrm{L}}$ stands for an approximation by a lower-order scheme (midpoint rule integration with first-order UDS) and $F_{\mathrm{e}}^{\mathrm{H}}$ is the higher-order approximation (either midpoint rule or Simpson's rule integration with higher-order interpolation). The term in brackets is evaluated explicitly

using values from the previous iteration, which is indicated by the superscript 'old'. It is normally relatively small compared to the implicit part, so that its explicit treatment does not slow down the convergence significantly. The term in brackets may also be multiplied by a factor $0 \leq \beta \leq 1$, thus enabling the blending of the two schemes.

It is important that the explicitly computed part of the flux is minimized; one should therefore not simply compute explicitly contributions of more distant nodes, but always try to achieve that the implicit part of the flux is itself a consistent approximation.

The convective fluxes are non-linear. To solve the equation system, linearization is necessary. In most applications the simple Picard iteration is appropriate enough. For example, in the generic conservation equation for property ϕ, we have:

$$F_{\mathrm{e}}^{\mathrm{c}} = \int_{S_{\mathrm{e}}} \rho\phi\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = \overline{\phi}\int_{S_{\mathrm{e}}} \rho\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S \approx \phi_{\mathrm{e}}\dot{m}_{\mathrm{e}}\ . \tag{42}$$

Here a mid-step is introduced in which the mean value of ϕ is first taken out of the integral; the integral of the remaining linear term, denoted $\dot{m}$ and representing the mass flux, is calculated explicitly using values from previous iteration and treated as a known quantity. When the midpoint-rule approximation is used (as in the above equation), the mass flux is expressed as $\dot{m}_{\mathrm{e}} = (\rho v_n S)_{\mathrm{e}}$, where v_n is the velocity component normal to the surface and S_{e} is the area of the face. To complete the formulation, the mean value of ϕ is replaced by the midpoint rule approximation, i.e. by ϕ_{e}, which needs to be further approximated by interpolating the neighboring nodal values. In higher-order approximations, the mean-value approximation involves values at more than one location, like in the above-mentioned Simpson's rule in 2D. Both the mass flux and the mean value of ϕ have to be approximated by higher-order approximations; see Lilek and Perić (1995) for an example of a 4th-order scheme.

Differentiation Schemes

In order to calculate the diffusive flux through a cell face using midpoint rule approximation, one needs the gradient of ϕ at the cell-face center:

$$F_{\mathrm{e}}^{\mathrm{d}} = \int_{S_{\mathrm{e}}} \Gamma\boldsymbol{\nabla}\phi\cdot\mathbf{n}\,\mathrm{d}S \approx (\Gamma\boldsymbol{\nabla}\phi\cdot\mathbf{n})_{\mathrm{e}}\,S_{\mathrm{e}} = \left(\Gamma\frac{\partial\phi}{\partial n}\right)_{\mathrm{e}} S_{\mathrm{e}} = \Gamma_{\mathrm{e}}\sum_i \left(\frac{\partial\phi}{\partial x_i}\right)_{\mathrm{e}} S_{\mathrm{e}}^i\ . \tag{43}$$

When the line connecting two neighbor CV-centers is orthogonal to the cell face (which is true not only for Cartesian grids, but also for curvilinear orthogonal body-fitted grids and grids made of equilateral triangles), the normal derivative can be easily approximated using second-order central differences:

$$\left(\frac{\partial\phi}{\partial n}\right)_{\mathrm{e}} \approx \frac{\phi_{\mathrm{E}} - \phi_{\mathrm{P}}}{|\mathbf{r}_{\mathrm{E}} - \mathbf{r}_{\mathrm{P}}|}\ . \tag{44}$$

When the line connecting neighbor nodes is not orthogonal to the cell face, the computation of the diffusive flux is more complicated.

On structured grids, one can use coordinate transformation to express either the derivatives with respect to Cartesian coordinates, or the derivative with respect to the normal n, through derivatives in the direction of local coordinates aligned with grid lines. Central differences can

then be applied to the derivatives along local coordinates. For the coordinate along the line connecting cell centers, the approximation given in Eq. (44) is obtained at cell-face center; for other two coordinates, the derivatives are evaluated at cell centers and interpolated to the cell face. These so called *cross-derivatives* are usually treated explicitly in order to keep the computational molecule limited to nearest neighbors.

Higher-order derivatives can be obtained by fitting a polynomial of a higher degree through a certain number of points along grid lines. For example, from a cubic polynomial passed through four nodes (two on either side of the cell face), one obtains on a uniform grid:

$$\left(\frac{\partial\phi}{\partial\xi}\right)_{\mathrm{e}} = \frac{27\,\phi_{\mathrm{E}} - 27\,\phi_{\mathrm{P}} + \phi_{\mathrm{W}} - \phi_{\mathrm{EE}}}{24\,\Delta\xi} \ . \tag{45}$$

On unstructured grids with arbitrary CVs, the above approach is not practical. If the spatial variation of ϕ in the vicinity of the location 'e' is described by an analytical shape function, then the derivatives with respect to Cartesian coordinates are easily calculated. Another simple approximation which is up to second-order accurate can be obtained using Gauss' theorem without using complicated shape functions. The derivative at CV center can be evaluated using midpoint-rule approximation of the volume integral as follows:

$$\left(\frac{\partial\phi}{\partial x_i}\right)_{\mathrm{P}} \approx \frac{\int_V \dfrac{\partial\phi}{\partial x_i}\,\mathrm{d}V}{\Delta V} \ . \tag{46}$$

Since the derivative $\partial\phi/\partial x_i$ can be interpreted as divergence of the vector $\phi\,\mathbf{i}_i$, one can transform the volume integral in the above equation into a surface integral:

$$\int_V \frac{\partial\phi}{\partial x_i}\,\mathrm{d}V = \int_S \phi\,\mathbf{i}_i\cdot\mathbf{n}\,\mathrm{d}S \approx \sum_k \phi_k S_k^i \ , \quad k = \mathrm{e},\mathrm{n},\mathrm{w},\mathrm{s},\ldots \tag{47}$$

Therefore, in order to calculate the gradient of ϕ with respect to x at the CV center, one needs to sum the products of ϕ_k with the x-component of the surface vector $S_k\mathbf{n}$ at all CV faces and divide this sum with CV volume:

$$\left(\frac{\partial\phi}{\partial x_i}\right)_{\mathrm{P}} \approx \frac{\sum_k \phi_k S_k^i}{\Delta V} \ . \tag{48}$$

As ϕ_k one can use those values used to calculate the convective flux. In the case of Cartesian grids, this approximation leads to the usual CDS-expression for the first derivative at the cell center. The derivatives calculated this way at CV centers can now be interpolated to cell faces and the diffusive flux can be calculated according to Eq. (43).

Another second-order approximation, which is also applicable to arbitrary grids, involves definition of auxiliary nodes P′ and E′ on the normal passing through the cell face center, see Fig. 5. The values of ϕ at these auxiliary nodes can be expressed by means of Eq. (36) as follows:

$$\phi_{\mathrm{P}'} = \phi_{\mathrm{P}} + (\boldsymbol{\nabla}\phi)_{\mathrm{P}}\cdot(\mathbf{r}_{\mathrm{P}'} - \mathbf{r}_{\mathrm{P}}) \ ; \quad \phi_{\mathrm{E}'} = \phi_{\mathrm{E}} + (\boldsymbol{\nabla}\phi)_{\mathrm{E}}\cdot(\mathbf{r}_{\mathrm{E}'} - \mathbf{r}_{\mathrm{E}}) \ . \tag{49}$$

The derivative with respect to n, which is the only one needed to calculate the diffusive flux, see Eq. (43), can now be approximated by a central difference using Eq. (49) as:

$$\left(\frac{\partial\phi}{\partial n}\right)_{\mathrm{e}} \approx \frac{\phi_{\mathrm{E}} - \phi_{\mathrm{P}}}{|\mathbf{r}_{\mathrm{E}'} - \mathbf{r}_{\mathrm{P}'}|} + \left[\frac{(\boldsymbol{\nabla}\phi)_{\mathrm{E}}\cdot(\mathbf{r}_{\mathrm{E}'} - \mathbf{r}_{\mathrm{E}}) - (\boldsymbol{\nabla}\phi)_{\mathrm{P}}\cdot(\mathbf{r}_{\mathrm{P}'} - \mathbf{r}_{\mathrm{P}})}{|\mathbf{r}_{\mathrm{E}'} - \mathbf{r}_{\mathrm{P}'}|}\right]^{\mathrm{old}} \ . \tag{50}$$

The first term on the right-hand side can be treated implicitly while the second term can be calculated using values from previous iteration. Note that $|\mathbf{r}_{\mathrm{E}'} - \mathbf{r}_{\mathrm{P}'}| = (\mathbf{r}_{\mathrm{E}} - \mathbf{r}_{\mathrm{P}}) \cdot \mathbf{n}$.

Another second-order approximation on any grid can be obtained by assuming linear variation of the variable in the vicinity of the node P and using central-difference approximations of derivatives along lines which connect the node P with neighbor nodes, e.g.:

$$(\boldsymbol{\nabla}\phi)_{\mathrm{P}} \cdot (\mathbf{r}_{\mathrm{E}} - \mathbf{r}_{\mathrm{P}}) \approx \phi_{\mathrm{E}} - \phi_{\mathrm{P}} \ .$$

There are as many such expressions as there are neighbors of node P (six in the case of a hexahedral grid); in all of them, the gradient of ϕ at node P is the only unknown. Thus, there are more equations than unknowns (three components of the gradient vector) so the least-squares method has to be used to compute the gradient; see Demirdžić and Muzaferija (1995) for more details.

The cell-center derivatives can be interpolated to the cell-face centers using the same interpolation technique as used for convective terms. However, oscillatory solutions may develop in this case; see Ferziger and Perić (2002) for detailed discussion. Muzaferija (1994) introduced an effective approach which both avoids de-coupling problems and acts as a deferred correction, preserving the simplicity of the coefficient matrix and its diagonal dominance, e.g. for the face 'e':

$$(\boldsymbol{\nabla}\phi)_{\mathrm{e}} \cdot \mathbf{n}_{\mathrm{e}} \approx \frac{\phi_{\mathrm{E}} - \phi_{\mathrm{P}}}{|\mathbf{r}_{\mathrm{E}} - \mathbf{r}_{\mathrm{P}}|} - \underline{\overline{(\boldsymbol{\nabla}\phi)}_{\mathrm{e}}^{\mathrm{old}} \cdot \left(\frac{\mathbf{r}_{\mathrm{E}} - \mathbf{r}_{\mathrm{P}}}{|\mathbf{r}_{\mathrm{E}} - \mathbf{r}_{\mathrm{P}}|} - \mathbf{n}_{\mathrm{e}} \right)} \ . \tag{51}$$

The underlined term is calculated using prevailing values of the variables and treated as another deferred correction, see above. If the line connecting nodes P and E is orthogonal to the cell face, the underlined term is zero (since the vectors $\mathbf{r}_{\mathrm{E}} - \mathbf{r}_{\mathrm{P}}$ and $\mathbf{n}_{\mathrm{e}}$ are then co-linear) and the approximation reduces to the standard CDS-expression. The explicitly calculated gradient at the cell face (denoted by over-bar in the above expression and obtained by interpolation) only accounts for the cross-derivative. The additional diffusive terms in the momentum equations can be obtained by interpolating the cell-center values, as they cause no problems.

Approximations of the diffusive flux of the order higher than second are difficult to develop for arbitrary CVs. For structured, and especially for orthogonal grids, this task is much easier, but the applicability of the solution method is then limited to geometries which allow generation of such grids.

One should also note that the implementation of turbulence models, especially the more sophisticated ones, requires that the first – and sometimes also the second – derivatives of velocity components be calculated at both CV center and cell faces. The above approximations are applicable to arbitrary CVs and involves no coordinate transformation. This is a very attractive feature, since many turbulence models involve complex equations even when written in Cartesian coordinates; their transformation in non-orthogonal coordinate systems is a painstaking procedure, and the result is not easy to check. The possibility to avoid coordinate transformation should therefore be used wherever available.

Boundary conditions

Some faces of next-to-boundary CVs lie in the boundary surface enclosing the solution domain. The surface integrals over boundary faces – which represent convective and diffusive fluxes through the solution domain boundary – must either be known (if flux boundary conditions are provided) or be expressed in terms of variable values at the boundary and in the interior. Since there may be only as many unknowns as there are CVs, the cell-face values of variables and their gradient – if unknown – must be approximated using extrapolation and one-sided differences.

Usually, convective fluxes are prescribed at inlet. They are zero at impermeable walls and symmetry planes and are usually approximated by upwind schemes at outlet boundaries. Diffusive fluxes are sometimes specified at solid surfaces (e.g., specified heat flux); in this case, they are used directly and, if the wall value of the variable is required, an approximation for the flux in terms of nodal variable values can be used to find it. More often, the value of the variable is prescribed; in this case, the diffusive flux is evaluated using a one-sided approximation for the normal gradient. For a more detailed description of the various boundary conditions, see Ferziger and Perić (2002).

Each CV provides one algebraic equation of the form given by Eq. (28), which represents a linearized and discretized approximation of the conservation equation. It should be noted that, in any conservative method, the following relation holds:

$$A_{\mathrm{P}} = -\sum_{l} A_l + \sum_{k} \dot{m}_k \ , \tag{52}$$

where l runs over neighbor nodes used in the computational molecule and k runs over all cell faces. For incompressible flows, $\sum_k \dot{m}_k = 0$ holds.

The methods used to solve the system of algebraic equations, Eq. (29), will be described below.

2.3 Finite-Element Methods

The finite element (FE) methods are similar to FV methods in many ways. The principal differences are twofold:

- The solution domain is subdivided into a finite number of elements (which may look the same as CVs in FV methods), usually by an unstructured grid. The computational nodes are located at the corners and possibly also at other locations at the surface or within the element. Depending on the number of nodes per element, a suitable shape function is defined which describes the approximated variation of the variable over the element.

- The algebraic equation system from which the unknown nodal values of the variable are obtained can be constructed by different methods. One variant multiplies the equations by a *weight function*, integrates over the entire domain, and requires that the derivative of the integral with respect to each nodal value be zero.

These methods involve more mathematics than the FV methods; they are also more suitable for the mathematical analysis of the properties of the numerical solution method and are therefore

preferred by mathematicians and more mathematically-minded engineers to FV methods. A vast literature exists on FE methods, mostly because they are primarily used in the structural analysis in solid mechanics. Details on FE methods for fluid dynamics can be found in many books, e.g. Oden (1972), Chung (1978), Baker (1983), Girault and Raviart (1986), and Fletcher (1991).

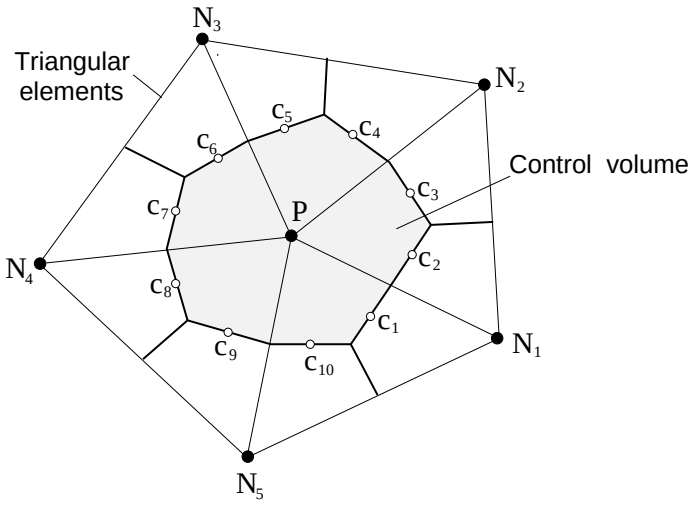

Fig. 6: A control volume in a CVFEM method

A hybrid method called *control-volume-based finite element method* (CVFEM) deserves also to be mentioned. In it, the elements (three-node triangles in 2D) are used to describe the variation of the variables (linear shape functions of the form $\phi = ax + by + c$). The control volumes are formed around each node by joining the centroids of the elements, cf. Fig. 6. The conservation equations in integral form are applied to these CVs in exactly the same way as described above for the FV method. The surface and volume integrals are calculated *element-wise*. For the CV shown in Fig. 6, the CV surface consists of 10 sub-faces, while its volume consists of five sub-volumes, since the node P is common to five elements. Since the variation of ϕ over an element is prescribed in the form of an analytical function, both surface and volume integrals can easily be calculated (i.e. expressed through the unknown values of ϕ at node P and its nearest neighbors, N_1 to N_5 in Fig. 6). Even when the grid consists of triangles only, the number of neighbors may vary from one CV to another, leading to an irregular structure of the coefficient matrix. This restricts the range of solvers which can be used; se below.

