3 Numerical aspects of CFD

This chapter introduces commonly used numerical methods. The aim is to explain the various methods so that the reader will be able to choose the appropriate method with which to perform CFD simulations. There is an extensive literature on numerical methods, and the interested reader can easily find textbooks. Appropriate references are [3, 4].

3.1 Introduction

In the previous chapter the physics of fluid flow has been presented. The concepts of how the flow is modelled, with and without turbulence, have been introduced. The Navier–Stokes, continuity and pressure equations have been derived, and model equations for some turbulence quantities will be discussed in following chapters. Expressions for heat and mass transfer have also been presented. The expressions for velocity, pressure, turbulence quantities and heat and mass transfer, together with the appropriate boundary conditions, constitute the core of the CFD problem.

So far there has been no discussion of how these equations are solved. This chapter will deal with the most fundamental aspects of numerical procedures for solving problems with CFD. The scope of the chapter is to give the reader a numerical background and an understanding of some of the numerical problems that can occur. Being aware of the existence of these problems and being able to avoid them are of crucial importance.

The *general transport equation for an arbitrary variable* ϕ in conservative form is now stated:

$$\rho\frac{\partial\phi}{\partial t}+\rho\frac{\partial(U_j\phi)}{\partial x_j}=\frac{\partial}{\partial x_j}\left(\Gamma\frac{\partial\phi}{\partial x_j}\right)+S_\phi. \tag{3.1}$$

This equation has been used in the previous chapter, e.g. the Navier–Stokes equations, Eq. (2.21), which actually are nothing other than transport equations for momentum (or rather velocity). The equation also appears as the transport equation for various turbulence quantities in Chapter 4. Unfortunately, it is generally not possible to solve equations of this type analytically, since they are nonlinear and often contain both spatial and temporal derivatives. This requires the application of numerical methods.

3.2 Numerical methods for CFD

The pioneers of CFD employed finite differences to approximate the governing equations describing fluid mechanics. With finite differences, the partial spatial and temporal derivatives appearing in the equations are approximated through Taylor series. Although there is no formal restriction, finite differences are typically employed only on Cartesian geometries. Since most of the problems engineers tackle do not take place in a square box, finite differences are not often used for practical problems.

Finite-element methods require a 2D or 3D mesh and are very flexible in terms of geometry and mesh elements; almost any type of mesh element can be employed. At each mesh element, a base function is used. This base function should locally describe the solution of (part of) the governing equation to be approximated. The finite-element method aims to minimize the difference between the exact solution and the collection of base functions; this can be done e.g. by a Galerkin method. There is no dispute that finite-element methods are the preferred method of choice for solid-mechanics problems.

However, problems in the fluid-mechanics area are generally governed by *local* conservation. For instance, the continuity equation dictates the local conservation of mass. Local conservation is not necessarily a property of the finite-element method, since the difference between the base functions and the exact solution is minimized globally. The adaptation of the finite-element method to reflect local conservation is still very much the focus of numerical research, therefore the method has historically not been used as much for CFD.

3.2.1 The finite-volume method

The principle of the finite-volume method is local conservation, and this is the key reason for its success in CFD. To solve the equations numerically with the finite-volume method, the entire computational domain is divided into 'small' sub-volumes, so-called *cells*. Employing Gauss' law, the partial derivatives expressing a conservation principle, such as div u, can be rewritten at each cell as an algebraic contribution. The governing equation, expressed in the partial differential equations, is reformulated, at each computational cell, into a set of linear algebraic equations

Usually, these equations are solved numerically in an iterative manner. The price for this so-called *discretization* of the domain is the introduction of a numerical error into the solution. It is important to control the magnitude of the error after a solution has been obtained, and, since it can be shown that this error vanishes as the cell size approaches zero, a sufficient decrease of the cell size will often reduce the error well enough. On the other hand, reducing the cell size too much will create an unnecessarily large number of cells, which will increase the computational effort required and possibly yield prohibitive simulation times. Finding a fast but still accurate way of solving the CFD problem is one of the CFD engineer's most important tasks.

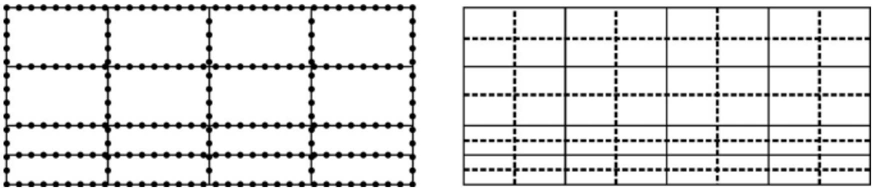

Figure 3.1 Control volumes. The cell-centred algorithm (left) and the node-centred algorithm (right). The mesh elements are depicted with thin lines, and the control volumes with dashed lines.

3.2.2 Geometrical definitions

Now some definitions have to be made. The *cell* has already been defined. Each cell is surrounded by its *faces*. These faces form a grid pattern throughout the domain. A grid that contains only cells with all internal angles equal to 90° is called a *structured grid*, and this is the type of grid that will be dealt with in this chapter. Since a structured grid requires that the physical geometry itself must be rectangular, this type of grid is not very common in reality. Many industrial cases contain parts of complex geometry that cannot be divided into purely rectangular cells. Grids can be created in 1D, 2D or 3D, depending on the number of computational dimensions. A problem can often be regarded as being of 1D or 2D even though all fluid flows are 3D in 'reality'.

In the work of defining the grid it must also be taken into account whether the solver is to use a cell-centred or a node-centred algorithm. A cell-centred solver algorithm creates control volumes that are completely identical to the grid. A node-centred solver creates its control volumes around the grid nodes instead. A grid node is situated at each intersection of cell edges, see also Figure 3.1. Even though choosing the algorithm is primarily a solver issue, the choice has to be made already during the stage of grid creation. This follows as a consequence of the fact that the two algorithms put different demands on the grid, especially in near-wall regions.

3.3 Cell balancing

Integrating the general transport equation, Eq. (3.1), over a control volume (c.v.) yields

$$\int\limits_{\text{c.v.}} \rho \frac{\partial \phi}{\partial t}\,\mathrm{d}V + \int\limits_{\text{c.v.}} \rho \frac{\partial \left(U_j \phi\right)}{\partial x_j}\,\mathrm{d}V = \int\limits_{\text{c.v.}} \frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial \phi}{\partial x_j}\right)\mathrm{d}V + \int\limits_{\text{c.v.}} S_\phi\,\mathrm{d}V. \tag{3.2}$$

For the time being, only steady problems are considered. Thus, the accumulation term will be zero. Transient problems (unsteady problems) are studied in later sections in this chapter.

The next step in solving a problem with finite volumes is to reformulate Eq. (3.2) in algebraic form. This requires the elimination of all integral signs and derivatives. To do

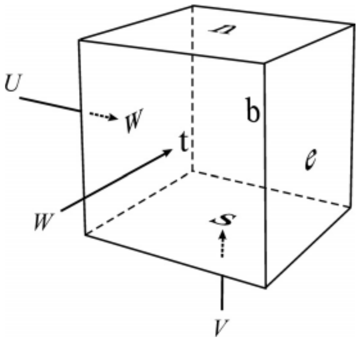

Figure 3.2 The faces of a control volume.

this, some approximations have to be introduced. The following section will discuss the process in detail.

3.3.1 The convective term

The terms in Eq. (3.2) are now studied one by one, starting with the convective term

$$\int_{\text{c.v.}} \rho \frac{\partial (U_j \phi)}{\partial x_j} \, \mathrm{d}V. \tag{3.3}$$

This term represents the *net* flux of ϕ transported out of the cell by convection, i.e. transported with the flow. The flow can enter or leave the cell only through any of its faces. According to Gauss' theorem it is possible to rewrite Eq. (3.3) as

$$\int_{\text{c.s.}} \rho U_j n \phi \, \mathrm{d}A, \tag{3.4}$$

where c.s. denotes the control-volume surfaces, i.e. the faces that surround the cell, and dA is the area that surrounds the volume dV. Here n is a normal vector pointing outwards from dA. The product $U_j n$ is thus the velocity perpendicular to the surface dA. In a structured grid one can evaluate Eq. (3.4) into

$$-\rho \left[(AU\phi)_{\mathrm{w}} - (AU\phi)_{\mathrm{e}} + (AV\phi)_{\mathrm{s}} - (AV\phi)_{\mathrm{n}} + (AW\phi)_{\mathrm{t}} - (AW\phi)_{\mathrm{b}} \right]. \tag{3.5}$$

Here, A is the appropriate face area and U, V and W are the velocities in the x, y and z directions, respectively. The indices w, e, s, n, b and t denote the west, east, south, north, bottom and top cell faces. The term $(AV\phi)_{\mathrm{n}}$ thus takes into account the flux of ϕ through the northern cell face that has size $\Delta x \Delta z$. See also Figure 3.2.

The negative signs in Eq. (3.5) come from the definition of west, east etc. in relation to the coordinate axis. For example, n equals -1 at the west surface, giving that the flux $(U\phi)_{\mathrm{w}}$ is negative if U is positive there. If Eq. (3.5) is regarded as an integral mass balance, it makes sense in the way that what goes into the cell also comes out for source-free, steady conditions when there is no other means of transport than convection.

An apparent problem when evaluating the convective term is that the face values of U_j and ϕ must be known. As mentioned earlier, the transport equations are not solved on the faces. Several approaches to overcome this will be discussed later in the section about discretizing schemes.