This approach was followed – although only in 2D and using second order approximations – by Baliga and Patankar (1983), Schneider and Raw (1987), Masson et al. (1994), Baliga (1997), and others. Its extension to 3D is straightforward, but more complicated.

2.4 Spectral and Spectral-Element Methods

Spectral methods are usually used for specialized applications, especially for LES and DNS of flows in simple geometries. No attempt will be made to describe them here in any more detail but to say that they use Fourier series or generalizations of them to evaluate the spatial derivatives. The grids are uniformly spaced, and the method is especially suited for problems

which allow for periodic boundary conditions to be applied at the opposite boundaries.

The discretization error decreases in spectral methods exponentially with the number of grid points N, making them very accurate if N is sufficiently large (which, as is usually the case with the descriptor *sufficient*, depends on the problem being solved and can not be accurately determined a priori). Interested reader may refer to the book by Canuto et al. (1987) for further details.

The spectral element methods do away with the requirement that the grid must be uniform and structured (conforming), while trying to retain the spectral accuracy. They can use locally refined (non-conforming) grids and are more flexible regarding the geometrical complexity of problems that can be solved than the classical spectral method; see Henderson and Karniadakis (1995) for an example of such a method.

These methods have so far been mostly used to solve the Navier-Stokes equations; they are mathematically more complicated than other methods presented so far and are not so easy to extend to problems which involve additional (coupled) phenomena, which is the reason for their use in a limited range of applications.

3 METHODS FOR INTEGRATION IN TIME

The methods for integration in time can be grouped into two major categories:

- Explicit methods, which calculate the solution at the new time step by using only the variable values from previous time steps;
- Implicit methods, which use in the evaluation of the integral the unknown new values and thus require the solution of an equation system, see Eq. (29).

The explicit methods are thus much simpler and they require less storage and computing time per time step than the implicit methods. However, the explicit methods suffer from instability if the time step is larger than a certain limit. Thus, they are not suitable for problems which do not require (on the grounds of accuracy) small time steps, like periodic flows or time-marching towards steady solutions.

The integration in time can be best explained by re-writing the conservation equations (1) and (2) in the following form:

$$\frac{\partial \psi}{\partial t} = F \ , \tag{53}$$

where both ψ and F are functions of time. The meaning of these two quantities is obvious from Eqs. (1) and (2). At an initial time level, the solution must be known (the initial condition), i.e. $\psi(t_0) = \psi_0$. At the time $t_0 + \Delta t$, the solution is found by integration. It serves than as the initial solution for the integration over the next time step, and so on. In general, to advance the solution from time t_n to time $t_{n+1} = t_n + \Delta t$ at a given point in space, one can write:

$$\int_{t_n}^{t_n+\Delta t} \frac{\partial \psi}{\partial t} \, \mathrm{d}t = \psi(t_n + \Delta t) - \psi(t_n) = \int_{t_n}^{t_n+\Delta t} F(t, \psi(t)) \, \mathrm{d}t \ . \tag{54}$$

In the following, ψ^{n+1} and F^{n+1} will be used to denote the values of ψ and F at the time level t_{n+1}, assuming that F also depends on ψ.

As shown in the above equation, the integral on the left-hand side is easy to evaluate, but the integral on the right-hand side requires an approximation. Many possibilities exist, both explicit and implicit. Basically one has to assume a certain variation of F over the time step Δt and integrate it.

If one uses only the values at the ends of the integration interval, the so called *two-level methods* are obtained. They are described below.

3.1 Two-Level Methods

The simplest approximation of the time integral is obtained by assuming a constant value of F over Δt. If the old (known) value is used, one obtains the *explicit Euler* method:

$$\psi^{n+1} = \psi^n + F^n \Delta t \ . \tag{55}$$

Since all quantities on the right-hand side are known, the new value is directly obtained – hence the name *explicit* method.

If, on the other hand, the new value is used, one has the *implicit Euler* method:

$$\psi^{n+1} = \psi^n + F^{n+1} \Delta t \ . \tag{56}$$

Since F^{n+1} on the right-hand side depends on ψ^{n+1} (not only at the given point in space, but also in the surroundings), explicit calculation of ψ^{n+1} is not possible. F^{n+1} involves the convective and diffusive fluxes at the new time level, as well as the source terms (see Eq. (2)); when these terms are discretized, an algebraic equation results at each grid point. Thus, to calculate the new value of ψ, the equation system (29) must be solved. The main diagonal element of the matrix A receives a contribution from the unsteady term, and the known old value contributes to the source vector $\mathbf{Q}$. One usually divides the whole equation by Δt, which than attains the form of the steady-state equation extended by an approximation of the time-derivative:

$$\frac{\psi^{n+1} - \psi^n}{\Delta t} = F^{n+1} \ . \tag{57}$$

In the case of FV methods, the Eq. (28) can then be re-cast in the following form:

$$\left(A_{\mathrm{P}} + \frac{\rho \Delta V}{\Delta t} \right) \phi_{\mathrm{P}}^{n+1} + \sum_l A_l \phi_l^{n+1} = Q_{\mathrm{P}}^{n+1} + \frac{\rho \Delta V}{\Delta t} \, \phi_{\mathrm{P}}^n \ . \tag{58}$$

Note that, since the conservation equations are in general non-linear, the matrix elements also depend on the new solution.

As can be guessed from the kind of approximation used, both of these Euler methods are first-order accurate. Although the approximation of the time derivative in Eq. (57) looks like a central-difference approximation, it is actually a backward-difference relative to the right-hand side. The approximation in Eq. (55) represents a forward-difference relative to the right-hand side.

The implicit Euler method, owing to its stability (there is virtually no limit on the time step that can be used from the stability point of view), is often used, especially when steady-state solutions are sought. The explicit Euler method is stable only when

$$\frac{\Gamma\,\Delta t}{\rho(\Delta\xi)^2} \leq \frac{1}{2}\ , \tag{59}$$

where $\Delta\xi$ is the mesh spacing in a particular direction. When the mesh spacing is halved, the time step must be reduced by a factor of four to satisfy the above criterion. This is too restrictive and owing to the first-order accuracy, the method is of little use.

One of the most often used two-level approximations, which is second-order accurate, is based on the trapezoid rule:

$$\psi^{n+1} = \psi^n + \frac{1}{2}(F^n + F^{n+1})\Delta t\ . \tag{60}$$

If written in the form of Eq. (57), the equation reveals that the right-hand side is an approximation of the value at the center of the interval by means of linear interpolation, and the left-hand side would then represent the central-difference approximation with respect to the center of the interval. The method is implicit and requires the solution of an equation system. However, the accuracy is of second order and the computing effort and storage requirement are hardly any larger than for the implicit Euler scheme. The method is known under the name *Crank-Nicolson* scheme and is the only two-level method of second order. It should be noted that, although the solution at only one old level needs to be stored, one usually has to store the contribution of old fluxes and source terms, F^n, since these form a part of the final source term in the algebraic equation which does not change during iterations within the time step.

3.2 Multi-Level Methods

Two-level methods cannot have order higher than second; higher-order methods must use information at more time levels. The additional data may be at points at which the solution has already been computed (past data) or at points between t_n and t_{n+1} which are used strictly for computational convenience; the former are called *multipoint methods*, the latter, *Runge-Kutta methods*.

The methods derived by fitting a polynomial to the derivative at a number of points in time are called *Adams methods*. If the derivative at t_{n+1} is not used, explicit or *Adams-Bashforth methods* are obtained. The first-order method is explicit Euler; the second-order one is:

$$\psi^{n+1} = \psi^n + \frac{\Delta t}{2}(3\,F^n - F^{n-1})\ . \tag{61}$$

If data at t_{n+1} is included in the interpolation polynomial, implicit or *Adams-Moulton* methods are obtained. The first-order method is implicit Euler while the second-order one is trapezoid rule. A commonly used method is a combination of the $(m-1)$st-order Adams-Bashforth method as a predictor and the mth-order Adams-Moulton method as a corrector.

A fully implicit scheme of second-order can be obtained by integrating over an interval centered about t_{n+1} and using the mean-value approach:

$$\int_{t_{n+1}-\Delta t/2}^{t_{n+1}+\Delta t/2} \frac{\partial \psi}{\partial t}\,\mathrm{d}t \approx \Delta t \left(\frac{\partial \psi}{\partial t}\right)^{n+1} \approx F^{n+1}\Delta t \ . \tag{62}$$

The derivative at t_{n+1} is approximated by fitting a quadratic polynomial through variable values at three time levels: t_{n+1}, t_n and t_{n-1}. For a constant time step the following approximation is obtained:

$$\left(\frac{\partial \psi}{\partial t}\right)^{n+1} \approx \frac{3\,\psi^{n+1} - 4\,\psi^{n} + \psi^{n-1}}{2\,\Delta t} \ . \tag{63}$$

This scheme is easy to implement as it differs from the first-order implicit Euler scheme only in that an additional term involving the value at t_{n-1} is added:

$$\psi^{n+1} = \frac{4}{3}\,\psi^{n} - \frac{1}{3}\,\psi^{n-1} + \frac{2}{3}\,F^{n+1}\Delta t \ . \tag{64}$$

In the case of FV methods, the Eq. (28) obtains the following form when this scheme is used:

$$\left(A_{\mathrm{P}} + \frac{3\rho\,\Delta V}{2\,\Delta t}\right)\phi_{\mathrm{P}}^{n+1} + \sum_{l} A_l \phi_l^{n+1} = Q_{\mathrm{P}}^{n+1} + \frac{2\rho\,\Delta V}{\Delta t}\,\phi_{\mathrm{P}}^{n} - \frac{\rho\,\Delta V}{2\,\Delta t}\,\phi_{\mathrm{P}}^{n-1} \ . \tag{65}$$

If the values of ϕ_{P} at old time levels are also considered as neighbors in the computational molecule, then obviously the relation (52) still holds.

This scheme has a temporal truncation error which is four times as large as in the Crank-Nicolson method. However, the differences between solutions obtained by these two methods in practical applications are very small; the three-time-level scheme is less prone to oscillations and somewhat easier to implement than the Crank-Nicolson scheme, which explains why it is often used.

The attractiveness of this scheme also lies in the fact that it allows re-gridding between time steps (e.g. when flows around moving bodies are computed). Since surface and volume integrals need to be computed only at the new time level, the shape and the number of CVs at previous time levels is irrelevant; one only needs to interpolate the old solutions to the computational nodes of the new grid in order to be able to compute the local time derivative according to Eq. (63). The same is true for the implicit Euler scheme, but since it is only first-order accurate, this feature does not improve its value for the simulation of unsteady flows.

Multipoint methods using more than three time levels are easy to construct and program and require only one evaluation of the derivative per time step, making them relatively cheap. Their principal disadvantage is that they cannot be started solely with data at the initial time point; one has to use another method to get started. In periodic flows, the initial solution does not affect the final periodic behavior so one may simply start with implicit Euler. Another possibility is to use very small time steps initially and a lower-order method, and than gradually increase the time step size and switch to the higher-order method as more data is generated. The multipoint methods can easily be expressed as a blend of a lower-order method plus a correction (this can be done in a nested way); thus, one can easily switch from one method to another.

The difficulties in starting multipoint methods can be avoided by using points between t_n and t_{n+1}, yielding *Runge-Kutta methods*. Second-order Runge-Kutta methods consist of at least two steps; the simplest one uses a half-step Euler predictor followed by a midpoint-rule corrector:

$$\psi^*_{n+\frac{1}{2}} = \psi^n + \frac{\Delta t}{2} F^n \ , \tag{66}$$

$$\psi^{n+1} = \psi^n + \Delta t \, F^*_{n+\frac{1}{2}} \ . \tag{67}$$

This method is easy to use and is self-starting. Runge-Kutta methods of higher order have been developed; the most popular one is of fourth order. The first two steps of this method use an explicit Euler predictor and an implicit Euler corrector at $t_{n+\frac{1}{2}}$. This is followed by a midpoint rule predictor for the full step and a Simpson's rule final corrector that gives the method its fourth order. The method is:

$$\psi^*_{n+\frac{1}{2}} = \psi^n + \frac{\Delta t}{2} F^n \ ; \tag{68}$$

$$\psi^{**}_{n+\frac{1}{2}} = \psi^n + \frac{\Delta t}{2} F^*_{n+\frac{1}{2}} \ ; \tag{69}$$

$$\psi^{***}_{n+1} = \psi^n + \Delta t \, F^{**}_{n+\frac{1}{2}} \ ; \tag{70}$$

$$\psi^{n+1} = \psi^n + \frac{\Delta t}{6} \left[F^n + 2 \, F^*_{n+\frac{1}{2}} + 2 \, F^{**}_{n+\frac{1}{2}} + F^{***}_{n+1} \right] \ . \tag{71}$$

By using methods of different order, it is possible to estimate errors and to construct methods with automatic error control.

The major problem with Runge-Kutta methods is that an nth-order method requires the derivative (i.e., surface and volume integrals representing the convective and diffusive fluxes and source terms) to be evaluated at least n times per time step, making it expensive. For a given order, Runge-Kutta methods are more accurate and more stable than multipoint methods.

Methods for solving RANS-equations – especially the general-purpose ones – usually use implicit time-integration schemes (implicit Euler or second-order methods like Crank-Nicolson or three-level method). Methods for DNS and LES often use higher-order multi-level methods (third or fourth order).

4 SOLUTION OF ALGEBRAIC EQUATION SYSTEMS

Irrespective of which method is used for integration in space and time, one ends up eventually with an algebraic equation system of the form (29) – even in explicit methods, where the calculation of pressure requires that an equation system be solved. Since the conservation equations are in general non-linear, iterative solution methods are necessary. One could still use direct methods to solve the linear equation systems, but since the matrix A and the source vector $\mathbf{Q}$ are not final, an accurate solution of the linearized equations is not necessary. Thus, iterative solvers are preferred for the linear equation systems as well.

An iterative solver starts with an initial solution ϕ^0 and tries to improve it by iterating. When the initial solution is inserted into the equation (29), the equation is not satisfied and we obtain the *residual vector* $\mathbf{R}^0$; at any iteration level m one can write:

$$A\boldsymbol{\phi}^m = \mathbf{Q} - \mathbf{R}^m \ . \tag{72}$$

By subtracting this equation from the one in which ϕ^m is replaced by the exact solution ϕ, one obtains an equation which expresses the link between the residual and *iteration error* $\epsilon^m = \phi - \phi^m$:

$$A\boldsymbol{\epsilon}^m = \mathbf{R}^m \ . \tag{73}$$

The purpose of the iteration procedure is to drive the residual to zero and thus also the iteration error.

If the inversion of the original matrix A were easy, there would be no need to iterate on the linear equation system; however, since the inversion of A is too expensive, one usually constructs an iterative procedure by choosing an *iteration matrix* M (which is easy to invert) and using it to drive the residual to zero:

$$M(\boldsymbol{\phi}^{m+1} - \boldsymbol{\phi}^m) = \mathbf{Q} - A\boldsymbol{\phi}^m \ . \tag{74}$$

If the iterations converge the left-hand side becomes zero and the original equation is satisfied. This equation can be re-written as:

$$M\boldsymbol{\delta}^m = \mathbf{R}^m \ , \tag{75}$$

where $\delta^m = \phi^{m+1} - \phi^m$ is the correction to the solution at the previous iteration level. One usually solves the system (75) instead of the original system (29) and then updates the solution and calculates the new residual vector.

For an iterative method to be effective, solving the system (75) must be cheap and the method must converge rapidly. The first requirement means that M must be easy to invert. The simplest iterative solver, the *Jacobi method*, uses the main diagonal of A as M; the Gauss-Seidel method uses the triangular matrix made of the main diagonal and all the elements below it from A. In both cases the inversion of M is trivial.

For rapid convergence, M should be a good approximation to A, i.e. $(M - A)\boldsymbol{\phi}$ should be small in some sense. For the solution of linear equation systems resulting from discretized conservation equations, one can construct efficient methods based on *incomplete lower-upper decomposition* or ILU-methods. The idea is to find two *sparse* triangular matrices, lower L and upper U, which have the same sparsity as the original matrix A, and use their product LU as the iteration matrix M, i.e. $M = LU$. Since the product of two such triangular matrices in general has more non-zero elements that the matrix A, the decomposition is not exact, hence the ILU-name. There are many variants of these methods. The most popular ones use the idea introduced by Stone (1968), which uses the smoothness property of solutions of transport equations to reduce $(M - A)\boldsymbol{\phi}$. In addition to the original Stone's description of the method for five-diagonal matrices (2D), variants for nine-diagonal matrices (2D, non-orthogonal grid; see e.g. Schneider and Zedan, 1981 and Perić, 1987) and seven-diagonal matrices (3D, compact

computational molecule with nearest neighbors only) have been published. Vectorized version of the latter has been published by Leister and Perić (1994).

The ILU-type methods mentioned above are effective, both on their own and as smoothers in multigrid solvers. However, they are only applicable to structured and block-structured grids. When the grid is unstructured, and especially if the computational molecule is allowed to vary (i.e. the number of neighbors in the algebraic equation at any node is not the same), one has to use more general solvers like those based on *conjugate gradient methods*. Alternatively, one can use a multigrid method with simple smoothers like the Gauss-Seidel method which is applicable to any grid. However, these methods may not be robust as an anisotropic grid may affect the convergence adversely. The methods from the conjugate gradient (CG) family are robust, although more expensive per iteration than the ILU-type solvers. When an accurate solution of a linear equation system is required, CG-type methods belong to the most efficient ones, but since in some FD and FV methods the linear equation systems need not be solved very accurately, ILU-type methods may turn out to be more cost-effective on structured grids.

Space does not allow detailed description of particular solvers here; some are described in the book by Ferziger and Perić (2002), and the interested reader is referred for details to the original references: for CGSTAB-solver, see Van den Vorst and Sonneveld (1990) and Van den Vorst (1992), and for GMRES-solver, see Saad and Schultz (1986).

All of these solvers require more iterations to converge to a prescribed tolerance as the number of unknowns increases. The increase is usually linear for the better solvers, but since each iteration requires more computing time as the grid is refined, the computing time increases over-proportionally (typically quadratically). Multigrid techniques can be used to obtain nearly constant number of iterations irrespective of the grid fineness, so that computing time increases only linearly with grid refinement. Multigrid methods can be viewed as acceleration techniques and can be used in conjunction with any of the aforementioned solvers; experience showed that ILU-type solvers profit especially well from multigrid acceleration.

Multigrid methods can be subdivided into geometric and algebraic ones; the former create coarse grids by joining CVs of the finer grid (FV-methods) or by skipping some points (FD-methods), while the latter do not create specific coarse grids but deduce the corresponding coarse-grid matrix from algebraic manipulations of the fine-grid matrix (based on properties of the matrix). A vast literature exists on multigrid methods, both for linear and non-linear equation systems; here only the book by Hackbush (1985) will be mentioned.

5 COMPUTATION OF INCOMPRESSIBLE FLOWS

We have dealt with the discretization of a generic conservation equation. The same principles apply to the momentum and continuity equations as well. The difference is that the momentum equations have few more terms, and that there is a special relation between the momentum equations and the continuity equation. The integral form of the momentum equation for the *i*th Cartesian component is:

$$\frac{\partial}{\partial t}\int_V \rho u_i \,\mathrm{d}V + \int_S \rho u_i \mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = -\int_S p\,\mathbf{i}_i\cdot\mathbf{n}\,\mathrm{d}S + \int_S \tau_{ij}\mathbf{i}_j\cdot\mathbf{n}\,\mathrm{d}S + \int_V \rho b_i \,\mathrm{d}V\ , \qquad (76)$$

where u_i is the component of the velocity vector $\mathbf{v}$ in the direction of the ith Cartesian coordinate x_i, $\mathbf{i}_i$ is the ith Cartesian unit vector, p is the static pressure, b_i is the ith component of the body force, and τ_{ij} is the viscous part of the stress tensor, defined for incompressible Newtonian fluids as:

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \ , \tag{77}$$

where μ is the dynamic viscosity.

The continuity equation reads:

$$\int_S \rho \mathbf{v} \cdot \mathbf{n} \, \mathrm{d}S = 0 \ . \tag{78}$$

The integrals in these equations can be approximated using the same approximations described for the similar terms of the generic conservation equation. Although there are over 70 forms of the momentum equation, depending on the base vectors and coordinates used (covariant, contravariant, etc.), the form shown above is the most often used one. The reason is that the use of the Cartesian base vectors leads to the simplest, fully conservative form of the component equations. If, on the other hand, one uses the covariant or contravariant components and general, non-orthogonal coordinates, the equations contain additional terms which act as apparent forces and account for the change of velocity components due to changes of coordinates. These terms are non-conservative and involve the so called *curvature terms* or *Christoffel symbols*, which require the computation of second derivatives along grid lines. Since these terms are sensitive to grid smoothness and difficult to calculate numerically, they are best avoided. Fortunately, the simplest form of the equations is also the easiest to handle numerically, so in what follows only the Cartesian components and the equations in the above form will be considered.

In the discussion below we shall use the symbolic notation as the same approach is usually applicable to both FD and FV methods; only where necessary the expressions specific to one method will be used.

5.1 Choice of Variable Arrangement

An important issue when solving the fluid flow problems numerically is the arrangement of variables on the numerical grid. In addition to the natural arrangement in which all the variables share the same location (grid nodes in FD and CV centers in FV methods), there are many other possibilities, some of which have found wide-spread use.

For Cartesian grids, the staggered arrangement introduced by Harlow and Welsh (1967) is both popular and effective. In this arrangement, the x-component velocity, u_x, is stored at points midway between the pressure nodes in x-direction and the y-component velocity, u_y, is stored at points midway between the pressure nodes in y-direction. On this grid each equation has a different set of control volumes. The CVs are selected so that pressure nodes which 'drive' one velocity component are located at the faces of its control volume; the pressure node resides then at the center of the continuity control volume, and the velocities are at its faces. Several terms that require interpolation with the colocated arrangement can be calculated (to second-order accuracy) without interpolation on the staggered grid. Another advantage of this arrangement

is the strong coupling between the velocity and pressure, preventing oscillations which are the major problem of the colocated arrangement. The staggered grid approach does not extend easily to arbitrary grids, which is its main drawback; see below.

When all variables are stored at the same location, such a grid is called *colocated*. One needs to define only one set of grid nodes or CVs. Since many terms in the various equations are nearly identical, the number of coefficients that must be computed and stored is minimized and the programming is simplified, as is the use of multigrid procedures. The colocated arrangement was out of favor for a long time for incompressible flow computation due to the difficulties with pressure-velocity coupling; obvious discretization procedures led to oscillations in pressure. Due to the success of the staggered arrangement, little effort was invested in colocated grids until problems in complex geometries began to be tackled, requiring the use of non-orthogonal grids. An effective way of coupling pressure and velocity on colocated grids was presented by Rhie and Chow (1983). Many methods based on that idea are currently in use. The methods described below will refer to this arrangement, although most of them are equally applicable to staggered grid arrangements as well.

Perić et al. (1988) compared staggered and colocated grid methods for flows in simple geometries using Cartesian grids. They found that both the differences in the solutions on similar grids and in the convergence properties of the solution method were relatively small. Ferziger and Perić (2002) also compared the two arrangements and demonstrated that the differences in solutions were much smaller than the discretization errors (by a factor of 20 in the flows studied). Also, the numbers of iterations required to converge the iterations to the same tolerance was almost identical for several grids of different fineness. Although some authors reported different observations in such comparisons (with appreciable differences), there is no reason to believe that the results should be much different when the same kind of approximations is used on the same grid.

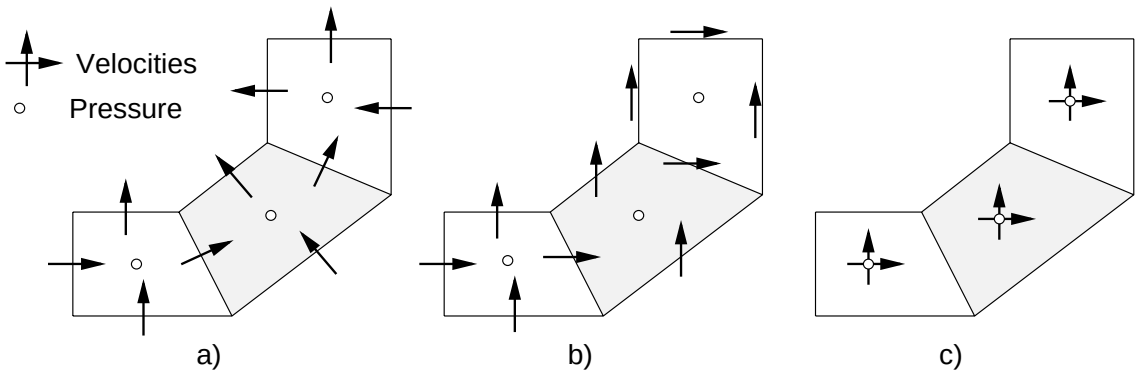

Fig. 7: Arrangements of variables in a FV method: (a) staggered with contravariant velocity components, (b) staggered with Cartesian velocity components, and (c) colocated with Cartesian velocity components.

When the grid is non-orthogonal, only the grid-oriented (covariant or contravariant) velocity components can be used in a staggered arrangement without the occurrence of problems with pressure-velocity-coupling. However, since the conservation equations then lose their strong-conservation form and since the discretization and solution of such equations is difficult, this arrangement is not recommended. It was used by Demirdžić et al. (1987), among others. If

Cartesian velocity components are staggered on a non-orthogonal grid, they may become parallel to the cell faces and thus may not contribute to the mass flux there. Also, there is no longer a pair of pressure nodes 'driving' the velocity component, so oscillations may occur. The measures to avoid these problems are no simpler than those used on colocated grids, so there is no advantage in staggering the storage locations. Several other staggered arrangements were proposed by various authors (e.g. storing velocities at corners of pressure cells, or storing all Cartesian velocity components at all faces of a pressure-CV), but these have not found wide-spread use.

The simplest arrangement is the colocated one, see Fig. 7. The mass fluxes through the CV faces, which are needed to approximate the convective fluxes in all equations, can (to a second order approximation) be calculated by interpolating the values from two cell centers on either side of the face, irrespective of whether the grid is orthogonal or not. Since an effective coupling of velocities and pressure is possible and has been used to construct solution methods up to fourth order (see Lilek and Perić, 1995), only this arrangement will be considered below.

The most important issue in computing incompressible flows is the calculation of pressure. The construction of an iterative solution method which provides velocity and pressure fields which satisfy both the momentum equations and the continuity equation is the subject of the following sections.

5.2 The Pressure Equation

One can derive an equation for pressure by taking the divergence of the vector form of the momentum equation, leading to:

$$\boldsymbol{\nabla} \cdot (\boldsymbol{\nabla}\, p) = -\boldsymbol{\nabla} \cdot \left[\boldsymbol{\nabla} \cdot (\rho \mathbf{v}\mathbf{v} - \tau_{ij}\mathbf{i}_i\mathbf{i}_j) - \rho\mathbf{b} + \frac{\partial(\rho\mathbf{v})}{\partial t} \right] \ . \tag{79}$$

The differential form of this equation in Cartesian coordinates reads:

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{\partial}{\partial x_j}\left(\rho u_i u_j - \tau_{ij}\right)\right] + \frac{\partial(\rho b_i)}{\partial x_i} + \frac{\partial^2 \rho}{\partial t^2} \ . \tag{80}$$

For the case of constant density and viscosity, this equation becomes much simpler; the viscous and unsteady terms disappear by virtue of the continuity equation leaving:

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{\partial(\rho u_i u_j)}{\partial x_j}\right] \ . \tag{81}$$

One could now set up the following iterative solution procedure:

1. Discretize and solve momentum equations in turn, using pressure values from the previous iteration;

2. Discretize and solve the pressure equation, using velocities obtained in the previous step;

3. Repeat these steps until changes in velocity and pressure fields become negligible.

The pressure equation can be discretized and solved using any of the approaches described for the generic conservation equation. It is important to note that the right-hand side of the pressure equation represents the sum of derivatives of the terms from the momentum equations; these must be approximated in a manner consistent with their treatment in the equations they are derived from. The second derivative of pressure on the left-hand side has been intentionally written in the form shown in Eq. (81): the derivative inside parenthesis stems from the gradient operator on pressure in the momentum equation and it has to be discretized in the same way here; the outer derivative stems from the divergence operator and should be approximated in the way the continuity equation is discretized. These two approximations need not be the same, but it is essential that the consistency issues mentioned above are observed. Note also that the outer derivatives on both sides of the above equation must be approximated in the same way, since they stem from the same divergence operator applied to the momentum equation and representing the continuity equation constraint. If the left-hand side is discretized using an approximation for the second derivative (the Laplacean) which does not guarantee the above consistency requirement, problems with mass conservation are encountered. The continuity equation is not explicitly solved – it is used to derive the pressure equation and the only way to enforce the mass conservation is through obeying the above rules regarding the discretization of the pressure equation.

The pressure equation of the above form is seldom used in numerical methods for solving the Navier-Stokes equations; it is used to calculate pressure when the velocity field is known, like when the streamfunction-vorticity method is used to calculate it. However, these methods will not be described here as they are difficult to implement in three dimensions. The most often used methods of computing the pressure in incompressible flows is based on deriving a discrete equation for pressure or pressure-correction from the discretized form of the continuity equation or by using the so called *projection methods*; these are described below.

5.2 SIMPLE and Related Methods

Consider the momentum equations written in symbolic difference form (the choice of the approximations to the spatial derivatives is not important for the description of the steps leading to an equation for pressure):

$$\frac{\partial(\rho u_i)}{\partial t} = -\frac{\delta(\rho u_i u_j)}{\delta x_j} - \frac{\delta p}{\delta x_i} + \frac{\delta \tau_{ij}}{\delta x_j} + b_i = H_i - \frac{\delta p}{\delta x_i} \ , \tag{82}$$

where $\delta/\delta x$ represents a discretized spatial derivative (which could represent a different approximation in each term) and H_i is shorthand notation for the convective, viscous, and source terms.

For simplicity, assume that the Eq. (82) is solved using the explicit Euler method for time advancement. We then have:

$$(\rho u_i)^{n+1} - (\rho u_i)^n = \Delta t \left(H_i^n - \frac{\delta p^n}{\delta x_i} \right) \ . \tag{83}$$

The velocity field at the new time step, which can be obtained from this equation once the

pressure is known, does not necessarily satisfy the continuity equation:

$$\frac{\delta(\rho u_i)^{n+1}}{\delta x_i} = 0 \ . \tag{84}$$

The mass conservation has to be enforced by taking the numerical divergence (using the numerical operator used to approximate the continuity equation) of Eq. (83) leading to:

$$\frac{\delta(\rho u_i)^{n+1}}{\delta x_i} - \frac{\delta(\rho u_i)^{n}}{\delta x_i} = \Delta t \left[\frac{\delta}{\delta x_i} \left(H_i^n - \frac{\delta p^n}{\delta x_i} \right) \right] \ . \tag{85}$$

The first term on the left-hand side is the divergence of the new velocity field, which is required to be zero. The second term is the divergence of the velocity field at time step n, which was forced to be zero in the previous time step. Setting the right-hand side equal to zero leads to the discrete Poisson equation for the pressure p^n:

$$\frac{\delta}{\delta x_i} \left(\frac{\delta p^n}{\delta x_i} \right) = \frac{\delta H_i^n}{\delta x_i} \ . \tag{86}$$

Note that the operator $\delta / \delta x_i$ outside the parentheses is the divergence operator inherited from the continuity equation, while $\delta p \, / \delta x_i$ is the pressure gradient from the momentum equations. If the pressure p^n satisfies this discrete Poisson equation, the velocity field at time step $n+1$ calculated from Eq. (83) will be divergence-free (in terms of the discrete divergence operator). The time step to which this pressure belongs is arbitrary; one can use p^{n+1} in place of p^n without changing anything in the algorithm.