3.3.2 The diffusion term

Next, a closer look is taken at the diffusion term,

$$\int_{\text{c.v.}} \frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial \phi}{\partial x_j}\right) \mathrm{d}V. \tag{3.6}$$

The diffusion term takes into account the transport of ϕ by diffusion. Equation (3.6) can be treated in a similar way to the convective term, Eq. (3.3). By making use of Gauss' theorem,

$$\int_{\text{c.s.}} \Gamma \frac{\partial \phi}{\partial x_j} n \,\mathrm{d}A. \tag{3.7}$$

Using the same notation as in the convective case, Eq. (3.7) can be evaluated to give a similar expression:

$$-\left[\left(A\Gamma\frac{\partial \phi}{\partial x}\right)_{\text{w}} - \left(A\Gamma\frac{\partial \phi}{\partial x}\right)_{\text{e}} + \left(A\Gamma\frac{\partial \phi}{\partial y}\right)_{\text{s}} - \left(A\Gamma\frac{\partial \phi}{\partial y}\right)_{\text{n}} + \left(A\Gamma\frac{\partial \phi}{\partial z}\right)_{\text{t}} \left(A\Gamma\frac{\partial \phi}{\partial z}\right)_{\text{b}}\right]. \tag{3.8}$$

3.3.3 The source term

The last term in the general transport equation is the source term,

$$\int_{\text{c.v.}} S_\phi \,\mathrm{d}V. \tag{3.9}$$

The source term takes into account any generation or dissipation of ϕ. The body force due to gravity in the Navier–Stokes equation for the y momentum is an example of a source term as discussed previously. The pressure gradient term in the Navier–Stokes equation is another example. To be able to rewrite Eq. (3.9) without using any integral signs, simply take a cell mean value of S_ϕ and move it outside the integral sign. Thus,

$$\int_{\text{c.v.}} S_\phi \,\mathrm{d}V \approx \overline{S}_\phi V. \tag{3.10}$$

Here, $\overline{S}_\phi$ is the mean value of S_ϕ in the cell.

The original transport equation, Eq. (3.1), is now transformed into equations that can be solved algebraically, Eqs. (3.5), (3.8) and (3.10). It has also been concluded

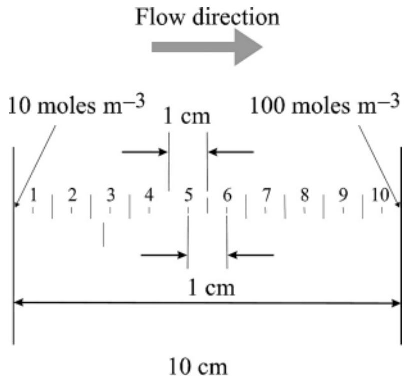

Figure 3.3 The experimental set-up in Example 1.

that the face values of ϕ, Γ and U_j need to be predicted, and the gradient of ϕ at the faces is needed. If a source is present, the correct mean values of the sources in all cells must be created. It must also be remembered that there can be many different scalars, and that they are usually dependent on each other, forcing us to solve their equations simultaneously in each cell. Hence, the set of equations often becomes nonlinear.

Equation (3.5), (3.8) and (3.10) will now be used to solve an easy and clarifying 1D problem, Example 1.

3.4 Example 1 – 1D mass diffusion in a flowing gas

The flow of an inert gas I between two perforated surfaces is simulated. The distance between the surfaces is 10 cm, and it is assumed that the surface areas are infinite. This assumption is made so that the problem can be treated as being 1D. The gas velocity is 1 mm s^{-1}. At the west surface the species concentration is 10 moles m^{-3} of A and at the east surface the concentration is 100 moles m^{-3} of A. Assuming that the total density is 1 kg m^{-3}, the diffusion constant for A in I is 10^{-4} m^2 s^{-1}. The objective is to predict the profile of A between the surfaces, using ten equidistant cells. See Figure 3.3 for the simulation set-up.

3.4.1 Solution

This is a 1D problem. The general transport equation, Eq. (3.1), can be evaluated according to the previous sections with a somewhat simpler expression than for the general 3D case. By way of illustration, the steps are presented. Here, ϕ denotes the molar concentration of A.

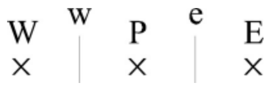

Figure 3.4 Neighbouring cells and cell faces.

Integrating Eq. (3.1) over a 1D control volume forms

$$\int_{w}^{e} \rho \frac{\partial \phi}{\partial t}\,\mathrm{d}x + \int_{w}^{e} \rho \frac{\partial (U\phi)}{\partial x}\,\mathrm{d}x = \int_{w}^{e} \frac{\partial}{\partial x}\left(\Gamma \frac{\partial \phi}{\partial x}\right)\mathrm{d}x + \int_{w}^{e} S_{\phi}\,\mathrm{d}x. \qquad (3.11)$$

Here $\Gamma = \rho D$, where D is the diffusion constant and ρ the total density.

Compared with Eq. (3.2) for the 3D case, it can be seen that the 3D control volumes have been replaced with 1D control volumes. This simply requires integration from the west face to the east instead of over the 3D volume. As shown earlier, it is assumed that steady conditions prevail and thereby it is possible to neglect the accumulation term. Further, there is no internal source of species A in our system. This results in the reduced equation

$$\int_{w}^{e} \rho \frac{\mathrm{d}(U\phi)}{\mathrm{d}x}\,\mathrm{d}x = \int_{w}^{e} \frac{\mathrm{d}}{\mathrm{d}x}\left(\Gamma \frac{\mathrm{d}\phi}{\mathrm{d}x}\right)\mathrm{d}x.$$

This can be evaluated into the algebraic form

$$\left[(\rho U\phi)_{e} - (\rho U\phi)_{w}\right] = \left[\left(\Gamma \frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_{e} - \left(\Gamma \frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_{w}\right]. \qquad (3.12)$$

This equation holds for all cells. To proceed, estimates are required for the face values of ϕ and U and the gradient of ϕ at the faces. This can be done in many ways, but the most straightforward solution is to use a linear interpolation from neighbouring cells. Starting with the face values of ϕ,

$$\begin{aligned} \phi_{w} &= \frac{\phi_{W} + \phi_{P}}{2}, \\ \phi_{e} &= \frac{\phi_{P} + \phi_{E}}{2}. \end{aligned} \qquad (3.13)$$

Here, W denotes the western neighbour cell, E the eastern neighbour cell and P the present cell (w and e are the face values as defined earlier). See also Figure 3.4. Be aware that Eq. (3.13) is merely an approximation and through this a numerical error is introduced into the solution. U is constant at 1 mm s^{-1} so the face values of U are already known. To estimate the gradient of ϕ at the faces, the following equations are used:

$$\begin{aligned} \left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_{w} &= \frac{\phi_{P} - \phi_{W}}{x_{P} - x_{W}}, \\ \left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_{e} &= \frac{\phi_{E} - \phi_{P}}{x_{E} - x_{P}} \end{aligned} \qquad (3.14)$$

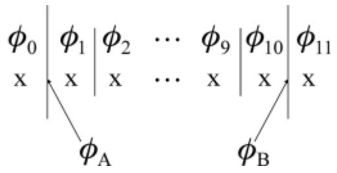

Figure 3.5 The cells. The shadow cells 0 and 11 are situated outside the physical domain. ϕ_A and ϕ_B are the boundary values at the western and eastern physical surfaces, respectively.

This is a first-order Taylor approximation and x_E, x_W and x_P are the cell coordinates of the eastern, western and present cells, respectively. Also, by using Eq. (3.14) a numerical error is introduced.

Equations (3.12), (3.13) and (3.14) can now be combined, which results in

$$\left[\left(\rho U\frac{\phi_P+\phi_E}{2}\right)-\left(\rho U\frac{\phi_P+\phi_W}{2}\right)\right]=\left[\left(\Gamma\frac{\phi_E-\phi_P}{x_E-x_P}\right)-\left(\Gamma\frac{\phi_P-\phi_W}{x_P-x_W}\right)\right]. \tag{3.15}$$

Equation (3.15) can be solved once for each cell, giving a total of ten equations (one per cell). Each equation will contain three unknowns, ϕ_W, ϕ_E and ϕ_P. However, the unknown value of, for example, ϕ_E in the equation for cell number 4 will come back as ϕ_P on solving Eq. (3.15) for cell number 5. Thus, the total number of unknown variables will be 12; one ϕ per cell plus two extra ϕ values for the farmost eastern and farmost western surfaces, i.e. the boundaries. Numerical values for the boundaries are given in the assignment, giving ten equations and ten unknown variables. The equation system can thus be solved and the profile determined.

Before solving the equation system, Eq. (3.15) is re-arranged in a slightly different form:

$$\underbrace{\left(\frac{\Gamma}{x_E-x_P}+\frac{\Gamma}{x_P-x_W}\right)}_{B}\phi_P+\underbrace{\left(\frac{\rho U}{2}-\frac{\Gamma}{x_E-x_P}\right)}_{C}\phi_E+\underbrace{\left(-\frac{\rho U}{2}-\frac{\Gamma}{x_P-x_W}\right)}_{A}\phi_W=0. \tag{3.16}$$

Cells 2–9 are identical in the sense that the distances to the neighbouring cells are identical. For the cells that are closest to the boundaries, i.e. cells 1 and 10, there are no cells directly to the west and east, respectively. These cells thus require extra attention. A way to avoid having to treat these two cells in a special way is to create two ‘imaginary’ cells, so-called *shadow cells*, outside the computational domain. If the shadow cells are placed so that the distance from them to the closest real cell is the same as the distance between two real cells, then cells 1 and 10 are not treated any differently. This will make the solution procedure easier. See Figure 3.5 for more details. The shadow cells are called ϕ_0 and ϕ_{11}, respectively, and the boundary values are called ϕ_A and ϕ_B. Using linear interpolation, the shadow cell is calculated from $\phi_A = (\phi_0+\phi_1)/2$.