This procedure for time-advancing the Navier-Stokes equations is:

- Start with a velocity field u_i^n at time t_n which satisfies both the momentum and continuity equations.
- Compute the convective, viscous, and source terms (expressed jointly above as H_i^n) and take the divergence of the result.
- Solve the Poisson equation for the pressure p^n.
- Compute the velocity field at the new time step from Eq. (83). It satisfies both the momentum and continuity equations.
- Advance to the next time step.

Methods similar to this are used when explicit time integration schemes are employed, the major difference being that the schemes of higher order are used rather than the explicit Euler.

When steady flows are computed, implicit time integration methods with large time steps (or equivalents of them) are used. The discretized equations to be solved at the new time step are the non-linear algebraic equation systems which may be written as:

$$A_{\mathrm{P}}^{u_i} u_{i,\mathrm{P}}^{n+1} + \sum_l A_l^{u_i} u_{i,l}^{n+1} = Q_{u_i}^{n+1} - \left(\frac{\delta p^{n+1}}{\delta x_i} \right)_{\mathrm{P}} \ , \tag{87}$$

where 'P' is the index of a velocity node, and the sum is over the neighbor points (how many neighbor nodes are involved depends on the approximations of surface integrals in the momentum equations; usually only the nearest neighbors are treated implicitly). The source term Q contains terms dependent on u_i^n, body forces and, perhaps, terms which depend on u_i^{n+1} and are treated by a kind of deferred correction. The pressure term is written symbolically to emphasize the independence of the solution method from the spatial discretization approximation. The discretization of the spatial derivatives may be of any type or order.

Due to non-linearity and coupling, Eqs. (87) cannot be solved by a direct method; the coefficients A, and possibly the source term, depend on the unknown u_i^{n+1}. An iterative method is the only choice. For unsteady flows, iteration must be continued until the equations are accurately satisfied. For steady flows, the tolerance can be more generous; the only requirement is that the steady equations are satisfied at the end, so one often performs just one iteration per time step. An alternative is to use an infinite time step and discard the unsteady term altogether; one then has to use under-relaxation methods to ensure convergence, which can be interpreted as a kind of time stepping with time steps which vary from point to point; see Patankar (1980) or Ferziger and Perić (2002) for details.

The solution process is similar to the one described above. At each time step, one first improves the velocities by solving the linearized momentum equations. Then, an equation for pressure or pressure correction is solved in order to correct the pressure and velocities to enforce mass conservation. These steps are repeated until all equations are satisfied to within an acceptable tolerance; one can then proceed to the next time step. The above steps which are repeated within each time step are called *outer iterations*, as distinguished from the *inner iterations* or *solver iterations* used to solve the systems of linearized equations. The index for the time step $n+1$ shall be dropped hereafter and an outer iteration index m will be introduced; u_i^m thus represents the current estimate (after the mth outer iteration) of the new time step solution u_i^{n+1}. At the beginning of each outer iteration, the terms on the right-hand side of Eqs. (87) are evaluated using the variables from the preceding outer iteration. This means that the pressure is treated explicitly. Thus one solves Eqs. (87) with p^m replaced by p^{m-1}; the result is called u_i^{m*} to indicate that it is a provisional velocity field that needs to be corrected to enforce the mass conservation.

A common method of this kind uses a pressure-correction (rather than the pressure itself) to correct the velocity. The corrections to the velocity and pressure fields can be expressed as:

$$u_i^m = u_i^{m*} + u' \quad \text{and} \quad p^m = p^{m-1} + p' \; . \tag{88}$$

The velocity u^{m*} satisfies the linearized momentum equation in which the pressure from the previous outer iteration is used:

$$A_{\mathrm{P}}^{u_i} u_{i,\mathrm{P}}^{m*} + \sum_l A_l^{u_i} u_{i,l}^{m*} = Q_{u_i}^{m-1} - \left(\frac{\delta p^{m-1}}{\delta x_i} \right)_{\mathrm{P}} \; . \tag{89}$$

If we subtract this equation from the corresponding equation in terms of corrected velocities and pressure, u_i^m and p^m, use Eq. (88), and divide the resulting equation by A_{P}, we obtain an equation relating the velocity and pressure corrections:

$$u'_{i,\mathrm{P}} = \tilde{u}'_{i,\mathrm{P}} - \frac{1}{A_{\mathrm{P}}^{u_i}} \left(\frac{\delta p'}{\delta x_i} \right)_{\mathrm{P}} \; , \tag{90}$$

where the following shorthand notation was used:

$$\tilde{u}'_{i,\mathrm{P}} = -\frac{\sum_l A_l^{u_i} u'_{i,l}}{A_{\mathrm{P}}^{u_i}} \ . \tag{91}$$

By requiring that the corrected velocities u_i^m satisfy the discretized continuity equation, one obtains:

$$\frac{\delta(\rho u_i^{m*})}{\delta x_i} + \frac{\delta(\rho u'_i)}{\delta x_i} = 0 \ . \tag{92}$$

Finally, using Eq. (90) to express the three velocity corrections u'_i through pressure correction yields the pressure-correction equation:

$$\frac{\delta}{\delta x_i}\left[\frac{\rho}{A_{\mathrm{P}}^{u_i}}\left(\frac{\delta p'}{\delta x_i}\right)\right]_{\mathrm{P}} = \left[\frac{\delta u_i^{m*}}{\delta x_i}\right]_{\mathrm{P}} + \left[\frac{\delta \tilde{u}'_i}{\delta x_i}\right]_{\mathrm{P}} \ . \tag{93}$$

The major approximation is now made: the velocity corrections $\tilde{u}'_i$ are unknown so it is common practice to neglect them. This crude approximation requires special attention for the algorithm to work efficiently, as will be shown below. Once the simplified pressure-correction equation has been solved, the velocity corrections are calculated from Eq. (90), in which $\tilde{u}'_i$ is omitted, and added to the provisional velocities according to Eq. (88). If the pressure-correction equation is solved exactly, the corrected velocity satisfies the continuity equation. This is the SIMPLE algorithm, which was developed in early 70ies and is usually attributed to Patankar and Spalding (1972).

Due to the approximation introduced above, SIMPLE requires small time steps or under-relaxation in the momentum equations and converges slowly. The under-relaxation (which is usually applied in all non-linear equations within outer iterations) is incorporated by modifying the central coefficient A_{P} and source term Q (which in this case contains also the pressure term) as follows:

$$\underbrace{\frac{A_{\mathrm{P}}^{u_i}}{\alpha_u}}_{\tilde{A}_{\mathrm{P}}^{u_i}} u_{i,\mathrm{P}}^{m*} + \sum_l A_l u_{i,l}^{m*} = \underbrace{Q_{u_i}^{m-1} + \frac{1-\alpha_u}{\alpha_u} A_{\mathrm{P}}^{u_i} u_{i,\mathrm{P}}^{m-1}}_{\tilde{Q}_{u_i}^{m-1}} \ , \tag{94}$$

where $\tilde{A}_{\mathrm{P}}^{u_i}$ and $\tilde{Q}_{u_i}^{m-1}$ are the modified main diagonal matrix elements and source vector components, respectively. In the following, tilde will be omitted but it is assumed that under-relaxation has been applied and that A_{P} and Q have been modified according to the above expression. This modified equation is solved within inner iterations. When the outer iterations converge, the terms involving α_u cancel out and we obtain the solution of the original equation.

This kind of under-relaxation was proposed by Patankar (1980). It has a positive effect on many iterative solution methods since the diagonal dominance of the matrix A is increased (the element $\tilde{A}_{\mathrm{P}}^{u_i}$ is larger than $A_{\mathrm{P}}^{u_i}$, while other elements A_l remain the same). It is also much more efficient than an explicit under-relaxation in which the newly calculated value from the unmodified equation is simply blended with the value from the previous iteration.

The rate of convergence of the SIMPLE-algorithm is improved if only a portion of the pressure correction p' is added to p^{m-1}, i.e. if instead of Eq. (88) we use:

$$p^m = p^{m-1} + \alpha_p p' \ , \tag{95}$$

where the under-relaxation factor is $0 \leq \alpha_p \leq 1$. Patankar (1980) suggested to use $\alpha_u \approx 0.5$ and $\alpha_p \approx 0.8$; however, subsequent analysis showed that lower values of α_p allow higher values of α_u and faster convergence. The nearly-optimum relation between α_p and α_u has been independently derived via different routes by Raithby and Schneider (1979) and Perić (1985):

$$\alpha_p = 1 - \alpha_u \ . \tag{96}$$

One can usually work with $\alpha_u \approx 0.8$ and $\alpha_p \approx 0.2$ with nearly-optimum results.

Instead of neglecting the last term in the pressure correction equation (93), one can approximate it. For example, the velocity correction $u'_{i,\mathrm{P}}$ at any node could be approximated by a weighted mean of the neighbor values, which leads to:

$$u'_{i,\mathrm{P}} \approx \frac{\sum_l A_l^{u_i} u'_{i,l}}{\sum_l A_l^{u_i}} \quad \Longrightarrow \quad \tilde{u}'_{i,\mathrm{P}} \approx -u'_{i,\mathrm{P}} \frac{\sum_l A_l^{u_i}}{A_\mathrm{P}^{u_i}} \ , \tag{97}$$

where Eq. (91) has been used. When this is inserted in Eq. (90), the following approximate relation between u'_i and p' is obtained:

$$u'_{i,\mathrm{P}} = -\frac{1}{A_\mathrm{P}^{u_i} + \sum_l A_l^{u_i}} \left(\frac{\delta p'}{\delta x_i} \right)_\mathrm{P} \ , \tag{98}$$

which is used to derive the pressure-correction equation from (92). Since the neighbor coefficients are negative and $A_\mathrm{P}^{u_i}$ is larger than the sum of the absolute values of the neighbor coefficients due to under-relaxation or unsteady term, the denominator in the above expression on the right-hand side is positive but smaller than in the simplified expression used in SIMPLE. The algorithm based on Eq. (98) is called SIMPLEC (van Doormal and Raithby, 1984); it converges more rapidly than the SIMPLE method. SIMPLEC does not need an under-relaxation of the pressure correction, but if in SIMPLE one uses the relation (96), it becomes equivalent to SIMPLEC.

Issa (1986) proposed the following predictor-corrector procedure. In the first step – which is equivalent to the SIMPLE method – the term involving $\tilde{u}'_i$ is neglected, i.e. the velocity correction u'_i is expressed as (see Eq. (90)):

$$u'_{i,\mathrm{P}} = -\frac{1}{A_\mathrm{P}^{u_i}} \left(\frac{\delta p'}{\delta x_i} \right)_\mathrm{P} \ . \tag{99}$$

After solving the pressure-correction equation:

$$\frac{\delta}{\delta x_i} \left[\frac{\rho}{A_\mathrm{P}^{u_i}} \left(\frac{\delta p'}{\delta x_i} \right) \right]_\mathrm{P} = \left[\frac{\delta u_i^{m*}}{\delta x_i} \right]_\mathrm{P} \ , \tag{100}$$

the velocities and pressure are corrected as described above. In a second step, the velocity correction u'_i is corrected by introducing a second correction defined by:

$$u'_{i,\mathrm{P}} + u''_{i,\mathrm{P}} = \tilde{u}'_{i,\mathrm{P}} - \frac{1}{A_\mathrm{P}^{u_i}} \left(\frac{\delta p'}{\delta x_i} + \frac{\delta p''}{\delta x_i} \right)_\mathrm{P} \ , \tag{101}$$

where $\tilde{u}'_i$ is calculated from Eq. (91) after u'_i has been calculated from Eq. (99). Using Eq. (99) and considering that the velocities $u_i^{m*} + u'_i$ already satisfy the continuity equation, the

requirement that it remains satisfied after the second correction is applied leads to the following equation for the second pressure correction:

$$\frac{\delta}{\delta x_i}\left[\frac{\rho}{A_{\mathrm{P}}^{u_i}}\left(\frac{\delta p''}{\delta x_i}\right)\right]_{\mathrm{P}} = \left[\frac{\delta \tilde{u}_i'}{\delta x_i}\right]_{\mathrm{P}} . \tag{102}$$

Further corrector steps can be added, but this is rarely done. This method is known as PISO algorithm (Issa, 1986). No under-relaxation is needed for the pressure corrections – they are simply added together. The equation for the second pressure correction has the same coefficient matrix as the equation for p' but different right-hand side, which can be exploited in some linear equation solvers (one can store the inverted iteration matrix and use it with the new source term to obtain p'').

The second corrector is needed because the first pressure-correction equation ensures only that the continuity equation is satisfied but disregards the momentum equations by simplifying the relationship between u' and p'. If the momentum equations were linear (in the limit of zero Reynolds number), one would not need to solve them at all – just to calculate the coefficient matrix. One could then start with a guessed velocity and pressure fields and by repeating the corrector steps in PISO achieve velocities and pressure which satisfy both the momentum and continuity equations. It can be also shown that the error after the second correction is proportional to $(\Delta t)^2$ when unsteady problems are solved, which means that one need not do more than one outer iteration per time step. However, since the order of an approximation does not give information about the magnitude of the error, only about the rate at which it is reduced with further refinements, the non-iterative computation of unsteady flows is feasible only when Δt is sufficiently small.

A disadvantage of the PISO method is that the coefficient matrix of the momentum equations is needed after the pressure-correction equation is solved. One usually overwrites the matrix of the previous equation, but in this case one has to store the matrix of the momentum equations for use in corrector steps of PISO algorithm. Since in 3D the matrix involves usually seven elements per grid node, the memory overhead is large (alternatively, one can write and read the matrix from hard disc, which causes longer computing times). Also, further simplifications of the pressure-correction equation (e.g. when the grid is non-orthogonal, see below) are not allowed if the advantages of PISO are to be retained.

Another method from the same family is SIMPLER (Patankar, 1980). It begins in the manner of the SIMPLE method; the pressure-correction equation (100) is solved first. The pressure correction so obtained is used only to correct the velocity field, but not to update the pressure field. A new pressure field (not a correction) is calculated from a *pressure equation*, which is obtained by requiring that the corrected velocity and the new pressure filed satisfy also the linearized momentum equation:

$$A_{\mathrm{P}}^{u_i} u_{i,\mathrm{P}}^m + \sum_l A_l^{u_i} u_{i,l}^m = Q_{u_i}^{m-1} - \left(\frac{\delta p^m}{\delta x_i}\right)_{\mathrm{P}} . \tag{103}$$

Since the corrected velocities are now known, one can directly express the nodal velocity through the pressure gradient:

$$u_{i,\mathrm{P}}^m = \frac{Q_{u_i}^{m-1} - \sum_l A_l^{u_i} u_{i,l}^m}{A_{\mathrm{P}}^{u_i}} - \frac{1}{A_{\mathrm{P}}^{u_i}}\left(\frac{\delta p^m}{\delta x_i}\right)_{\mathrm{P}} = \hat{u}_{i,\mathrm{P}}^m - \frac{1}{A_{\mathrm{P}}^{u_i}}\left(\frac{\delta p^m}{\delta x_i}\right)_{\mathrm{P}} . \tag{104}$$

From the requirement that the right-hand side of the above equation also satisfies the continuity equation (the left-hand side was forced to do so in the first step), the pressure equation of the SIMPLER method is obtained:

$$\frac{\delta}{\delta x_i}\left[\frac{\rho}{A_{\mathrm{P}}^{u_i}}\left(\frac{\delta p^m}{\delta x_i}\right)\right]_{\mathrm{P}} = \left[\frac{\delta \hat{u}_i^m}{\delta x_i}\right]_{\mathrm{P}} . \tag{105}$$

Note that the coefficient matrix of the pressure equation is the same as that of the pressure-correction equation – only the source term is different. While the pressure correction is reset to zero after each outer iteration (it remains zero when the outer iterations converge), the pressure equation uses pressure from the previous outer iteration, p^{m-1}, as the starting field.

The pressure-correction algorithms of the above kind can be summarized as follows:

- Advance time to the new level, $t_{n+1} = t_n + \Delta t$;
- Start outer iteration on the new time level:
  1. Assemble and solve the linearized momentum equations, using velocity, pressure, and fluid properties based on solutions from previous time step or outer iteration; call the result u_i^{m*}.
  2. Assemble and solve the pressure-correction equation for p'.
  3. Correct the velocity and pressure to obtain u_i^m, which satisfies the continuity equation, and p^m.