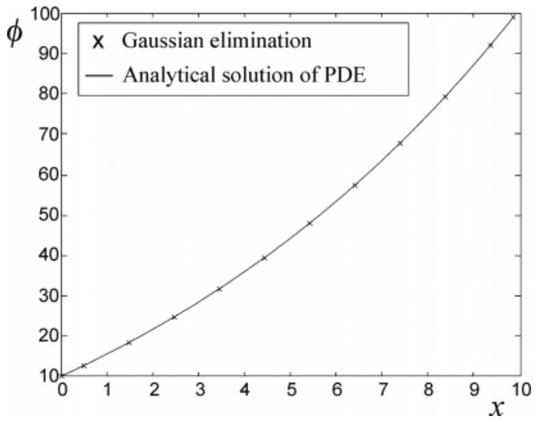

Figure 3.6 A plot of molar concentration as a function of distance from the left plate solved with Gaussian elimination. The exact solution of the PDE has been added for comparison.

Ten identical equations are now arranged for cells 1–10 and two slightly modified ones for the shadow cells. In matrix form,

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ A & B & C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & A & B & C & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & A & B & C & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A & B & C & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & A & B & C & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A & B & C & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A & B & C & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & A & B & C & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A & B & C & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A & B & C \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} \phi_0 \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \\ \phi_{10} \\ \phi_{11} \end{pmatrix} = \begin{pmatrix} 2\phi_A \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2\phi_B \end{pmatrix}, \quad (3.17)$$

where the constants A, B and C are defined in Eq. (3.16). Since the coefficients A, B and C are not functions of ϕ_i, it is possible to solve Eq. (3.17) analytically with Gaussian elimination. This gives the following numerical values of ϕ_i at the cells: $\phi_0 = 7.38$, $\phi_1 = 12.61$, $\phi_2 = 18.39$, $\phi_3 = 24.78$, $\phi_4 = 31.84$, $\phi_5 = 39.65$, $\phi_6 = 48.28$, $\phi_7 = 57.81$, $\phi_8 = 68.35$, $\phi_9 = 80.00$, $\phi_{10} = 92.88$ and $\phi_{11} = 107.11$.

Remember that cells 0 and 11 are shadow cells and are used only to make the solution procedure easier. They are listed here for paedagogical reasons. For a plot of the results, please refer to Figure 3.6. It can be observed that there is only a very small deviation from the exact solution of Eq. (3.11) even though the approximations in Eq. (3.13) and Eq. (3.14) have been introduced.

3.4.2 Concluding remarks

The solution strategy for a simple CFD problem has now been demonstrated. However, it must be kept in mind that the solved example contained many simplifications and was well defined in many ways. Generally, it is not possible to solve a CFD problem with a *direct* method, such as Gaussian elimination. The focus is then turned towards *iterative* methods.

Some examples of simplifications in the solved Example 1 are the following.

- The problem could be treated as 1D due to symmetries. A problem in 2D or 3D would, of course, generate more cells; in this case a 3D treatment would give 1000 cells instead of 10, assuming that the grid density was kept constant and the computational domain had a cubic geometry. The cells were placed with constant spacing, generating a so-called *equidistant* grid.
- Further, the presence of a constant velocity made the solution process easier. Usually, the velocities cannot be predetermined, and thus one requires a solution to a transport equation for each velocity component as well. If this is the case, the equation system, Eq. (3.16), will contain nonlinear terms like $\rho U\phi$, where both U and ϕ are variables. This means that matrices can no longer be created with constant coefficients like in Eq. (3.17). The equation system must then be solved by some iterative technique, e.g. the *Gauss–Seidel* method (see below for more details).
- Another simplification introduced was the linear approximations of the variables and the gradients at the faces. Obviously, at high positive flow rates, the face values between, for example, cell 1 and cell 2 must be more dependent on cell 1 than on cell 2. Thus, linear interpolation, or *central differencing*, must be used with caution.
- The fluid properties were assumed to be constant. In non-isothermal situations, the temperature must be calculated in order for these entities to be predicted. A transport equation for temperature has to be solved. However, in order to solve the energy equation, the fluid properties must be known. This requires iterative methods.

Questions

(1) Propose a general expression for the transport of the entity ϕ. Give a physical interpretation and units to the terms.
(2) What is discretization? Is it always a necessary step in the solving of a CFD problem?
(3) How many boundary conditions are required in order to solve a steady diffusion–convection problem in n computational dimensions? Does this number change if diffusion is neglected? Give a physical explanation!
(4) What sources of error were introduced in Example 1?

3.5 The Gauss–Seidel algorithm

As discussed earlier, iterative methods are always used in CFD. Most commercial CFD codes use some variant of the Gauss–Seidel algorithm (GSA). Some basic knowledge of the GSA makes it easier to understand how these codes work and also what problems can occur.

For demonstration, Example 1 will be solved again, this time with the GSA. First, some general words about the GSA are in order. Starting with Eq. (3.16), it is possible to isolate ϕ_P from the equation:

$$\phi_P = \frac{a_E\phi_E + a_W\phi_W}{a_P}, \tag{3.18}$$

where

$$\begin{aligned} a_E &= \left(-\frac{\rho U}{2} + \frac{\Gamma}{x_E - x_P}\right), \\ a_W &= \left(\frac{\rho U}{2} + \frac{\Gamma}{x_P - x_W}\right), \\ a_P &= \left(\frac{\Gamma}{x_E - x_P} + \frac{\Gamma}{x_P - x_W}\right). \end{aligned} \tag{3.19}$$

The main difference between using the GSA and Gaussian elimination is that the GSA is an *iterative* method. Equation (3.18) is solved for each cell in an iterative manner. Start with cell 1; to be able to solve for this cell, the numerical value of the variable in the shadow cell, ϕ_0, and also the numerical value in cell 2, ϕ_2, are needed. Since there is no numerical value of ϕ in any cell, a starting guess, an *initialization*, for all ϕ_i must be made. ϕ_1 is then solved for. Shifting our attention to ϕ_2, Eq. (3.18) is solved for this cell. In the calculation of ϕ_2, the calculated value of ϕ_1 and the starting guess for ϕ_3 are used. Thus, the GSA uses the values calculated already in the same iteration sweep, which makes the solution converge faster. The procedure is then carried out until ϕ_i in all cells have been calculated. But, there is a problem here; since ϕ_1 was calculated as a function of ϕ_0 and ϕ_2, the equation for ϕ_1 is no longer satisfied since a new value of ϕ_2 has been calculated. Thus, the procedure must be repeated many times; i.e. it requires *iteration*. This is done until *convergence* is reached, i.e. the numerical values of ϕ_i change by less than a specified threshold amount set by the user. Discussion of how this is usually done in practice will come later.

3.6 Example 2 – Gauss–Seidel

This time, Example 1 will be solved numerically using the GSA. The procedure given in Figure 3.7 will be followed.

On looking at the expressions for a_W, a_E and a_P, one may see that they are identical for all cells and do not change during the iterations. We use the numerical values given in the assignment, $a_E = 0.0095$, $a_W = 0.0105$ and $a_P = 0.0200$, for all cells. A high numerical value of e.g. a_E means that cell (P) is highly influenced by it, whereas a low number means the opposite; see Eq. (3.18). As can be seen here, a_W is larger than a_E, meaning that the value for cell P is more influenced by its western neighbour than by its eastern one. This shouldn't come as any surprise, knowing that the gas is flowing from west to east and, thus, the western cell should play a more dominant role in the calculation of cell P.

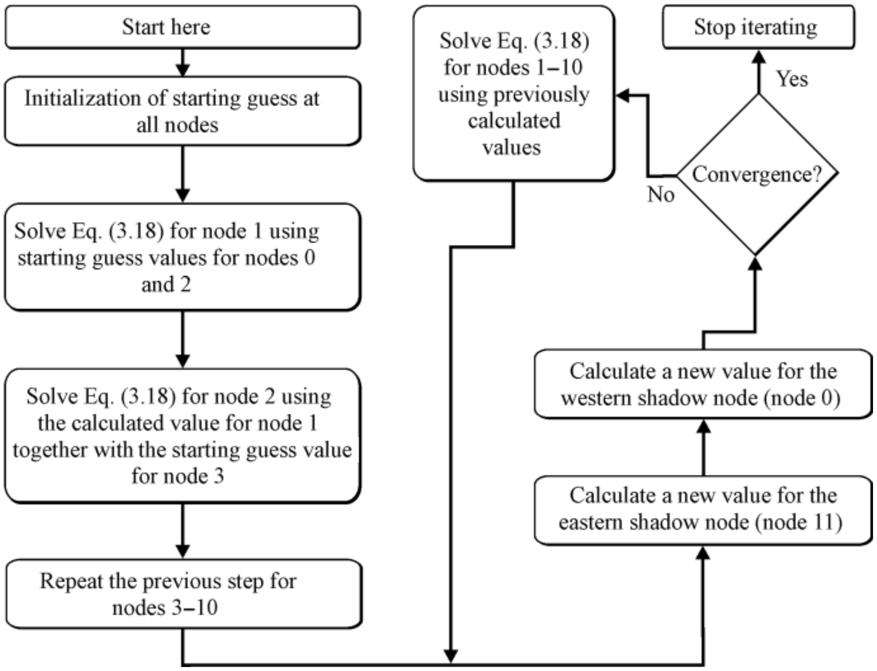
Start here
Initialization of starting guess at all nodes
Solve Eq. (3.18) for node 1 using starting guess values for nodes 0 and 2
Solve Eq. (3.18) for node 2 using the calculated value for node 1 together with the starting guess value for node 3
Repeat the previous step for nodes 3–10
Solve Eq. (3.18) for nodes 1–10 using previously calculated values
Stop iterating
Yes
Convergence?
No
Calculate a new value for the western shadow node (node 0)
Calculate a new value for the eastern shadow node (node 11)

Figure 3.7 The Gauss–Seidel solving procedure for Example 2.

Next, starting values for all cells are initialized. For simplicity, set all ϕ_i equal to 50 moles m^{-3}. Making a 'good' starting guess is not that important in this example, but in more complex problems the starting guess can be vital even to reach convergence. This will be discussed further in the section about unsteady problems. Special attention is required for the shadow cells. As discussed earlier, it was through these cells that the boundary conditions were introduced. Given the numerical value of the boundaries, the numerical value of the shadow cells can be determined.

$$\phi_0 = 2\phi_A - \phi_1 = -30,$$
$$\phi_{11} = 2\phi_B - \phi_{10} = 150.$$

Now, Eq. (3.18) can be solved for all cells, beginning with cell 1 (see also Figure 3.7):

$$\begin{aligned}
\phi_1 &= \frac{0.0095 \times 50 + 0.0105 \times (-30)}{0.0200} = 8,\\
\phi_2 &= \frac{0.0095 \times 50 + 0.0105 \times 8}{0.0200} = 27.95,\\
\phi_3 &= \frac{0.0095 \times 50 + 0.0105 \times 27.95}{0.0200} = 38.42,\\
&\dots\\
\phi_9 &= \frac{0.0095 \times 50 + 0.0105 \times 49.53}{0.0200} = 49.75,\\
\phi_{10} &= \frac{0.0095 \times 150 + 0.0105 \times 49.75}{0.0200} = 97.37.
\end{aligned}$$

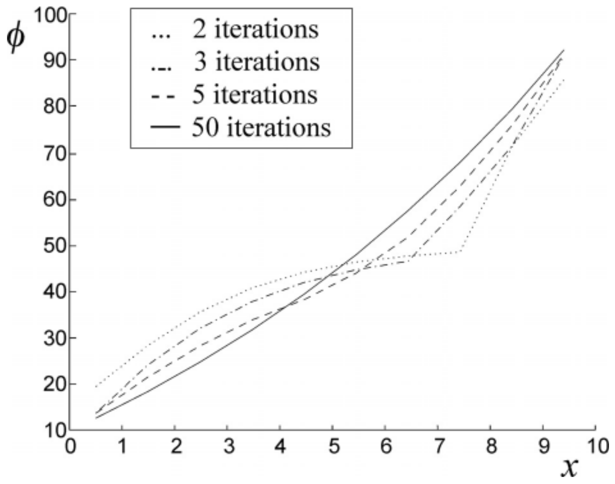

Figure 3.8 A plot of molar concentration as a function of the distance from the left plate at different stages in the iterative process.

On looking at the numerical values of the cells, a tendency for the low value at the western boundary to ‘spread’ through the domain can be seen. This is because iterations are sweeping from west to east, not vice versa. Iterating from east to west would have resulted in a profile, at this stage, with cell values higher than 50 for almost all cells. In the present iteration, information is transported from west to east, from low cell numbers to higher cell numbers. At convergence, the mode of iteration should not have any impact on the result, but it often affects the computational time that is required for convergence.

Additional iterations are run before continuing the discussion. For iteration 40 $\phi_1 = 12.6027$, $\phi_2 = 18.3587$, $\phi_3 = 24.7271$, $\phi_4 = 31.7741$, $\phi_5 = 39.5721$, $\phi_6 = 48.2000$, $\phi_7 = 57.7443$, $\phi_8 = 68.2998$, $\phi_9 = 79.9706$ and $\phi_{10} = 92.8714$. See Figure 3.8.

From the numerical data produced during the iterative process, it is possible to see that the rate of change of a numerical value at a specific cell decreases with iteration. This is also shown in Figure 3.9, where all cell values have been plotted against the number of iterations. Eventually, after continuing iterations for a ‘very long’ time, the same solution as was obtained in Example 1 will be reached, but already after a few iterations this solution is very close. Notice that there have been no additional sources of error introduced into the equations apart from the previously discussed discretization error and the error stemming from face-value or face-gradient approximations. However, if the iterations are interrupted ‘too soon’, the solution obtained can be far from the ‘true numerical’ solution. Thus, knowing when to stop iterating is important. In previous sections, the concept of convergence was briefly discussed; and it was stated that a solution is converged when it’s ‘close enough’ to the solution to the original set of partial differential equations (PDEs). A proper measure of convergence will now be discussed.

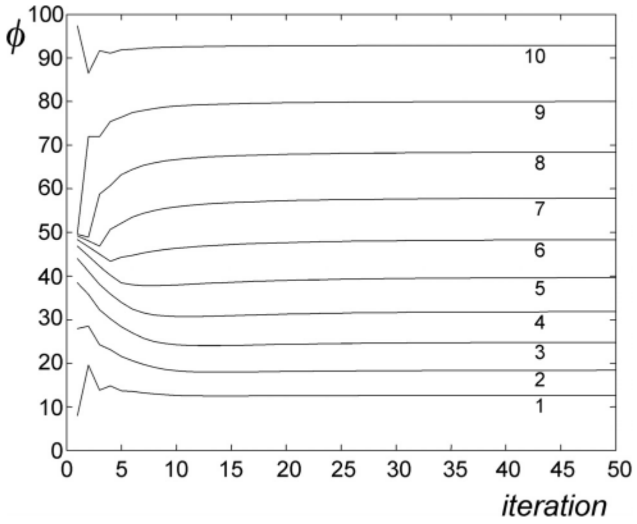

Figure 3.9 A plot of molar concentration at the cells during iterations.

3.7 Measures of convergence

The objective with measuring convergence is to know when to stop a CFD simulation. The code should iterate until a criterion is fulfilled. It should then stop. There are several different approaches that can be employed to find out when a solution has converged. Since the exact solution to the set of PDEs is generally unknown (otherwise there would be no need for any iterations), it is not possible to compare the numerical solution with this exact solution. Other methods must be found. Maybe the easiest one is to state that 'when no cell value differs by more than a small threshold, for example 0.01, from iteration to iteration, the solution has converged'. In Example 2, this would probably be quite a good measure. On the other hand, if the boundary values were very small, like 0.001 for the western boundary and 0.002 for the eastern boundary, then the solution would probably have converged before we'd even started iterating. The cell values would not change enough in an absolute sense. So just looking at the absolute difference

$$\left|\phi_i^{\text{new}} - \phi_i^{\text{old}}\right| < \varepsilon_{\text{thres}}$$

is not always an appropriate measure.

A similar approach would instead be to study the relative change

$$\left|\frac{\phi_i^{\text{new}} - \phi_i^{\text{old}}}{\phi_i^{\text{old}}}\right| < \varepsilon_{\text{thres}}.$$

A problem with this approach can occur if the cell values are very close to zero, since then the relative change would always be very large, and numerical problems can arise. However, this approach is used quite often, primarily because it's relatively easy to implement in a code and it turns out to be quite good.

A very common approach is to calculate the error in Eq. (3.18),

$$R = \sum_{\text{all cells}} |a_P\phi_P - a_W\phi_W - a_E\phi_E|. \tag{3.20}$$

If this error is scaled with an appropriate factor, a dimensionless number on which one can impose a criterion results. For example, if F is an appropriate scaling factor,

$$\frac{R}{F} < \varepsilon_{\text{thres}}, \qquad \text{e.g.} \qquad F = \sum_{\text{all cells}} |a_p\phi_p| + |a_W\phi_W| + |a_E\phi_E|.$$

In some commercial CFD software the scaling factor is simply set to the highest unscaled residual obtained within the first iterations. Hence, a bad starting guess makes it easier to reach convergence, and vice versa.

A completely different way to measure convergence is to check whether the domain upholds conservativeness. In the previous examples one could check the amount of A being transported into the domain and subtract the amount that is being transported out from it. For convergence, this difference should of course be zero, since there are no sources inside the domain. In mathematical terms, the total flux (convection plus diffusion) is given as F and this criterion is expressed as

$$|F_W + F_E| < \varepsilon_{\text{thres}}.$$

It must also be mentioned here that $\varepsilon_{\text{thres}}$ can take different numerical values for different variables. The numerical values of $\varepsilon_{\text{thres}}$ are often in the range 10^{-3}–10^{-6} for single precision modes depending on which definition of a convergent solution is used. Probably, the best way to ensure convergence is a combination of the criteria described above. Being able to determine convergence is one of the most crucial parts of CFD. Almost all commercial codes have various built-in functions to check convergence, but nevertheless it is often worth the extra effort to do some extra manual checks. Remember that a solution that has not converged is an 'incorrect' solution.

More about convergence can be found in Chapter 7.

Question

(1) In Examples 1 and 2 the same problem was solved in two completely different ways. Which solution procedure puts the highest demand on the computer in terms of memory usage and processor performance?

3.8 Discretization schemes

Look back at Example 1. When Eq. (3.18) was solved for the cells, the numerical values of a_E, a_W and a_P were used. These were obtained from Eq. (3.19). In turn, these expressions relied on the assumptions made in Eqs. (3.13) and (3.14). A linear interpolation was assumed in order to get accurate face values for ϕ. On the other hand, it was later argued that this was an assumption that lacks physical reliability in cases with strong convection. This means that the western-face value of, for example, cell 4 should be more influenced by cell 3 than by cell 4 as a consequence of the flow direction.

To investigate the possible effects of non-physical assumptions, Example 1 is solved yet again, now with an increased flow rate.