     In PISO, solve the second pressure-correction equation and correct both velocity and pressure again.

     In SIMPLER, solve the pressure equation for p^m.
  4. Return to step 1 and repeat, until all corrections are negligible.
- Set $u_i^{n+1} = u_i^m$ and $p^{n+1} = p^m$ and advance to the next time step.

The algorithms described above show the principles of pressure-velocity coupling using the pressure or pressure-correction equation. Depending on the discretization method and the grid arrangement, a detailed scheme can easily be derived.

A word about the boundary conditions for the pressure-correction equation is appropriate here. When the velocity component normal to solution domain boundary (i.e. the mass fluxes through the boundary cell faces) is known, it does not need to be corrected, i.e. $u' = 0$. This is equivalent to setting the gradient of the pressure correction in the same direction to zero, see Eq. (99). Thus, the pressure-correction equation has Neumann boundary conditions on all boundaries where the velocity is prescribed (inlet, wall, symmetry etc.). Even at boundaries where the velocity is not prescribed one usually first applies an extrapolation (or another kind of approximation) of the velocity and applies a global correction to make velocities at boundaries satisfy the global mass conservation. When this is done, the velocity is – for the sake of deriving the pressure-correction equation – considered as specified and the velocity correction is set to zero.

If an equation has zero-gradient boundary condition on all boundaries, the sum of all source terms in the interior has to vanish. This is indeed the case in the pressure-correction equation, as will be shown below when a pressure-correction equation in a FV method is derived. The solution is not unique, but that is not a problem since in incompressible flows only pressure gradient counts; one usually keeps the pressure fixed at one node (by subtracting the pressure correction at that node from all other corrections) and adjusts the pressure at other nodes relative to it.

5.3 Fractional-Step Methods

The fractional step concept is more a generic approach than a particular method; there are many variants of it. Only one particular method will be shown here, which has been proposed by Choi and Moin (1993) and which was found suitable for DNS and LES in complex geometries. It is a four-step method based on the Crank-Nicolson time integration method; the authors used staggered Cartesian grid and central differences of second order for all spatial derivatives. However, the fractional step approach can be applied even if different parts of the equation are advanced in time using different methods (often explicit for convective and implicit for viscous terms, sometimes implicit only in the direction normal to wall and explicit in other directions); the spatial discretization can be of any kind and order. Note, however, that there are no outer iterations in fractional step methods of the same kind as in pressure-correction methods; after the last step, the solution at the new time level is obtained and one proceeds to the next time level.

In the first step, the velocity is advanced using pressure from the previous time step; convective terms, viscous terms, and body forces (if present) are represented by an equal blend of old and new values (Crank-Nicolson method in this particular case):

$$\frac{(\rho u_i)^* - (\rho u_i)^n}{\Delta t} = \frac{1}{2}\left[H(u_i^n) + H(u_i^*)\right] - \frac{\delta p^n}{\delta x_i} \ , \tag{106}$$

where $H(u_i)$ is an operator representing the discretized convective, diffusive, and source terms. This system of equations must be solved for u_i^*; any method can be used. Unless the time step is very small, one should iterate to account for the non-linearity of the equations; Choi and Moin (1993) used a Newton iterative method.

In the second step, half of the old pressure gradient is removed from u_i^*:

$$\frac{(\rho u_i)^{**} - (\rho u_i)^*}{\Delta t} = \frac{1}{2}\frac{\delta p^n}{\delta x_i} \ . \tag{107}$$

The final velocity at the new time level is given the required contribution from the (as yet unknown) new pressure:

$$\frac{(\rho u_i)^{n+1} - (\rho u_i)^{**}}{\Delta t} = -\frac{1}{2}\frac{\delta p^{n+1}}{\delta x_i} \ . \tag{108}$$

The requirement that the new velocity satisfies the continuity equation leads to a Poisson equation for pressure:

$$\frac{\delta}{\delta x_i}\left(\frac{\delta p^{n+1}}{\delta x_i}\right) = \frac{2}{\Delta t}\frac{\delta (\rho u_i)^{**}}{\delta x_i} \ . \tag{109}$$

Upon solution of the pressure equation, the new velocity field is obtained from Eq. (108). It satisfies the continuity equation and also the momentum equation in a slightly modified form:

$$\frac{(\rho u_i)^{n+1} - (\rho u_i)^n}{\Delta t} = \frac{1}{2}\left[H(u_i^n) + H(u_i^*)\right] - \frac{1}{2}\left(\frac{\delta p^n}{\delta x_i} + \frac{\delta p^{n+1}}{\delta x_i}\right) \ . \tag{110}$$

To be correct, $H(u_i^*)$ should be replaced by $H(u_i^{n+1})$ in the above equation. However, from Eqs. (107) and (108) one can easily show that the error is of second order in time and thus consistent with other errors:

$$u_i^{n+1} - u_i^* = -\frac{\Delta t}{2\rho}\frac{\delta}{\delta x_i}\left(p^{n+1} - p^n\right) \approx \frac{(\Delta t)^2}{2\rho}\frac{\delta}{\delta x_i}\left(\frac{\delta p}{\delta t}\right) \ . \tag{111}$$

Note that, by subtracting Eq. (106) from Eq. (110), one obtains an expression for the pressure correction $p' = p^{n+1} - p^n$:

$$\frac{(\rho u_i)^{n+1} - (\rho u_i)^*}{\Delta t} = -\frac{1}{2}\frac{\delta p'}{\delta x_i} \ . \tag{112}$$

By requiring that u_i^{n+1} satisfies the continuity equation, a Poisson equation for p' is obtained; it has the same form as Eq. (109), only p^{n+1} needs to be replaced by p' and u_i^{**} by u_i^*. Thus, SIMPLE and fractional-step methods are very similar; SIMPLE includes outer iterations to eliminate the error due to not updating $H(u_i^*)$ in Eq. (110), i.e. when outer iterations converge, Eq. (110) will be solved with $H(u_i^{n+1})$ in place of $H(u_i^*)$. In the SIMPLE method, the pressure-correction equations need not be solved accurately at each outer iteration, since the mass conservation needs to be enforced only in the converged solution; in the fractional-step methods, the pressure-correction equation is solved only once per time step and thus has to be iterated to a tight tolerance to assure mass conservation; multigrid methods are often used for this purpose.

5.3 Other Methods

For incompressible two-dimensional flows with constant fluid properties, the Navier-Stokes equations can be simplified by introducing the *streamfunction* ψ and *vorticity* $\boldsymbol{\omega}$ as dependent variables. The number of equations reduces than from three in primitive variables to two, since the pressure can be eliminated by virtue of continuity equation. However, for three-dimensional problems and flows with variable fluid properties, this approach becomes more complicated than when primitive variables are used and is therefore not recommended.

Another approach which is sometimes used is that of *artificial compressibility*. There are several variants of these methods, the oldest being proposed by Chorin (1967). The idea is to introduce a time derivative of pressure in the continuity equation and make it similar to the corresponding equation for compressible flows, in which a time derivative of density is present. The mass conservation is thus violated until the procedure converges, where the time derivative of pressure becomes zero. In the case of steady flows, this poses no problem; however, when unsteady flows are considered, one has to introduce pseudo time for marching within each step of real time. The method thus becomes very similar to pressure-correction methods of SIMPLE-type, the pseudo time corresponding to outer iterations (which can indeed be interpreted as

marching in time, the time step being proportional to the under-relaxation factor and varying from CV to CV). Actually, one can show that the pressure-correction methods and artificial-compressibility methods are closely related and basically use the same principle of linking the velocity and pressure gradient via momentum and continuity equations (see Ferziger and Perić, 2002). Examples of artificial compressibility methods can be found in Kwak et al. (1986) and Belov et al. (1995), among others.

6 COMPUTATION OF COMPLEX FLOWS

Solution methods for complex geometries use non-orthogonal, boundary-fitted grids. Structured grids are limited to relatively simple topologies. Branching geometry and multiply-connected domains require the use of either block-structured or unstructured grids. Block-structured grids can be further subdivided in overlapping and non-overlapping; furthermore, the non-overlapping blocks of grid may be matching or non-matching at block interfaces. Some of the issues related to block-structured FV grids will be discussed below.

In the case of a block-structured grid made of hexahedra, one uses within one block the regular data structure, i.e. the location of a CV is identified by indices (i, j, k). The neighbors are identified by increasing or lowering the indices; the east neighbor has the indices $(i+1, j, k)$ etc. However, along block interfaces a different data structure is necessary if an arbitrary block connectivity and possibly non-matching grid is allowed. One can create a list of all cell faces lying in the block interface, which must contain all the information necessary to approximate the surface integrals. These are the indices of CV-centers on either side (which reside in different blocks), interpolation factors, components of surface vector etc. The surface integrals over interface cell faces are calculated only once and applied to CVs on either side (with opposite signs), thus ensuring conservation of fluxes.

In an explicit method the computation can be performed block-wise without any special measures. In implicit methods one needs to modify the iteration matrix. The usual practice, which is also suitable for parallel computing using domain-decomposition technique, is to split the coefficient matrix A in a local part, A_{l}, and a coupling part, A_{c}. The local part represents the matrix as it would have been if one were considering each block on its own, regarding the variables in neighbor blocks as known. The coupling part contains those elements of A which reside in one block but multiply variable values from other blocks. The iteration matrix is then decoupled; for example, in an ILU method, one would perform the decomposition of each local block matrix, while the coupling part would not be considered. Thus, the iteration matrix M would be an approximation of A_{l} alone. This basically means that the variable values in neighbor blocks are within one inner iteration considered as known, i.e. values from previous inner iteration are used. The implication is that one usually needs more inner iterations to achieve the same level of convergence; however, the increase is moderate and can become significant only if the number of blocks and block interfaces is very large.

The approach described above can be used even if the grids in two neighbor blocks do not match. An example is presented in Fig. 8, which shows the interface between two blocks of different fineness. Some authors use the so called “hanging nodes” on either side of the

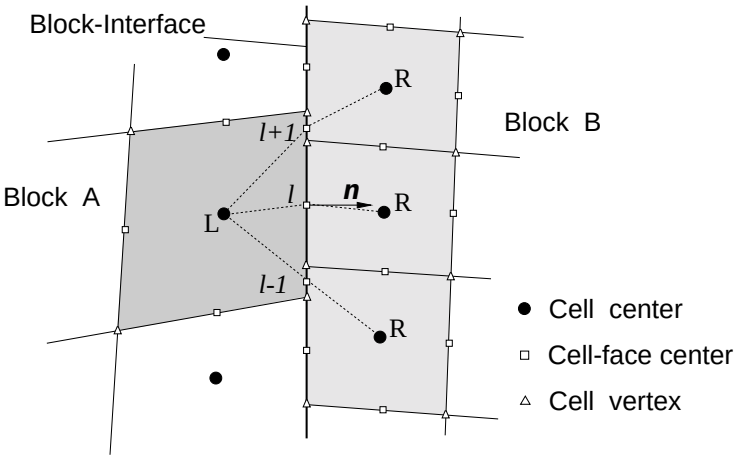

Fig. 8: An example of a block-structured grid with a non-matching interface.

interface as boundary nodes of each block; here another possibility will be described. Rather than having hanging nodes one can treat CVs along interfaces as polyedra (polygons in 2D), with more faces (and neighbors) than the CVs within the block.

Since the shaded CV in block A of Fig. 8 has three neighbors on its east face, one cannot use the usual notation for structured grids here. This face is not of the regular type (with one neighbor on the opposite side), so it is not considered when working in block A. The coefficient matrix and the source term for this CV will thus be incomplete, since the contribution from its east side is missing; in particular, the coefficient A_{E} will be zero.

In order to treat the irregular cell faces found at block interfaces, one has to use another kind of data structure here – one similar to that used when the whole grid is unstructured. Each piece of the interface common to two CVs must be identified (by a pre-processing tool) and placed on a list together with all of the information needed to approximate the surface integrals: the indices of the left (L) and right (R) neighbor cells, the surface vector (pointing from L to R) and the coordinates of cell-face center. With this information, one can use the method used in the interior of each block to approximate the fluxes through these faces.

Each interface cell face contributes to the source terms for the neighboring CVs (explicit contributions to the convective and diffusive fluxes treated by deferred correction), to the main diagonal coefficient (A_{P}) of these CVs, and to two off-diagonal coefficients: A_L for node R and A_R for node L. The problem of irregularity of data structure due to having three east neighbors is thus overcome by regarding the contributions to the global coefficient matrix as belonging to the interface cell faces (which always have two neighbor cells) rather than to the CVs. It is then irrelevant how the blocks are ordered relative to each other (the east side of one block can be connected to any side of the other block): one has only to provide the indices of the neighbor CVs to the interface cell faces.

The contributions from interface cell faces, namely A_L and A_R, become parts of the coupling matrix A_{c}. The iteration procedure is now:

1. Assemble the elements of matrix A and the source term Q in each block, ignoring the contributions of the block interfaces.

2. Loop over the list of interface cell faces, updating A_{P} and Q_{P} at nodes L and R, and calculate the matrix elements stored at the cell face, A_L and A_R.

3. Calculate elements of matrices L and U in each block disregarding neighbor blocks, i.e. as if they were on their own (when ILU-type solvers or pre-conditioners are used).

4. Calculate the residuals in each block using the local part of the matrix A; the residuals for the CVs along block interfaces are incomplete, since the coefficients that refer to neighbor blocks are zero.

5. Loop over the list of interface cell faces and update the residuals at nodes L and R by adding the products $A_R\phi_R$ and $A_L\phi_L$, respectively; once all faces have been visited, all the residuals are complete.

6. Compute the variable update at each node in each block and return to step 1.

7. Repeat until the convergence criterion is met.

One may treat blocks with non-matching grids of any kind in the same way; this is in particular the case when a block-structured grid is block-wise refined. Especially when flows around bodies are calculated, it is advantageous to generate first a matching – but coarse – block-structured grid and then successively refine blocks near the body.

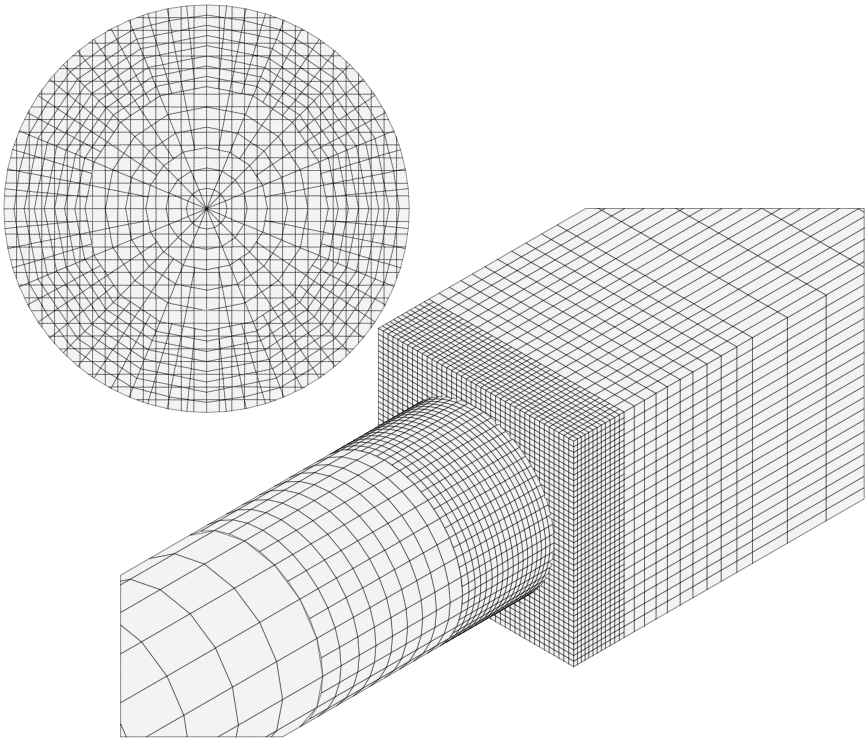

Fig. 9: Geometry of the square-to-round duct and the grid used, showing also the detail of grid on both sides of the interface

The most flexible approach to computing flows in complex geometries is to use an unstructured grid which may consist of CVs with an arbitrary number of faces. Although one usually

generates the grids made of tetrahedra, pyramids, prisms, or hexahedra, by performing a local, cell-wise grid refinement one can create cells with many faces. Also, by constructing CVs around vertices of tetrahedral elements, one usually obtains polyhedral cells with many faces. Another situation in which the allowance for arbitrary polyhedral cells can be useful is at non-matching block-interfaces or interfaces along which two grid blocks slide along each other. An example of a grid which includes both local refinements and a non-matching interface is shown in Fig. 9. By allowing polyhedral cells instead of hanging nodes, one achieves a more implicit coupling of all CVs in the iteration matrix and faster convergence of iterations. The general approach to handling such grids is to introduce an object-oriented data structure and define objects *vertex, edge, face,* and *volume*. Edges are defined by the two vertices on either end, faces by a list of edges (which must form a closed polygon), and volumes by a list of faces which enclose it. The discretization requires approximations to surface and volume integrals; these are computed in loops runing over the list of faces and volumes, respectively.