3.8.1 Example 3 – increased velocity

The same data as in Example 1 are used, but the gas velocity is now 5 cm s^{-1}. Using the same solution procedure as in Example 2, Eq. (3.19) gives

$$\begin{aligned} a_\mathrm{E} &= -0.0150, \\ a_\mathrm{W} &= 0.0350, \\ a_\mathrm{P} &= 0.0200. \end{aligned} \tag{3.21}$$

Equation (3.18) is now solved for all cells. One Gauss–Seidel iteration with the same starting guess as in Example 2 gives $\phi_1 = -90.0000$, $\phi_2 = -195.0000$, $\phi_3 = -378.7500$, $\phi_4 = -700.3125$, $\phi_5 = -1263.0469$, $\phi_6 = -2247.8320$, $\phi_7 = -3971.2061$, $\phi_8 = -6987.1106$, $\phi_9 = -12\,264.9435$ and $\phi_{10} = -21\,576.1512$.

If some more iterations are run, the numerical values of ϕ will become even larger. The values will fluctuate between very large positive values and very large negative values from iteration to iteration. This behaviour is typical for a *diverged* solution. A diverged solution gives incorrect results. In this case it is quite obvious since ϕ represents the molar concentration of species A, and of course can't take negative values. The question one should ask is that of why the solution diverged. The only difference from the successfully solved Example 2 is that the flow rate was increased and thereby the convective transport became more dominant.

Perhaps a look at the value of a_E, Eq. (3.21), can give a hint regarding what's wrong. It has already been concluded that the value of the coefficient is a measure of the strength of the interaction between the eastern and present cells. A high number means a strong interaction, and vice versa. With no interaction, the coefficient should be zero. A value below zero lacks physical reliability. Thus, there is reason to suspect that the failure stems from the 'incorrect' coefficient.

Equation (3.19) has been used to calculate the coefficients. These relations build on the assumption that the face values of ϕ can be estimated using a mean value of the neighbouring cells. As shown previously, this assumption cannot be used in cases with strong convection. In the present case, strong convection means that

$$a_\mathrm{E} < 0 \rightarrow \frac{\rho U}{2} > \frac{\Gamma}{x_\mathrm{E} - x_\mathrm{P}}. \tag{3.22}$$

This criterion can also be expressed in terms of a dimensionless number. The Péclet number is defined as the ratio between convective mass transfer and diffusive mass transfer,

$$Pe = \frac{\rho U}{\Gamma/(x_\mathrm{E} - x_\mathrm{P})} = \frac{\rho U (x_\mathrm{E} - x_\mathrm{P})}{\Gamma}. \tag{3.23}$$

The criterion for strong convection would then be (cf. Eq. (3.22))

$$|Pe| > 2. \tag{3.24}$$

Here, it is assumed that the velocity is always positive, i.e. the flow is from west to east. However, the flow could just as well go in the opposite direction, giving a negative value of a_W instead. This would also have resulted in a divergent solution. The absolute sign for *Pe* in Eq. (3.24) evolves from this fact. In Example 3, $Pe = 5$, which satisfies Eq. (3.24) and hence makes the problem diverge.

This example illustrates that it's important to keep in mind that all assumptions must have physical reliability. This will always be important in CFD, not only in the numerical aspects, but also in other parts of the CFD problem. When the various turbulence models are introduced in following chapters it is stated that each model has its physical limitations and that overlooking these limitations can result in an incorrect solution. Bearing in mind the physical background of the problem is thus always important.

3.8.2 Boundedness and transportiveness

A desired property of a discretization scheme is that it should uphold *boundedness*. A *bounded* variable has a value that is neither larger nor smaller than any of the values that are used to calculate it. In Example 1 central differencing was used to estimate the face values of ϕ. The face value was then simply the mean value of the two surrounding cells. Thus, the face values are bounded. The cell values were calculated using Eq. (3.18), which actually is nothing other than a weighted mean value of the two neighbouring cells. This follows from the fact that $a_W + a_E = a_P$. On the other hand, Eq. (3.18) can be seen as a mean value only for cases with positive values of the coefficients. If one of the values is negative, it is no longer possible to say that ϕ_P is bounded, since it can take values that are either larger or smaller than those for any of the neighbouring cells. Thus, the central-differencing scheme is said to be *conditionally bounded*. It should be noted that there are more advanced central-differencing schemes that are bounded, but they are beyond the scope of this book.

Transportiveness is another desired property of the schemes. It has already been discussed briefly, but not defined properly. The ability of the numerical scheme to 'feel' in what direction the information is being transported is a description of transportiveness. As previously discussed, the solution in Example 3 failed partly due to the lack of this property. Transportiveness is linked to boundedness in this case, since lack of transportiveness gave an unbounded solution. The central-differencing scheme calculates the face values without taking any notice of the direction of the flow, i.e. the direction of the information. Thus, the central-differencing scheme does not satisfy the transportiveness requirement.

3.8.3 The upwind schemes

From Example 3, and the discussion above, it is clear that in cases with strong convection the face values of ϕ have to be estimated in some other way than using Eq. (3.13). A reasonable line is to suggest that the face values between, for example, cell 4 and cell 5 are dependent only upon cell values from cell 4 or cells further upstream. Schemes that let face values be dependent only on upstream conditions are called *upwind schemes*. We will now examine two of these schemes closer.

First-order upwind

The idea behind the first-order upwind scheme is to have physical reliability for convective flows simply by letting the face value between two cells be equal to the value for the nearest upstream cell, i.e.

$$\begin{aligned} \phi_{\mathrm{w}} &= \phi_{\mathrm{W}}, \\ \phi_{\mathrm{e}} &= \phi_{\mathrm{P}}, \end{aligned} \tag{3.25}$$

or, for negative velocities,

$$\begin{aligned} \phi_{\mathrm{w}} &= \phi_{\mathrm{P}}, \\ \phi_{\mathrm{e}} &= \phi_{\mathrm{E}}. \end{aligned} \tag{3.26}$$

The gradients are still estimated using Eq. (3.14).

If we return to our previous examples and use the first-order upwind scheme instead of the central-differencing scheme, we can rewrite Eq. (3.15) as

$$[(\rho U \phi_{\mathrm{P}}) - (\rho U \phi_{\mathrm{W}})] = \left[\left(\Gamma \frac{\phi_{\mathrm{E}} - \phi_{\mathrm{P}}}{x_{\mathrm{E}} - x_{\mathrm{P}}} \right) - \left(\Gamma \frac{\phi_{\mathrm{P}} - \phi_{\mathrm{W}}}{x_{\mathrm{P}} - x_{\mathrm{W}}} \right) \right]. \tag{3.27}$$

The physical meaning of the terms in the equation is the same as before; the difference in convective transport of ϕ is balanced out by the difference in diffusion. The only difference from Eq. (3.15) is that face values of ϕ have been expressed using the first-order upwind scheme instead of the central-differencing scheme. If we write Eq. (3.27) in the same form as Eq. (3.18), we get

$$\begin{aligned} a_{\mathrm{E}} &= \frac{\Gamma}{x_{\mathrm{E}} - x_{\mathrm{P}}}, \\ a_{\mathrm{W}} &= \frac{\Gamma}{x_{\mathrm{P}} - x_{\mathrm{W}}} + \rho U, \\ a_{\mathrm{P}} &= \rho U + \frac{\Gamma}{x_{\mathrm{E}} - x_{\mathrm{P}}} + \frac{\Gamma}{x_{\mathrm{P}} - x_{\mathrm{W}}}. \end{aligned} \tag{3.28}$$

Then, 40 iterations with the GSA will yield the following values for the cells: $\phi_1 = 10.0004$, $\phi_2 = 10.0003$, $\phi_3 = 10.0003$, $\phi_4 = 10.0007$, $\phi_5 = 10.0034$, $\phi_6 = 10.0199$, $\phi_7 = 10.1191$, $\phi_8 = 10.7143$, $\phi_9 = 14.2858$ and $\phi_{10} = 35.7143$.

The results are as expected; the high flow rate makes the western boundary more dominant. Almost all cells in the domain, except the most eastern, have much the same value of ϕ of approximately 10. Again, this seems reasonable.

From Eq. (3.28) it follows that the first-order upwind scheme is bounded. It also fulfils the requirement of transportiveness since care is taken regarding the direction of the flow, cf. Eqs. (3.25) and (3.26). However, it also overestimates the transport of entities in the flow direction. This gives rise to so-called *numerical diffusion*.

Second-order upwind

To improve accuracy – we will discuss accuracy in more detail later – there is an upwind scheme that predicts the face values using information from two upwind cells. To estimate the eastern-face value, the scheme assumes that the gradient between the present cell and the eastern face is the same as that between the western cell and the present cell. In mathematical terms,

$$\frac{\phi_e - \phi_P}{x_e - x_P} = \frac{\phi_P - \phi_W}{x_P - x_W} \rightarrow \phi_e = \frac{(\phi_P - \phi_W)(x_e - x_P)}{x_P - x_W} + \phi_P. \tag{3.29}$$

For an equidistant grid, Eq. (3.29) gives that

$$\phi_e = 1.5\phi_P - 0.5\phi_W. \tag{3.30}$$

A major drawback with the second-order upwind scheme is that it is *unbounded*. To avoid the numerical problems that often arise as a result of unbounded schemes, some *bounded* second-order schemes have been developed, e.g. the van Leer scheme. The definitions will be stated here.

The van Leer scheme

If $|\phi_E - 2\phi_P + \phi_W| \leq |\phi_E - \phi_W|$, the value of ϕ at the eastern face is (cf. Eq. (3.29))

$$\phi_e = \phi_P + \frac{(\phi_E - \phi_P)(\phi_P - \phi_W)}{\phi_E - \phi_W}. \tag{3.31}$$

Otherwise (cf. Eq. (3.25)),

$$\phi_e = \phi_P. \tag{3.32}$$

The velocity is assumed to be positive. The van Leer scheme implements the unbounded second-order upwind scheme, Eq. (3.31), if the gradient is 'smooth', i.e. the second derivative of ϕ is 'small'. Otherwise, the first-order upwind scheme, Eq. (3.32), is used.

3.8.4 Taylor expansions

Before proceeding, a short mathematical review of Taylor expansions will be given.

Taylor's theorem for a 1D expansion of a real function $f(x)$ about a point $x = x_0$ is given without a proof:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2!}f''(x_0) + \cdots$$
$$+ \frac{(x - x_0)^n}{n!}f^{(n)}(x_0) + \int_{x_0}^{x} \frac{(x - u)^n}{n!}f^{(n+1)}(u)\mathrm{d}u. \tag{3.33}$$

The last term in Eq. (3.33) is called the *Lagrange remainder*. Taylor's theorem, except for the Lagrange remainder, was devised by the English mathematician Brook Taylor in 1712 and published in *Methodus in crementorum directa et inversa* in 1715. The more terms are included in the series, the more accurate the estimation will be. Taylor expansions will be used when discussing accuracy.

3.8.5 Accuracy

Two different types of discretization schemes have been presented, the central-differencing scheme and the upwind schemes, one that fulfils the requirement of transportiveness and one that does not. The last section concluded that for problems with strong convection the central-differencing scheme failed. The first-order upwind scheme was then used instead. On the other hand, the first-order upwind scheme used only one cell to estimate the face value, compared with two cells for the central-differencing scheme, and thus we should expect the first-order upwind scheme to be less *accurate* than the central-differencing scheme. Generally, the more information is used, the better the estimation. Accuracy can be quantified in several ways.

For the central-differencing scheme, we have

$$\begin{aligned}\phi_{\mathrm{w}} &= \frac{\phi_{\mathrm{W}}+\phi_{\mathrm{P}}}{2},\\ \phi_{\mathrm{e}} &= \frac{\phi_{\mathrm{P}}+\phi_{\mathrm{E}}}{2}\end{aligned} \tag{3.13}$$

and

$$\begin{aligned}\left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_{\mathrm{w}} &= \frac{\phi_{\mathrm{P}}-\phi_{\mathrm{W}}}{x_{\mathrm{P}}-x_{\mathrm{W}}},\\ \left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_{\mathrm{e}} &= \frac{\phi_{\mathrm{E}}-\phi_{\mathrm{P}}}{x_{\mathrm{E}}-x_{\mathrm{P}}}.\end{aligned} \tag{3.14}$$

If it is assumed that the grid is equidistant (this has been the case in all our examples so far), the *grid spacing* can be defined as $\Delta x = x_{\mathrm{P}} - x_{\mathrm{W}} = x_{\mathrm{E}} - x_{\mathrm{P}}$. If a Taylor expansion of ϕ_{E} and ϕ_{P} is made about x_{e}, the following result is reached:

$$\begin{aligned}\phi_{\mathrm{E}} = \phi_{\mathrm{e}} &+ (\Delta x/2)\left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_{\mathrm{e}} + \frac{(\Delta x/2)^2}{2}\left(\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}\right)_{\mathrm{e}} + \frac{(\Delta x/2)^3}{6}\left(\frac{\mathrm{d}^3\phi}{\mathrm{d}x^3}\right)_{\mathrm{e}}\\ &+ \frac{(\Delta x/2)^4}{24}\left(\frac{\mathrm{d}^4\phi}{\mathrm{d}x^4}\right)_{\mathrm{e}} + O\left[(\Delta x)^5\right],\end{aligned} \tag{3.34}$$

$$\begin{aligned}\phi_{\mathrm{P}} = \phi_{\mathrm{e}} &- (\Delta x/2)\left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_{\mathrm{e}} + \frac{(\Delta x/2)^2}{2}\left(\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}\right)_{\mathrm{e}} - \frac{(\Delta x/2)^3}{6}\left(\frac{\mathrm{d}^3\phi}{\mathrm{d}x^3}\right)_{\mathrm{e}}\\ &+ \frac{(\Delta x/2)^4}{24}\left(\frac{\mathrm{d}^4\phi}{\mathrm{d}x^4}\right)_{\mathrm{e}} + O\left[(\Delta x)^5\right].\end{aligned} \tag{3.35}$$

Here, $O[(\Delta x)^n]$ is the *truncation error*. Next, Eqs. (3.34) and (3.35) are inserted into the right-hand side of the second equation in Eqs. (3.13) and (3.14), which results in

$$\frac{\phi_{\mathrm{P}}+\phi_{\mathrm{E}}}{2} = \phi_{\mathrm{e}} + \frac{(\Delta x)^2}{8}\left(\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}\right)_{\mathrm{e}} + \frac{(\Delta x)^4}{384}\left(\frac{\mathrm{d}^4\phi}{\mathrm{d}x^4}\right) + O\left[(\Delta x)^6\right], \tag{3.36}$$

$$\frac{\phi_{\mathrm{E}}-\phi_{\mathrm{P}}}{\Delta x} = \left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_{\mathrm{e}} + \frac{(\Delta x)^2}{24}\left(\frac{\mathrm{d}^3\phi}{\mathrm{d}x^3}\right)_{\mathrm{e}} + O\left[(\Delta x)^5\right]. \tag{3.37}$$

According to Eqs. (3.13) and (3.14), these expressions should be equal to the face value of ϕ and the gradient of ϕ at the eastern face, respectively, *assuming that central differencing is used*. Thus, since the second-order derivative $\mathrm{d}^2\phi/\mathrm{d}x^2$ is unknown,

$$\phi_\mathrm{e} = \phi_\mathrm{e}^\mathrm{CD} + O\left[(\Delta x)^2\right] \tag{3.38}$$

and

$$\left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_\mathrm{e} = \left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_\mathrm{e}^\mathrm{CD} + O\left[(\Delta x)^2\right]. \tag{3.39}$$

Before we comment on the results, we repeat the same procedure for the first-order upwind scheme. Since the face gradient is predicted in the same way as with central differencing, the face-value estimation, i.e. Eq. (3.25), must be examined. According to this relation, the face value of the eastern face is simply equal to the cell value in the present cell. A Taylor expansion of ϕ about x_P gives

$$\phi_\mathrm{e} = \phi_\mathrm{P} + (\Delta x/2)\left(\frac{\mathrm{d}\phi}{\mathrm{d}x}\right)_\mathrm{P} + \frac{(\Delta x/2)^2}{2}\left(\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}\right)_\mathrm{P} + O\left[(\Delta x)^3\right]. \tag{3.40}$$

Here, the first-order derivative is unknown and the outcome is

$$\phi_\mathrm{e} = \phi_\mathrm{e}^\mathrm{1u} + O(\Delta x). \tag{3.41}$$

If Eqs. (3.38) and (3.41) are compared, it can be seen that, for a reduction of Δx, the face-value estimation seems to approach the 'true' value quicker in the case with the central-differencing scheme. In other words, this means that, if the grid is made denser, i.e. more cells are introduced, the error in the central-differencing scheme will be reduced more quickly than in the case with the first-order upwind scheme. The lowest order of the grid spacing in the central-differencing scheme is 2, hence the central-differencing scheme is referred to as *second-order accurate*. For the first-order upwind scheme, the corresponding number is 1; hence this scheme is referred to as *first-order accurate*.

Higher-order schemes are more accurate but this can be at the expense of numerical stability. When stability is problematic it is recommended that one start with a simple first-order upwind scheme and change to a higher-order scheme after some iterations. Going to higher order is always necessary in order to minimize numerical diffusion when the grids are not aligned with the flow.

3.8.6 The hybrid scheme

An attempt to combine the positive properties of both the central-differencing scheme and the first-order upwind scheme has been proposed. This scheme is called the *hybrid differencing scheme*. This scheme implements the upwind scheme at faces where the criterion in Eq. (3.24) is fulfilled, and uses central differencing elsewhere. Thus, this scheme takes advantage of both the high accuracy of the central-differencing scheme and the more physical properties in terms of boundedness and transportiveness of the upwind scheme.

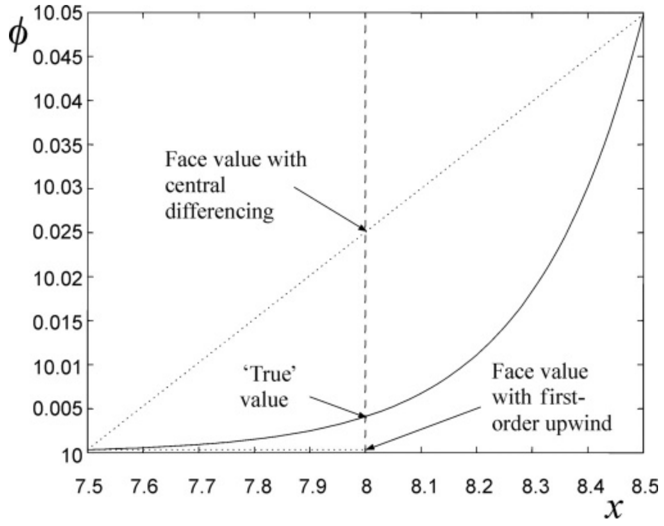

Figure 3.10 Face-value approximations of the face between cells 8 and 9 with central differencing and first-order upwind, respectively. The analytical ('true') solution which, in this specific case, corresponds to the power-law prediction has been added for comparison. $U = 5$ cm s^{-1}.