A method that allows the use of arbitrary polyhedral CVs has been presented by Demirdžić and Muzaferija (1995) and Demirdžić et al. (1997). The discretization method is based on midpoint rule integration, linear interpolation, and central differencing, as described here for the generic conservation equation. The only specialties are the data structure and the linear equation solver (a version of conjugate gradient solver) which allows arbitrary number of neighbors.

The derivation of an appropriate pressure-correction equation for a FV method using arbitrary CVs will now be briefly described. The surface and volume integrals in the momentum equation are assumed to be approximated using second-order approximations (midpoint rule integration, linear interpolation, and central differencing). The pressure term is also treated conservatively, i.e. in each equation the appropriate components of the pressure forces on the CV surface are calculated. However, one can express the pressure contribution to the source term also as a product of the mean pressure gradient and cell volume using the Gauss theorem:

$$Q^p_{i,\mathrm{P}} = -\int_S p\,\mathbf{i}_i \cdot \mathbf{n}\,\mathrm{d}S = -\int_V \frac{\partial p}{\partial x_i}\,\mathrm{d}V$$

$$\sum_k p_k S^i_k \approx \left(\frac{\partial p}{\partial x_i}\right)_{\mathrm{P}} \Delta V \quad \Rightarrow \quad \left(\frac{\partial p}{\partial x_i}\right)_{\mathrm{P}} = \frac{\sum_k p_k S^i_k}{\Delta V}\,. \tag{113}$$

As was described earlier, the linearized momentum equations are solved first using pressure from previous iteration. In order to enforce the continuity constraint, one needs to calculate mass fluxes through CV faces using new velocity components u_i^{m*}. To this end one needs velocity vector at the cell face, which requires interpolation from neighbor CV centers. Simple linear interpolation may lead to oscillations; the reason is that an oscillatory pressure distribution is not sensed by the pressure source term in Eq. (113) when the oscillation wavelength is two CV widths. Thus, interpolated velocity at the cell face – which contains interpolated pressure terms calculated at cell centers – will also not sense pressure gradient due to oscillatory pressure field. If the pressure term was calculated at the cell face (as is the case when a staggered arrangement is used), the pressure oscillation would result in a large term and lead to a pressure correction which will smooth the oscillation out. Rhie and Chow (1983) presented an interpolation practice which avoids oscillations: the velocity is interpolated, but a correction

to the interpolated value is applied. The correction is negligible when the pressure variation is smooth, but becomes large if oscillations are present. The correction term is nothing else but the difference between the interpolated pressure term and the one calculated at the cell face (see Eq. (89)):

$$v_{n,\mathrm{e}}^{m*} = \overline{(v_n^{m*})}_{\mathrm{e}} - \Delta V_{\mathrm{e}} \overline{\left(\frac{1}{A_{\mathrm{P}}^{v_n}}\right)}_{\mathrm{e}} \left[\left(\frac{\delta p^{m-1}}{\delta n}\right)_{\mathrm{e}} - \overline{\overline{\left(\frac{\delta p^{m-1}}{\delta n}\right)}}_{\mathrm{e}} \right] , \tag{114}$$

where $v_{n,\mathrm{e}} = (\mathbf{v}\cdot\mathbf{n})_{\mathrm{e}}$ is the velocity component normal to the cell face (here, the 'east' face of one CV is considered; the same expressions are valid for any face when the indices are substituted accordingly). Since the coefficient A_{P} is the same for all Cartesian velocity components, the same value is also used here as $A_{\mathrm{P}}^{v_n}$. The correction term is proportional to the third derivative of pressure multiplied by $(\Delta n)^2$ and vanishes when the variation of pressure is linear or quadratic; Δn is here the mesh spacing in the direction normal to the cell face. It reduces with grid refinement as a second-order error term and is therefore consistent with other approximations (see Ferziger and Perić, 2002, for a detailed discussion of this issue). The pressure derivative in the normal direction approximated at the cell face can be obtained from Eq. (50). Since that approximation is second-order accurate at a location midway between nodes P and E, see Fig. 5, the interpolated derivative should also be calculated at that location; thus, the double overbar in the above equation denotes arithmetic average.

The interpolated derivative at a cell face can be obtained by interpolating the components of the gradient vector from the two cell centers; these are calculated at each CV center during solution of the momentum equations according to Eq. (113). Thus, one can write:

$$\overline{\overline{\left(\frac{\delta p^{m-1}}{\delta n}\right)}}_{\mathrm{e}} = \overline{\overline{(\boldsymbol{\nabla} p)}}_{\mathrm{e}} \cdot \mathbf{n} \, . \tag{115}$$

The mass flux through the face 'e' can now be calculated as:

$$\dot{m}_{\mathrm{e}}^{m*} = \int_S \rho \mathbf{v} \cdot \mathbf{n} \, \mathrm{d}S \approx (\rho v_n^{m*} S)_{\mathrm{e}} \, . \tag{116}$$

The mass fluxes through other faces are calculated accordingly. The continuity equation is in general not satisfied by these mass fluxes, but results in a mass imbalance:

$$\sum_k \dot{m}_k^{m*} = \Delta \dot{m}_{\mathrm{P}} \, , \tag{117}$$

where the summation is over all cell faces (note that the same value of the mass flux is used for two CVs, but with opposite signs, i.e. $\dot{m}_{\mathrm{e}}$ for a CV centered around node P is equal to $-\dot{m}_{\mathrm{w}}$ for a CV centered around node E).

In order to enforce the mass conservation, the mass fluxes need to be corrected by correcting the normal velocity component. By analogy with Eq. (99), the velocity correction can be expressed through the gradient of the pressure correction as follows (see also Eq. (114)):

$$v_n' \approx -(\rho \, \Delta V \, S)_{\mathrm{e}} \overline{\left(\frac{1}{A_{\mathrm{P}}^{v_n}}\right)}_{\mathrm{e}} \left(\frac{\delta p'}{\delta n}\right)_{\mathrm{e}} \, . \tag{118}$$

The derivative of p' with respect to n at the cell face center can again be obtained from expression (50); however, the inclusion of the whole expression on the right-hand side of this equation would lead to a too complicated pressure-correction equation. Since the role of the pressure-correction equation is only to drive the algorithm towards convergence, where all corrections must become negligible, and since the pressure-correction equation of the SIMPLE algorithm is not exact anyway, one can here introduce another approximation: neglect the contribution of non-orthogonality and express the derivative of p' with respect to n by only the first term on the right-hand side of Eq. (50):

$$\left(\frac{\delta p'}{\delta n}\right)_{\mathrm{e}} \approx \frac{p'_{\mathrm{E}} - p'_{\mathrm{P}}}{(\mathbf{r}_{\mathrm{E}} - \mathbf{r}_{\mathrm{P}}) \cdot \mathbf{n}} \ . \tag{119}$$

When this simplified expression is introduced in Eq. (118) and the mass flux corrections are expressed as in Eq. (116),

$$\dot{m}'_{\mathrm{e}} \approx (\rho v'_n S)_{\mathrm{e}} \ , \tag{120}$$

the requirement that the corrected fluxes satisfy the continuity equation

$$\sum \dot{m}_k^{m*} + \sum \dot{m}'_k = 0 \tag{121}$$

leads to the algebraic pressure-correction equation at each CV of the form:

$$A_{\mathrm{P}}^{p'} p'_{\mathrm{P}} + \sum_l A_l^{p'} p'_l = -\Delta \dot{m}_{\mathrm{P}} \ . \tag{122}$$

Once the pressure-correction equation is solved, one can correct the cell-face mass fluxes, cell-center velocities, and pressure. Only an α_p–portion of p' is added to p^{m-1}, as described earlier for the SIMPLE method.

When the grid is not appreciably non-orthogonal, the neglected effects of non-orthogonality are hardly felt. However, when the grid is substantially non-orthogonal, one needs to further reduce α_p, typically to 0.1 or even below. One can take the effect of grid non-orthogonality into account through another corrector, similar to the PISO corrector described earlier; see Ferziger and Perić (2002) for a description of one such approach.

If mass fluxes are specified at boundaries, the pressure-correction equation has zero normal derivative there (Neumann boundary conditions, as described earlier). However, one may also specify pressure at inlet and outlet and let the mass flow rate be determined in the course of computation. In that case, $p' = 0$ at these boundaries (Dirichlet boundary condition), but the velocity is unknown. The velocity is first extrapolated to boundary using an expression like (114) and is corrected accordingly upon solving for p'. For details on pressure boundary conditions, see Ferziger and Perić (2002) and Gresho and Sani (1990).

7 COMPUTATION OF COMPRESSIBLE FLOWS

A large number of methods specially designed to compute compressible flows have been developed in the past. In many cases the viscous terms are neglected and one solves the special

version of the momentum equations – the *Euler equations*. Without viscous terms there are no boundary layers so one does not need as fine grids near walls as in the case of viscous flows. Especially in aerodynamics, methods using unstructured triangular (in 2D) or tetrahedral (in 3D) grids have been developed and applied to computing flows around whole aircrafts or space shuttles.

In the case of viscous compressible flows, one needs fine resolution near walls and the grids made of triangles or tetrahedra become then so stretched that both the accuracy and the computational efficiency becomes to suffer substantially. The cure was found in using quadrilateral or hexahedral cells near walls and triangles or tetrahedra further away. The grid generation becomes than more complicated, and if the solution method is specially designed for either cell type, such a combination becomes also complicated. However, one can design a method for arbitrary cells, e.g. as described by Demirdžić et al. (1997).

The methods specially designed for compressible flows solve the continuity equation for density and obtain the pressure from an equation of state. Both the spatial and temporal integration can be of any type described earlier for the generic conservation equation. There are, though, some special versions of discretization methods and special terminologies which are encountered in literature devoted to compressible flows. In most cases one can interpret these methods as a combination of the methods described earlier; in particular, very often the integration is first done using a lower-order method and then *re-constructed* by adding a correction which leads to a higher-order approximation, which is more or less the same as the deferred correction approach described earlier. Details on methods design specifically for compressible flows can be found in Hirsch (1992), among other books.

There are also special coupled solution methods, which consider the density, velocity components, and temperature as one vector of unknowns; iterative solution methods with multigrid acceleration are usually used to solve the resulting algebraic equation systems. With special preconditioning (see Weiss and Smith, 1995) such coupled solvers can be used for both compressible and incompressible flows; see Weiss et al (1999) for a description of one such method. These methods are somewhat more complicated to program and require more memory than the sequential solution methods of the SIMPLE-type, but due to the simultaneous solution for all variables they are potentially more robust.

Most of the methods design specifically for compressible flows usually become inefficient if the Mach number is low in some regions, since in that case the density does not change, neither in space nor in time. On the other hand, the pressure-correction methods, originally developed for truly incompressible flows, can easily be extended to compressible flows, leading to efficient methods for all flow speeds. This approach is described below since it simple and yet effective.

To compute compressible flows, it is necessary to solve not only the continuity and momentum equations but also a conservation equation for the thermal energy and an equation of state. The energy equation – which is usually written as an equation for enthalpy – must retain terms which are usually neglected in incompressible flows. For example, viscous dissipation may be a significant heat source and conversion of internal energy to kinetic energy (and vice versa) by means of flow dilatation is also important. In integral form the energy equation reads:

$$\frac{\partial}{\partial t}\int_V \rho h \,\mathrm{d}V + \int_S \rho h \mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = \int_S \lambda \boldsymbol{\nabla} T\cdot\mathbf{n}\,\mathrm{d}S + \int_V (\mathbf{v}\cdot\boldsymbol{\nabla} p + \boldsymbol{S}:\boldsymbol{\nabla}\mathbf{v})\,\mathrm{d}V + \frac{\partial}{\partial t}\int_V p\,\mathrm{d}V \ . \tag{123}$$

Here h is the enthalpy per unit mass, T is the absolute temperature, λ is the thermal conductivity and $\boldsymbol{S}$ is the viscous part of the stress tensor, $\boldsymbol{S} = \tau_{ij}\mathbf{i}_i\mathbf{i}_j$. The stress tensor components τ_{ij} now contain additional terms which account for compressibility effects:

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\boldsymbol{\nabla}\cdot\mathbf{v} \ , \tag{124}$$

where δ_{ij} is the Kronecker's delta ($\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise).

For a perfect gas with constant specific heats, c_p and c_v, the enthalpy becomes $h = c_\mathrm{p}T$, allowing the energy equation to be written in terms of the temperature. Furthermore, under these assumptions, the equation of state is:

$$\rho = \frac{p}{RT} \ , \tag{125}$$

where R is the gas constant. In pressure-correction methods for all flow speeds, the equation of state is used to compute density once the pressure and temperature have been updated.

The major extension to the pressure-correction scheme described earlier for incompressible flows is the addition of the unsteady term in the continuity equation, and the correction of density in addition to velocity correction. The discretized mass conservation equation thus reads in the first step (implicit Euler scheme used):

$$\frac{\Delta V}{\Delta t}(\rho^{m-1} - \rho^n) + \sum_k \dot{m}_k^{m*} = \Delta\dot{m}_\mathrm{P} \ . \tag{126}$$

In the correction step one has to take into account that by changing the pressure, both velocity and density will change. The velocity correction at cell-face center is expressed as a function of the normal gradient of pressure correction, see Eq. (118); the density correction at cell-face center can be expressed as a function of the pressure correction itself:

$$\rho'_\mathrm{e} \approx \left(\frac{\partial\rho}{\partial p}\right)_T p' = \frac{p'}{RT} \ . \tag{127}$$

The corrected mass flux can now be expressed as:

$$(\dot{m}^{m*} + \dot{m}')_\mathrm{e} = (\rho^{m-1} + \rho')_\mathrm{e}(v_n^{m*} + v'_n)_\mathrm{e} S_\mathrm{e} \ . \tag{128}$$

If one neglects the product of density and velocity corrections (since it tends to zero faster than other terms), the mass-flux correction can be expressed as:

$$\dot{m}'_\mathrm{e} \approx (\rho^{m-1} v'_n S)_\mathrm{e} + (\rho' v_n^{m*} S)_\mathrm{e} \ . \tag{129}$$

If one substitutes the velocity correction as given by Eq. (118) and the density correction as given by Eq. (127), after adopting an interpolation scheme for the cell-face center value of p'

and upon introducing mass-flux corrections into the mass conservation equation for corrected quantities,

$$\frac{\delta V}{\Delta t}\rho' + \sum_k \dot{m}'_k + \Delta \dot{m}_{\mathrm{P}} = 0 \ , \tag{130}$$

a pressure-correction equation is obtained. The expressions for the coefficients depend on the approximations of cell-face center values, which can be of any type described earlier for convective and diffusive fluxes. If shocks are present and one uses linear interpolation and central differences, oscillations around shocks will result. Some kind of damping is necessary to avoid them; one may blend first-order upwind schemes with central differences, or use some kind of non-oscillatory schemes or limiters (e.g. total-variation diminishing or TVD schemes, see Hirsch, 1992, or essentially non-oscillatory or ENO schemes, see Sonar, 1997).

There are important differences between the pressure-correction equation for incompressible and for compressible flows. The former represents the discretized Poisson equation, i.e. a transport equation with diffusive terms only. In the compressible case, the pressure-correction equation contains in addition convective and unsteady terms, i.e. it resembles the generic conservation equation. Also, the solution of the pressure-correction equation for incompressible flow – if the mass fluxes are prescribed at all boundaries – is indeterminate to within an additive constant; in the compressible case, the presence of convective terms makes the solution unique (the pressure must be prescribed somewhere on the boundary).