3.8.7 The power-law scheme

A more accurate scheme is the *power-law scheme*. Briefly, the face value of ϕ is estimated by solving a convection–diffusion equation (cf. Eq. (3.1)),

$$\rho \frac{\mathrm{d}(U\phi)}{\mathrm{d}x} = \frac{\mathrm{d}}{\mathrm{d}x}\left(\Gamma \frac{\mathrm{d}\phi}{\mathrm{d}x}\right). \tag{3.42}$$

Equation (3.42) has the following solution, assuming constant fluid and flow properties:

$$\frac{\phi(x) - \phi_0}{\phi_{\Delta x} - \phi_0} = \frac{\exp(\rho U x / \Gamma) - 1}{\exp(\rho U \, \Delta x / \Gamma) - 1}, \tag{3.43}$$

where the indices 0 and Δx represent two neighbouring cells. If the grid spacing is equidistant, the face is situated at $x = 0.5$. The face value can now be estimated using Eq. (3.43). If the parameters from the previous examples are used, the plot of $\phi(x)$ against x will clearly show that for $U = 5$ cm s^{-1} the face value is almost exactly the same as for the upstream cell; see Figure 3.10. The central-differencing scheme makes a bad estimation of the face value. This was the reason why the upwind scheme was chosen in Example 3.

3.8.8 The QUICK scheme

Numerous successful attempts have been made to create numerical schemes with higher accuracy than second order. One of them will very briefly be discussed, namely QUICK (Quadratic Upstream Interpolation for Convective Kinetics). QUICK combines the strengths of both the upwind schemes and of the central differencing; it uses three-point upstream quadratic interpolation to estimate the face values. For the western and eastern faces, respectively,

$$\phi_w = \frac{6}{8}\phi_W + \frac{3}{8}\phi_P - \frac{1}{8}\phi_{WW}, \tag{3.44}$$

$$\phi_e = \frac{6}{8}\phi_P + \frac{3}{8}\phi_E - \frac{1}{8}\phi_W. \tag{3.45}$$

It has been assumed here that the velocity is positive. A Taylor expansion of ϕ around the eastern face gives

$$\phi_E = \phi_e + (\Delta x/2)\left(\frac{d\phi}{dx}\right)_e + \frac{(\Delta x/2)^2}{2}\left(\frac{d^2\phi}{dx^2}\right)_e + O\left[(\Delta x)^3\right], \tag{3.34′}$$

$$\phi_P = \phi_e - (\Delta x/2)\left(\frac{d\phi}{dx}\right)_e + \frac{(\Delta x/2)^2}{2}\left(\frac{d^2\phi}{dx^2}\right)_e + O\left[(\Delta x)^3\right], \tag{3.35′}$$

$$\phi_W = \phi_e - (1.5\Delta x)\left(\frac{d\phi}{dx}\right)_e + \frac{(1.5\Delta x)^2}{2}\left(\frac{d^2\phi}{dx^2}\right)_e + O\left[(\Delta x)^3\right]. \tag{3.46}$$

On inserting Eqs. (3.34′), (3.35′) and (3.46) into Eq. (3.45), we get

$$\phi_e = \phi_e^{\text{QUICK}} + O\left[(\Delta x)^3\right]. \tag{3.47}$$

Thus, the QUICK scheme is *third-order accurate*. Further, it can be shown that it is *unbounded* but fulfils the *transportiveness* criterion. The use of QUICK is restricted to hexahedral meshes.

3.8.9 More advanced discretization schemes

The discretization schemes mentioned previously are only a few among the many available. In most commercial CFD codes there are many variations of schemes, and it is not our intention to provide a complete list in Table 3.1. However, two more will be mentioned here without going into too much detail.

- MUSCL (Monotone Upstream-centred Schemes for Conservation Laws). The MUSCL scheme shows a similar degree of accuracy to QUICK, but is not limited to hexahedral meshes.
- HRIC (High-Resolution Interface Capturing). The HRIC scheme is primarily used in multiphase flows when tracking the interface between the phases. HRIC has proven to be more accurate than QUICK for VOF simulations (see Chapter 6).

For all of the discretization schemes that have been presented here, the error tends to zero as the grid spacing is reduced infinitely. Schemes that uphold this property are said to be *consistent*. It should also be mentioned that the definitions of some of the discretization schemes presented here seem to differ in the literature.

Table 3.1 Discretization schemes

Discretization scheme	Advantages and disadvantages
Central	Works well when diffusion dominates. Bounded variants are recommended for LES simulations.
First-order upwind	Good when convection dominates and the flow is aligned with the grid. Bounded and robust, and a good scheme to start off the calculation. May introduce numerical diffusion and should be replaced with higher-order schemes for the final calculations.
Second-order upwind	Good for all Péclet numbers, but it is unbounded and not as robust as first-order upwind.
Power law	Good for intermediate values of the Péclet number ($Pe < 10$).
QUICK	Good for all Péclet numbers. Better accuracy than the second-order scheme for rotating or swirling flows. In general the second-order scheme is sufficient. Applicable only to quadrilateral or hexahedral meshes.
MUSCL	Better accuracy than the second-order scheme for rotating or swirling flows. The scheme is used on all types of meshes.
HRIC	Used primarily for interface tracking in VOF simulations. Better accuracy than QUICK.

3.9 Solving the velocity field

In the previous examples and discussions it has been assumed that the velocity field has somehow been determined; see Eq. (3.19) as an illustrative example of this. Example 1 also concluded that the predetermined velocity gave rise to a set of *linear* equations. However, most commonly the user does not know the velocity from the start, and hence has to solve for it as well. Solving the velocity field requires some extra attention; we will examine this now.

The transport equations for momentum (also called Navier–Stokes equations) and the continuity equation are

$$\frac{\partial \rho U_i}{\partial t} + \frac{\partial \rho U_i U_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\mu \frac{\partial U_i}{\partial x_j}\right) - \frac{\partial P}{\partial x_i} \tag{3.48}$$

and

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho U_j}{\partial x_j} = 0. \tag{3.49}$$

These expressions were derived in Chapter 2. As can be seen, the momentum equation is very similar to the transport equations that have been solved in previous examples. There is a small difference, though, that makes the solution procedure much more complicated than in previous cases, and that is the fact that there now exists a source term in the equation. The source term has the formulation of a pressure gradient, which shouldn't be surprising. A pressure gradient in e.g. a tube gives rise to the momentum in the streamwise direction. Summing the number of variables in Eqs. (3.48) and (3.49), there are four variables, namely three velocities and one pressure (remember that we have assumed incompressible flow!). The total number of equations is four (3 + 1), so it should be possible to solve the system straight away. Normally, Eq. (3.48) is used for the three velocity components. Unfortunately, Eq. (3.49) cannot be used to solve for pressure. Hence, the equation must be modified first. Taking the divergence of Eq. (3.48) gives

$$\frac{\partial^2}{\partial x_i\,\partial t}(\rho U_i) + \frac{\partial^2}{\partial x_i\,\partial x_j}\left(\rho U_i U_j\right) = \frac{\partial^2}{\partial x_i\,\partial x_j}\left(\mu\frac{\partial U_i}{\partial x_j}\right) - \frac{\partial^2 P}{\partial x_i\,\partial x_i}. \qquad (3.50)$$

The first term on the left-hand side and the first term on the right-hand side can be rewritten (assuming constant density and viscosity) as

$$\frac{\partial}{\partial x_i}\left(\frac{\partial \rho U_i}{\partial t}\right) = \frac{\partial}{\partial t}\left(\frac{\partial \rho U_i}{\partial x_i}\right),$$

$$\frac{\partial}{\partial x_i}\left[\frac{\partial}{\partial x_j}\left(\mu\frac{\partial U_i}{\partial x_j}\right)\right] = \frac{\partial}{\partial x_j}\left[\frac{\partial}{\partial x_j}\left(\mu\frac{\partial U_i}{\partial x_i}\right)\right].$$

Owing to continuity (see Eq. (3.49)), the right-hand sides of both these expressions are equal to zero. Equation (3.50) can thus be written in the following way:

$$\frac{\partial^2 P}{\partial x_i\,\partial x_i} = -\frac{\partial^2}{\partial x_i\,\partial x_j}\left(\rho U_i U_j\right). \qquad (3.51)$$

Equation (3.51), which is a scalar equation, can be used as a direct equation for pressure. However, numerical problems are commonly encountered, for reasons that will not be discussed in this book. Therefore, in almost all commercial CFD software an iterative procedure for solving Eq. (3.51) is adopted, thereby avoiding the numerical problems. Some of the most widely used algorithms are SIMPLE, SIMPLEC, SIMPLER and PISO. The algorithms of the SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) family use a starting guess for pressure and velocity to solve for the corresponding velocities via the momentum equations. Since the starting guess of the pressure will not be correct, the velocities obtained will not satisfy continuity. Hence, correction factors for pressure and velocity are introduced, and transport equations for these factors are proposed and solved to give corrected velocities and pressure. The other transport equations, e.g. for various scalars, are then solved. Then, a check for convergence is made and the procedure is repeated until convergence is reached. SIMPLER and SIMPLEC are improved variants of SIMPLE. The PISO algorithm is mostly used for unsteady flow. It can be seen as an extension of the SIMPLE algorithm, but it uses two levels of correction instead of one as is the case with SIMPLE.

It is generally not possible to say that a specific scheme is always better than another with respect to efficiency or robustness. These properties are very dependent on the flow conditions at hand, but a proper choice of scheme can sometimes speed up the simulations significantly. The performance and behaviour of the schemes have been thoroughly examined, and a few conclusions are mentioned here.