The nice feature of the pressure-correction equation is that it automatically adapts to the local state of the flow. The convective term is of the order of Ma^2 relative to the diffusive term; in regions of flow where the Mach number is low, the equation reduces to the form appropriate for incompressible flows. On the other hand, when and where the Mach number is large, the convective terms dominate and properly reflect the hyperbolic nature of the flow. Solving the pressure-correction equation is then equivalent to solving the continuity equation for density, as is traditionally done in methods designed for compressible flows.

This type of the method is especially attractive for flows in which both strongly compressible and almost incompressible regions exist. Examples are flows in valves and body wakes. Details of this kind of solution methods can be found in Demirdžić et al. (1993), Karki and Patankar (1989) and Van Doormal et al. (1987), among others. In these references, as well as in Ferziger and Perić (2002), details on various boundary conditions in compressible flows are also given.

8 COMPUTATION OF FLOWS WITH MOVING BOUNDARIES

In many engineering applications, the flow domain is changing in time due to the movement of boundaries. The movement may be determined by external effects (like in piston-driven flows) or it may depend on the flow itself and have to be determined as part of the solution (like in free-surface flows). In both cases the grid has to move with the boundary. If the base vectors remain fixed (e.g. if we use the Cartesian components as before), the only change in the conservation equations is the appearance of the relative velocity in convective terms; however, to

be conservative, one has to consider the so called *Space Conservation Law* (SCL). The conservation equations for space, mass, and momentum in integral form for a control volume which may move read:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathrm{d}V - \int_S \mathbf{v}_\mathrm{b}\cdot\mathbf{n}\,\mathrm{d}S = 0\ , \tag{131}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \rho\,\mathrm{d}V + \int_S \rho(\mathbf{v}-\mathbf{v}_\mathrm{b})\cdot\mathbf{n}\,\mathrm{d}S = 0\ , \tag{132}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \rho u_i\,\mathrm{d}V + \int_S u_i\rho(\mathbf{v}-\mathbf{v}_\mathrm{b})\cdot\mathbf{n}\,\mathrm{d}S = \int_S(\tau_{ij}\mathbf{i}_j - p\,\mathbf{i}_i)\cdot\mathbf{n}\,\mathrm{d}S + \int_V \rho b_i\,\mathrm{d}V\ . \tag{133}$$

Here is $\mathbf{v}_\mathrm{b}$ the velocity with which the CV boundary is moving. Obviously, if the control volume boundary moves with the same velocity as the fluid, the convective fluxes are equal to zero. The control volume becomes then the *control mass* and we are dealing with the Lagrangean approach to the description of fluid motion.

The Eq. (131) describes the conservation of space when the CV changes its shape and position with time. If the grid is not moving, this equation becomes redundant and in all other equations $\mathbf{v}_\mathrm{b} = 0$ applies. Thus, the equations for non-moving CVs are only a special case of the equations for moving CVs. The solution method designed for moving grids can then be used for fixed grids as well.

It should be noted that the time derivative in the above equations has a different meaning in fixed and moving grids, although it is approximated in the same way. If the CV does not move, the time derivative represents the local change of the conserved quantity, while in another extreme of a CV whose surface moves exactly with fluid velocity, it becomes the total (material) derivative since no fluid enters or leaves the CV. This change of meaning is accounted for by the convective fluxes, which become zero in the latter case.

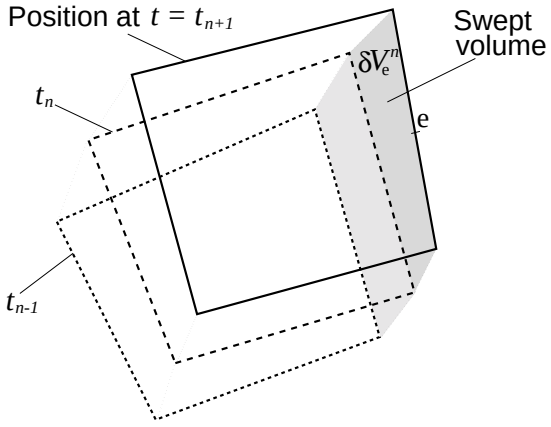

Fig. 10: A typical 2D CV at three time levels and the volumes swept by cell faces

Why is it important to obey the SCL can be seen by considering the mass-conservation equation (132) for a fluid of constant density; it can then be written as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathrm{d}V - \int_S \mathbf{v}_\mathrm{b}\cdot\mathbf{n}\,\mathrm{d}S + \int_S \mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = 0\ . \tag{134}$$

The first two terms represent the SCL and add up to zero, cf. Eq. (131); thus, for fluids with constant density, the mass conservation equation reduces to

$$\int_S \mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S = 0 \quad \text{or} \quad \boldsymbol{\nabla}\cdot\mathbf{v} = 0\ . \tag{135}$$

It is therefore important to ensure that the above two terms cancel out in the discretized equations as well (i.e. the sum of volume fluxes through CV faces due to their movement must equal the rate of change of volume); otherwise, artificial mass sources are introduced and they may accumulate in time and spoil the solution, as demonstrated by Demirdžić and Perić (1988).

The implementation of SCL in a FV method using fully implicit time integration scheme with three time levels will now be briefly outlined. The extension to other time integration schemes is straightforward.

The discretized SCL equation can be cast into the following form (see Eq. (63)):

$$\frac{3\,(\Delta V)^{n+1} - 4\,(\Delta V)^{n} + (\Delta V)^{n-1}}{2\,\Delta t} = \sum_k (\mathbf{v}_\mathrm{b}\cdot\mathbf{n})_k S_k\ , \tag{136}$$

where the summation is over all faces of the CV. Note that the difference between CV volumes at consecutive time levels can be expressed as the sum of volumes δV_k swept by each CV face when moving from its old to the new position, see Fig. 10, i.e.:

$$(\Delta V)^{n+1} - (\Delta V)^{n} = \sum_k \delta V_k^n\ . \tag{137}$$

When this expression is introduced in Eq. (136), one finds out that the SCL is satisfied identically if the volume fluxes through cell faces are defined as:

$$\dot{V}_\mathrm{e}^{n+1} = \left(\int_{S_\mathrm{e}} \mathbf{v}_\mathrm{b}\cdot\mathbf{n}\,\mathrm{d}S\right)^{n+1} \approx [(\mathbf{v}_\mathrm{b}\cdot\mathbf{n})_\mathrm{e} S_\mathrm{e}]^{n+1} \approx \frac{3\,\delta V_\mathrm{e}^{n} - \delta V_\mathrm{e}^{n-1}}{2\,\Delta t}\ . \tag{138}$$

Therefore, the volumes swept by each face over one Δt, δV_k, are calculated from the grid position at two time levels and used to calculate volume fluxes $\dot{V}_k^{n+1}$; there is then no need to define explicitly the velocity of the CV-surface, $\mathbf{v}_\mathrm{b}$. The alternative is to calculate the grid velocity $\mathbf{v}_\mathrm{b}$ from the movement of the cell-face center and calculate the volume flux by multiplying it with the new surface vector; however, it is difficult then to make sure that SCL is satisfied.

The mass flux through the face 'e' can now be calculated as (see Eq. (116)):

$$\dot{m}_\mathrm{e}^{m*} = \int_{S_\mathrm{e}} \rho\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S - \int_{S_\mathrm{e}} \rho\mathbf{v}_\mathrm{b}\cdot\mathbf{n}\,\mathrm{d}S \approx (\rho v_n^{m*} S)_\mathrm{e} - \rho_\mathrm{e}\dot{V}_\mathrm{e}^{m-1}\ . \tag{139}$$

The requirement that the discretized mass conservation equation be satisfied:

$$\frac{3\,(\rho\,\Delta V)^{m-1} - 4\,(\rho\,\Delta V)^{n} + (\rho\,\Delta V)^{n-1}}{2\,\Delta t} + \sum_k \dot{m}_k^{m*} + \sum_k \dot{m}_k' = 0\ , \tag{140}$$

and the introduction of the above defined expressions for $\dot{V}_k$ and $\dot{m}_k'$ lead finally to a pressure-correction equation of the same form as in the case of fixed grids (the same approach is valid for both compressible and incompressible flows). If the grid position at the new time level is known (e.g. in piston-driven flows, flows around rotating machinery etc.), the volume flux through cell faces $\dot{V}_k^{n+1}$ is not dependent on the outer iteration counter m; otherwise, the volume fluxes need

to be corrected during outer iterations together with other corrections. This is the case when free-surface flows, flows in blood vessels etc. are computed.

When the grid position is known at each time level, the implementation of the grid movement in the solution procedure is simple; see Demirdžić and Perić (1990) and Ferziger and Perić (2002) for more details and examples of application. When the boundary movement is not known in advance, an iterative procedure within each time step (outer iterations) has to be used, as outlined above.

In some cases bodies move relative to each other and it is not possible to use a single grid which adapts to new positions of each body (two cars or trains passing each other, train entering a tunnel, a propeller rotating between ship body and rudder etc.). These situations require special treatment. Sometimes it is feasible to subdivide solution domains in blocks and let the grid in some blocks move while keeping it fixed in others. The numerical method has to be capable of treating block interfaces with non-matching grids, since one block slides along interface with the other block. This is not difficult to implement, as outlined earlier when block-structured grids were discussed. At each time step the connectivity at the interface changes, so one needs an algorithm for finding which CVs are in contact. This approach is especially suited for studying flows around rotating objects (propellers etc.). The alternative is to use overlapping (Chimera) grids, in which case one block of grid is attached to a body and moves with it in a fixed background grid. The coupling of solutions in two grids is more difficult than when the blocks are sliding along an interface; interpolation must be used and the conservativeness is not so easy to ensure. However, there are several methods that have been proposed and used in the past (Tu and Fuchs, 1992; Perng and Street, 1991; Zang and Street, 1995; Hubbard and Chen, 1994, 1995).

9 GRID QUALITY AND OPTIMIZATION

The discretization errors are reduced when the grid is refined; however, an optimization of the grid quality can often reduce the errors by as much (or even more) as its refinement.

The optimization of the grid is aimed at improving the accuracy of approximations to surface and volume integrals. This, of course, depends on the discretization method used. Here, grid features which affect the accuracy of the method described above will be briefly described; most of them are of general relevance.

In order to obtain maximum accuracy of convective fluxes when using linear interpolation, the line connecting two neighbor CV centers should pass through the center of the common face. In certain cases, especially when the grid generator uses some kind of block-wise generation, situations like in Fig. 11 are unavoidable. However, one can then locally refine the grid as indicated in Fig. 11, in order to reduce the distance between cell-face center k and the location at which the straight line from node C to N_k passes through the cell face, k'. The distance $|\mathbf{r}_k - \mathbf{r}_{k'}|$ relative to some linear measure of the cell face (e.g. $\sqrt{S_k}$) can thus be used as one measure of the grid quality.

The maximum accuracy of diffusive fluxes is achieved when the line connecting the neighbor CV centers (i) is orthogonal to the cell face, and (ii) passes through the cell-face center. The

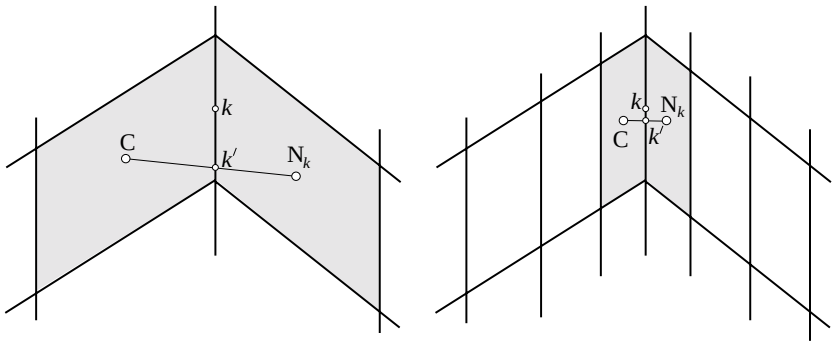

Fig. 11: An example of poor grid quality due to a large distance from k to k' (left) and the improvement through a selective local grid refinement (right)

latter property is also important for convective fluxes and has been discussed above; the former is another measure of the grid quality. For discretization methods of the kind presented here, it is not important that the grid lines are orthogonal to one another at CV corners; only the angle between the line connecting neighbor CV centers and the cell-face normal matters (see angle θ in Fig. 12). A tetrahedral grid can be orthogonal in this sense. Large angle θ can lead to both convergence problems and unphysical solutions and should be avoided.

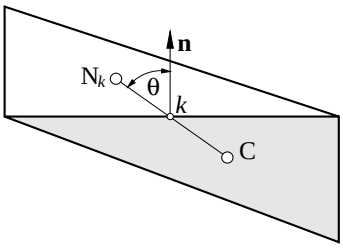

Fig. 12: An example of the measure of grid non-orthogonality

Different other kinds of undesirable distortions of CVs may be encountered. Two are depicted in Fig. 13; in one case, the upper cell face of a regular hexahedral CV is simply rotated around its normal, creating high warpage of adjacent faces, while in the other case, the top face is simply displaced by shear in its own plane. Both features are undesirable and should be avoided if at all possible, as they lead to poor approximations of both diffusive and convective fluxes.

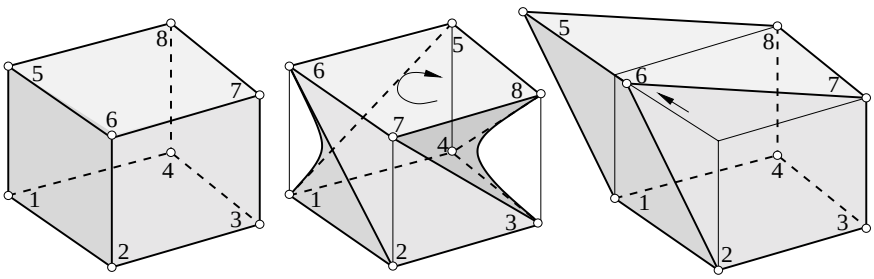

Fig. 13: An example of poor grid quality due to a large warpage (middle) and distortion (right)

Grids made of triangles in 2D and tetrahedra in 3D can also cause problems. One of the

problematic situations is shown in Fig. 14: the CV centered around node C is very narrow. While the velocity component in x-direction at node C is strongly coupled to pressures at nodes N_2 and N_3, there is only one "supporting" node for the velocity component in y-direction, the node N_1. In other words, the y-component of the velocity vector computed at node C can begin to oscillate and attain large values. Another kind of thin triangles or tetrahedra usually results near walls and causes problems due to strong non-orthogonality when diffusive fluxes are computed. The problem is even more pronounced in 3D, where all neighbor nodes may fall nearly in one plane, making computation of the gradient at v center difficult. The pressure-velocity coupling algorithm will not be able to remove these oscillations and the outer iterations may not converge. It is therefore important that such thin triangles or tetrahedra are avoided in computational grids.

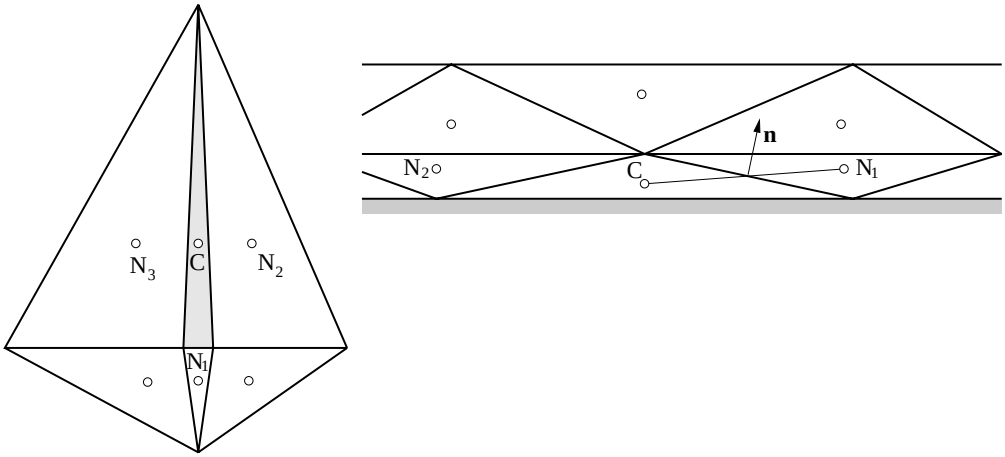

Fig. 14: An example of poor quality of triangular or tetrahdral grids

If the computational nodes are placed at CV centroids, the volume integrals approximated by the midpoint rule are second-order accurate. However, CVs may sometimes be so deformed, that the centroid is actually located outside the CV. This and other above mentioned grid features should be avoided, for the sake of computational efficiency and accuracy. Some of the problems can be avoided by subdividing problematic cells (and possibly some surroundings cells) into smaller ones in a way which reduces the problem, as indicated for one of the grid features above.

When computing flows in engineering practice, it is often difficult to generate *any* grid, let alone a *good* one. The task of grid generation is also the most time-consuming one; it may take weeks, while the computation (in the case of steady-flow computation) may be finished overnight on a standard workstation. The need for tools which could generate a good grid automatically is therefore obvious; the lack of such tools is the greatest obstacle for a more wide-spread use of CFD in engineering practice.

10 CLOSING REMARKS

Computation of laminar flows poses nowadays no problems: the available computing resources are big enough to enable accurate solutions even in complex 3D problems. The numerical

methods have become very powerful too: one can use unstructured grids with arbitrary control volumes, and the grids may move in time and adapt to complicated boundary conditions. On the other hand, the development of multigrid techniques and parallel computing has lead to a dramatic improvement of computational efficiency. These methods could not be described here, but interested readers can find a lot of information in literature devoted to these subjects (a more detailed discussion is also available in Ferziger and Perić, 2002).

The accuracy of numerical solutions is also easy to assess when laminar flows are considered, since only the discretization and iteration errors are present. The iteration errors are easily controlled by monitoring the residual levels and using sufficient precision of computer arithmetics. The discretization errors can also be reliably assessed by using Richardson extrapolation and a systematic grid refinement, as they depend on the approximations used and the grid fineness. For a method of order p and sufficiently fine grids (so that monotonic convergence is achieved), the grid-independent solution can be estimated as follows:

$$\phi \approx \phi_h + \frac{\phi_h - \phi_{2h}}{r^p - 1} \ , \tag{141}$$

where r is the refinement factor and h denotes a measure of grid fineness. For example, when the size of CVs is halved in all directions ($r = 2$) and a second-order method as the one described above is used ($p = 2$), the errors on the finest grid (spacing h) are approximately equal to one third of the difference between solutions on that and the next coarser grid (spacing $2h$). For more details and examples of application of these methods, see Ferziger and Perić (2002). The estimation of discretization errors can also be used as a guidance for a local grid refinement; indeed, fully-automatic adaptive methods do exist and will be certainly more common in the future.

Computation of turbulent flows is much more difficult. One can not use DNS for most engineering flows due to too high Reynolds numbers, and even LES is not always possible. On the other hand, the engineer needs reliable answers fast if CFD tools are to be useful in practice. The need to replace the exact equations which describe fluid flow by modeled equations which are easy enough to solve (RANS equations) introduces an additional source of error – the *modeling error*. Even if very fine grids are used and the numerical errors (iteration and discretization errors) are negligible, the solution contains modeling errors which may be substantial (usually of the order of few per cent, but the errors may even be as large as 20%; the solution may even be qualitatively wrong).

The numerical and modeling errors interact in an unpredictable way. At many workshops, solutions produced by different researchers for the same problem have been compared in the past (see Rodi et al., 1995, and Bradshaw et al., 1994). The conclusion was most often that it was not possible to identify the best model because differences in solutions obtained by different authors using the same model were often larger than the differences between solutions obtain by the same authors using different models. There are many details, especially associated with the treatment of wall boundaries, which can be responsible for these differences, in addition to programming and other 'bugs' which exist in all computer codes. In some cases one obtains very good agreement between numerical and experimental results when coarse grids are used; when the grid is refined, the agreement becomes less favorable. One should therefore be very

careful when assessing the quality of turbulence models and compare the results with experimental data only after the iteration and discretization errors have been reduced substantially below the expected level of modeling errors.

Out of a large variety of numerical methods for computing turbulent flows, the user should choose one which is most appropriate for the kind of flows that will be studied. There is no point in using too sophisticated, high-order numerical methods if the models used are known to produce modeling errors of the order of 10%; also, there is no point in trying a DNS with methods of first order. Methods of second order in space and time are hardly any more complicated than first order methods, while the difference in accuracy is enormous (see Ferziger and Perić, 2002, for examples of comparisons). If the method is to be a general purpose one, block-structured or unstructured FV grids and schemes of second order like Crank-Nicolson, central differencing, linear interpolation, midpoint rule integration etc. are probably the best choice. For special applications like in the analysis of some features of turbulence, methods of high order and regular grids are appropriate.

Any turbulence model can be implemented in any of the numerical solution method described above. The model equations always have the form of the generic conservation equation, the main new feature being the source term which usually contains many terms involving first (and sometimes also second) derivatives of velocity components with respect to spatial coordinates. Since these terms are usually evaluated at the CV center and the volume integral is approximated by the midpoint rule, the evaluation of these terms is easy as the gradients at cell center are computed anyway when approximating the diffusive fluxes; the values of velocity components from previous iteration are used. Some models also require distance from the nearest wall, which can also be easily computed a priori for an arbitrary unstructured grid during pre-processing. If the divergence of velocity vector is needed (like in compressible flows), it can also be easily computed for arbitrary CVs using Gauss theorem, as described above for the approximation of diffusive fluxes.

The implementation of boundary conditions, especially at walls, is of crucial importance. Since the conditions depend on the model used, they will not be further discussed here – only the importance of a careful treatment of boundaries is emphasized.

ACKNOWLEDGEMENTS

The author agknowledges contributions to the material presented in this note by J. H. Ferziger, I. Demirdžić, S. Muzaferija, and many other present and past co-workers. Financial support by the Deutsche Forschungsgemeinschaft through various research projects in the past is also greatly appreciated.

REFERENCES

1. Anderson, D.A, Tannehill J.C., Pletcher, R.H.: *Computational fluid mechanics and heat transfer*, Hemisphere, New York, 1984.

2. Baker, A.J.: *Finite element computational fluid mechanics*, McGraw-Hill, New York, 1983.

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

4. Baliga, R.B.: Control-volume finite element methods for fluid flow and heat transfer, in W.J. Minkowycz, E.M. Sparrow (eds.), *Advances in Numerical Heat Transfer*, chap. 3, pp. 97–135, Taylor and Francis, New York, 1997.

5. Belov, A., Martinelli, L., Jameson, A.: A new implicit algorithm with multigrid for unsteady incompressible flow calculations, *AIAA 95-0049*, 1995.

6. Bird, R.B., Stewart, W.E., Lightfoot, E.N.: *Transport phenomena*, Wiley, New York, 1962.

7. Bradshaw, P., Launder, B.E., J.L. Lumley,J.L.: Collaborative testing of turbulence models. In K.N. Ghia, U. Ghia, D. Goldstein (eds.), Advances in Computational Fluid Mechanics, ASME FED, **196**, ASME, New York (1994).

8. Canuto, C., Hussaini, M.Y., Quarteroni, A., Zang, T.A.: *Spectral methods in fluid mechanics*, Springer, Berlin, 1987.

9. Chorin, A.J.: A numerical method for solving incompressible viscous flow problems, *J. Copmput. Phys.*, **2**, 12–?? (1967).

10. Choi, H., Moin, P.: Effects of computational time step on numerical solutions of turbulent flow, *J. Comput. Phys.*, **113**, 1–4 (1994).

11. Chung, T.J.: *Finite element analysis in fluid dynamics*, McGraw-Hill, New York, 1978.

12. Demirdžić, I.: Approximation of derivatives by finite-volume method in elliptic boundary value problems on irregular domains, *ZAMM*, **66**, T298-T300 (1987).

13. Demirdžić, I., Perić, M.: Space conservation law in finite volume calculations of fluid flow, *Int. J. Numer. Methods Fluids*, **8**, 1037–1050 (1988).

14. Demirdžić, I., Perić, M.: Finite volume method for prediction of fluid flow in arbitrarily shaped domains with moving boundaries, *Int. J. Numer. Methods Fluids*, **10**, 771-790 (1990).

15. Demirdžić, I., Lilek, Ž., Perić, M.: A colocated finite volume method for predicting flows at all speeds, *Int. J. Numer. Methods Fluids*, **16**, 1029–1050 (1993).

16. Demirdžić, I., Muzaferija S.: Numerical method for coupled fluid flow, heat transfer and stress analysis using unstructured moving meshes with cells of arbitrary topology, *Comput. Methods Appl. Mech. Engrg.*, **125**, 235–255 (1995).

17. Demirdžić, I., Muzaferija, S., Perić, M.: Advances in computation of heat transfer, fluid flow, and solid body deformation using finite volume approach, in W.J. Minkowycz, E.M. Sparrow (eds.), *Advances in Numerical Heat Transfer*, chap. 2, pp. 59–96, Taylor and Francis, New York, 1997.

18. Ferziger, J.H., Perić, M.: *Computational Methods for Fluid Dynamics*, 3rd ed., Springer, Berlin, 2002.

19. Fletcher, C.A.J.: *Computational techniques for fluid dynamics*, vol. I., Springer, Berlin, 1991.

20. Girault, V., Raviart, P.-A.: *Finite element methods for Navier-Stokes equations*, Springer, Berlin, 1986.

21. Gresho, P.M., Sani, R.L.: On pressure boundary conditions for the incompressible Navier-Stokes equations, *Int. J. Numer. Methods Fluids*, **7**, 11–46 (1990).

22. Hackbusch, W.: *Multi-grid methods and applications*, Springer, Berlin, 1985.

23. Harlow, F.H., Welsh, J.E.: Numerical calculation of time dependent viscous incompressible flow with free surface, *Phys. Fluids*, **8**, 2182–2189 (1965).

24. Henderson, R.D., Karniadakis, G.E.: Unstructured spectral element method for simulation of turbulent flows, *J. Copmput. Phys.*, **122**, 191–217 (1995).

25. Hirsch, C.: *Numerical computation of internal and external flows*, vol. I & II, Wiley, New York, 1991.

26. Hubbard, B.J., Chen, H.C.: A Chimera scheme for incompressible viscous flows with applications to submarine hydrodynamics, *AIAA Paper 94-2210* (1994).

27. Hubbard, B.J., Chen, H.C.: Calculations of unsteady flows around bodies with relative motion using a Chimera RANS method, Proc. *10th ASCE Engineering Mechanics Conference*, vol II, 782-785, Univ. of Colorado at Boulder, Boulder, CO, May 21-24 (1995).

28. Issa, R.I.: Solution of implicitly discretized fluid flow equations by operator-splitting, *J. Copmput. Phys.*, **62**, 40–65 (1986).

29. Karki, K.C., Patankar, S.V.: Pressure based calculation procedure for viscous flows at all speeds in arbitrary configurations, *AIAA J.*, **27**, 1167–1174 (1989)

30. Khosla, P.K., Rubin, S.G.: A diagonally dominant second-order accurate implicit scheme, *Computers Fluids*, **2**, 207–209 (1974).

31. Kwak, D., Chang, J.L.C., Shanks, S.P., Chakravarthy, S.R.: A three-dimensional incompressible Navier-Stokes flow solver using primitive variables, *AIAA J.*, **24**, 390–396 (1986).

32. Leister, H.-J., Perić, M.: Vectorized strongly implicit solving procedure for seven-diagonal coefficient matrix, *Int. J. Num. Meth. Heat Fluid Flow*, **4**, 159–172 (1994).

33. Leonard, B.P.: A stable and accurate convection modelling procedure based on quadratic upstream interpolation, *Copmput. Meth. Appl. Mech. Engrg.*, **19**, 59–98 (1979).

34. Lilek, Ž., Perić, M.: A fourth-order finite volume method with colocated variable arrangement, *Computers Fluids*, **24**, 239–252 (1995).

35. Masson, C., Saabas, H.J., Baliga, R.B.: Co-located equal-order control-volume finite element method for two-dimensional axisymmetric incompressible fluid flow, *Int. J. Numer. Methods Fluids*, **18**, 1–26 (1994).

36. Oden, J.T.: *Finite elements of non-linear continua*, McGraw-Hill, New York, 1972.

37. Patankar, S.V.: *Numerical heat transfer and fluid flow*, McGraw-Hill, New York, 1980.

38. Perić, M.: Efficient semi-implicit solving algorithm for nine-diagonal coefficient matrix, *Numer. Heat Transfer*, **11**, 251–279 (1987).

39. Perić M., Kessler, R., Scheuerer, G.: Comparison of finite volume numerical methods with staggered and colocated grids, *Computers Fluids*, **16**, 389–403 (1988).

40. Perng, C.Y., Street, R.L.: A coupled multigrid–domain-splitting technique for simulating icompressible flows in geometrically complex domains, *Int. J. Numer. Methods Fluids*, **13**, 269–286 (1991).

41. Raithby, G.D., Schneider, G.E.: Numerical solution of problems in incompressible fluid flow: treatment of the velocity-pressure coupling, *Numer. Heat Transfer*, **2**, 417–440 (1979).

42. Rhie, C.M., Chow, W.L.: A numerical study of the turbulent flow past an isolated airfoil with trailing edge separation, *AIAA J.*, **21**, 1525–1532 (1983).

43. Roache, P.J.: Perspective: a method for uniform reporting of grid refinement studies, *ASME J. Fluids Engrg.*, **116**, 405–413 (1994).

44. Rodi, W., Bonnin, J.-C., Buchal, T. (eds.): Proc. ERCOFTAC Workshop on Data Bases and Testing of Caluclation Methods for Turbulent Flows, April 3 - 7, Univ. Karlsruhe, Germany (1995).

45. Saad, Y., Schultz, M.H.: GMRES: a generalized residual algorithm for solving non-symmetric linear systems, *SIAM J. Sci. Stat. Comput.*, **7**, 856–869 (1986).

46. Schneider, G.E., Zedan, M.: A modified strongly implicit procedure for the numerical solution of field problems, *Numer. Heat Transfer*, **4**, 1–19 (1981).

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

48. Schreck, E., Perić, M.: Computation of fluid flow with a parallel multigrid solver, *Int. J. Numer. Methods Fluids*, **16**, 303–327 (1993).

49. Sonar, Th.: On the construction of essentially non-oscillatory finite volume approximations to hyperbolic conservation laws on general triangulations: Polynomial recovery, accuracy and stencil selection, *Comput. Methods Appl. Mech. Engrg.*, **140**, 157–?? (1997).

50. Stone, H.L.: Iterative solution of implicit approximations of multidimensional partial differential equations, *SIAM J. Numer. Anal.*, **5**, 530–558 (1968)

51. Tu, J.Y., Fuchs, L.: Overlapping grids and multigrid methods for three-dimensional unsteady flow calculation in IC engines, *Int. J. Numer. Methods Fluids*, **15**, 693–714 (1992).

52. Van den Vorst, H.A., Sonneveld, P.: CGSTAB, a more smoothly converging variant of CGS. Tech. Report 90-50, Delft University of Technology, 1990.

53. Van den Vorst, H.A.: 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).

54. Van Doormal, J.P., Raithby, G.D.: Enhancements of the SIMPLE method for predicting incompressible fluid flows, *Numer. Heat Transfer*, **7**, 147–163 (1984).

55. Van Doormal, J.P., Raithby, G.D., McDonald, B.H.: The segregated approach to predicting viscous compressible fluid flows, *ASME J. Turbomachinery*, **109**, 268–277 (1987).

56. Weiss, J., Smith, W.A.: Preconditioning applied to variable and constant density flows, *AIAA J.*, **33**, 2050–2057 (1995).

57. Weiss, J., Maruszewski, J.P., Smith, W.A.: Implicit solution of preconditioned Navier-Stokes equations using algebraic multigrid, *AIAA J.*, **37**, 29–36 (1999).

58. Zang, Y., Street, R.L.: A composite multigrid method for calculating unsteady incompressible flows in geometrically complex domains, *Int. J. Numer. Methods Fluids*, **20**, 341–361 (1995).