For a laminar backward-facing step it has been shown that PISO performs twice as fast as SIMPLE, while it has been shown to be four times as slow for a case concerning flow through a heated fin. For steady flow problems, SIMPLER has been shown to need

30%–50% less computer time than SIMPLE to solve problems in general. The degree of coupling between the momentum equations and scalar equations has been shown to have a significant impact sometimes. In problems where there is only a weak coupling or no coupling at all between the momentum and scalar equations, PISO has been shown to have the most robust convergence and is quicker than SIMPLEC and SIMPLER. If the opposite is valid, i.e. there is a strong coupling between the momentum and scalar equations, SIMPLER and SIMPLEC have been shown to perform better than PISO. Further, it has not been possible to claim that either SIMPLER or SIMPLEC is superior to the other in general. SIMPLEC usually performs better in situations in which the rate of convergence is limited by the pressure–velocity coupling, such as non-complex laminar-flow cases.

In order to solve the transport equations for momentum, Eq. (2.21), it is required that the pressures at the cell faces are known. As in the cases with other transported quantities, this is achieved by interpolation of the values in the neighbouring cells. Most commercial CFD software offers a variety of interpolation schemes. Each scheme uses its own interpolation function.

The standard scheme uses the coefficients a_p, a_e, etc. to interpolate the pressure on the cell faces. As long as the pressure variation between the cells is smooth, the scheme usually works fine. For flows with large body forces the standard scheme has been shown to be troublesome. In such cases it is recommended that the body force-weighted scheme be used. This scheme assumes that the normal gradient of the difference between the body force and pressure is constant. For swirling flows and flows with natural convection, the PRESTO! scheme should be used. The PRESTO! scheme uses the discrete continuity equation to calculate the pressure field on a mesh that is geometrically shifted so that the new cell centres are where the faces of the ordinary mesh are placed. This means that the pressures on the faces are now known. The PRESTO! scheme is also recommended for VOF simulations. The second-order pressure scheme is analogous to the second-order discretization scheme presented earlier and is recommended for compressible flows. The second-order scheme can be numerically unstable if it is used at the start of a calculation or on a bad mesh.

3.9.1 Under-relaxation

The momentum equations, Eq. (3.48), contain nonlinear terms, e.g. ρV^2 in the momentum equation for V. To avoid divergence due to nonlinearities, so-called *under-relaxation* is used in the solving procedure. The under-relaxation factor α is defined as

$$\phi_\mathrm{new} = \alpha\phi_\mathrm{solver} + (1-\alpha)\,\phi_\mathrm{old}, \tag{3.52}$$

where ϕ_new is the new value at the cell, ϕ_solver is the value for the solution to the last iteration, and ϕ_old is the previous value at the cell that is used to compute ϕ_solver. Thus, a large value of α means that the new value will be very influenced by the solved value, and vice versa. α lies usually in the range between zero and one. In many cases the under-relaxation factor can be increased during iterations. Each transport equation or pressure equation has its 'own' corresponding under-relaxation factor. Too low a value

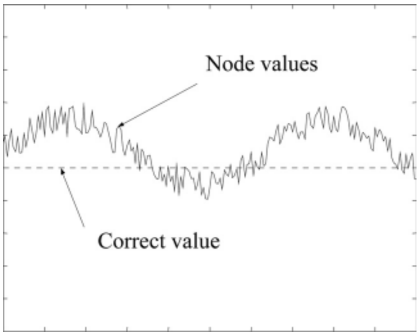

Figure 3.11 High- and low-frequency errors. The correct values correspond to a constant value throughout the domain.

of the under-relaxation factor gives unmotivated long computational times. Too large a value can give a diverged solution. Thus, choosing optimal values is of great practical importance.

Questions

(1) Since the central-differencing scheme uses a weighted average (see Eq. (3.18)) of the neighbouring cells to calculate the present cell, by definition it cannot be unbounded. This is obviously wrong! What restrictions must be put on a_E/a_P and a_W/a_P in order to keep ϕ_P bounded, i.e. between ϕ_E and ϕ_W.

(2) Describe what factors determining what discretization scheme should be used in a specific situation. Can these factors always be determined before the simulation has been performed?

(3) Is a higher-order differencing scheme always better than a scheme with lower order? Consider robustness, accuracy, CPU time and memory demand.

3.10 Multigrid

An efficient way to enhance convergence speed is to use a so-called multigrid solver. The idea is to use at least two levels of grid spacing. Take the previous examples as illustration! Since the Gauss–Seidel solver calculates the cell value only by using the values of the neighbouring cells, it is very efficient in eliminating local errors, see also Figure 3.11. However, there could be situations in which there is an extra need for information to be transported fast through the domain. Let’s say, for instance, that the pressure is increased in an area within the domain. This pressure increase will affect the entire domain instantly. In the CFD simulation the pressure increase will be transported as the Gauss–Seidel algorithm sweeps through the domain. However, if the computational domain contains many cells, it will take a significant time before

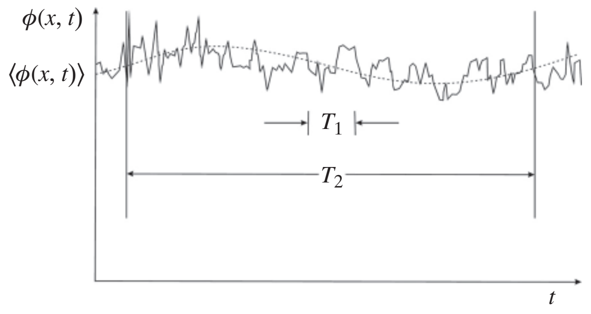

Figure 3.12 The mean and fluctuation of an instantaneous flow variable as a function of time. If $T_1 \ll T_2$, the mean flow and fluctuating quantities are uncorrelated.

information is transported from cells at one end to cells at the other. This will mean that much iteration has to be performed before a converged solution is obtained. Here is where the different grid levels come in. By using a level with larger cells, information can be transported faster than would be possible without this level. Information is transferred from coarser to denser grid levels via so-called *prolongation* and in the other direction by the so-called *restriction process*. Altogether, this means that the rate of convergence is enhanced, often by as much as several orders of magnitude.

3.11 Unsteady flows

Before starting to discuss unsteady flows, we need clarity regarding the definition of the term ‘unsteady’. Defining an unsteady flow as a flow that shows variation in time would imply that almost all flows are unsteady. Remember that most flows of industrial importance are turbulent and that turbulence is always time-dependent! Hence, the definition of unsteadiness must be tuned. In most CFD problems, the flow studied is turbulent, forcing one to solve for mean quantities instead of instantaneous ones and to use an appropriate turbulence model to model the interaction between sub-scale motions and the mean flow (see Chapter 4). In these cases unsteadiness must mean that the mean quantities rather than the instantaneous quantities change with time. Further, it is required that there is a *large* separation in timescales to be able to distinguish between turbulent fluctuations and mean flow fluctuations, see also Figure 3.12. Unfortunately, in most engineering applications concerning unsteady flows this criterion is not completely fulfilled, giving a non-zero correlation between the mean flow and the fluctuating quantity. In such cases, turbulence is moved from the turbulence model to the unsteady term in the time-averaged Navier–Stokes equation.

The procedure for solving unsteady problems follows the same path as for steady problems. First, the general transport equation is integrated, with respect to both space and time,

$$\int_{\text{c.v.}} \left(\int_t^{t+\Delta t} \rho \frac{\partial \phi}{\partial t} \, \mathrm{d}t \right) \mathrm{d}V + \int_t^{t+\Delta t} \left[\int_{\text{c.v.}} \rho \frac{\partial (U_j \phi)}{\partial x_j} \, \mathrm{d}V \right] \mathrm{d}t$$
$$= \int_t^{t+\Delta t} \left[\int_{\text{c.v.}} \frac{\partial}{\partial x_j} \left(\Gamma \frac{\partial \phi}{\partial x_j} \right) \mathrm{d}V \right] \mathrm{d}t + \int_t^{t+\Delta t} \left(\int_{\text{c.v.}} S_\phi \, \mathrm{d}V \right) \mathrm{d}t. \tag{3.53}$$

The order of integration is reversed in the accumulation term. To clarify the steps needed, an example is given.

3.11.1 Example 4 – time-dependent simulation

Using the same experimental set-up as in Example 1, but this time without any convection, the unsteady case is solved. Initially there is no species A in the domain. The boundary conditions are the same as in the original example.

The integrated transport equation for ϕ in the present example is

$$\int_{\text{w}}^{\text{e}} \left(\int_t^{t+\Delta t} \rho \frac{\mathrm{d}\phi}{\mathrm{d}t} \, \mathrm{d}t \right) \mathrm{d}x = \int_t^{t+\Delta t} \left[\int_{\text{w}}^{\text{e}} \frac{\mathrm{d}}{\mathrm{d}x} \left(\Gamma \frac{\mathrm{d}\phi}{\mathrm{d}x} \right) \mathrm{d}x \right] \mathrm{d}t. \tag{3.54}$$

Using central differencing allows us to rewrite Eq. (3.54) as

$$\int_{\text{w}}^{\text{e}} \left(\int_t^{t+\Delta t} \rho \frac{\mathrm{d}\phi}{\mathrm{d}t} \, \mathrm{d}t \right) \mathrm{d}x = \int_t^{t+\Delta t} \left(\Gamma_{\text{e}} \frac{\phi_{\text{E}} - \phi_{\text{P}}}{\Delta x} - \Gamma_{\text{w}} \frac{\phi_{\text{P}} - \phi_{\text{W}}}{\Delta x} \right) \mathrm{d}t. \tag{3.55}$$

To proceed with the time integration an appropriate temporal discretization scheme is needed. As in the case with spatial discretization there is a choice of scheme; either ‘earlier’ or ‘later’ times (or a mixture of them) can be used to estimate the temporal derivative at the ‘present’ time. For now,

$$\rho[\phi_{\text{P}}(t + \Delta t) - \phi_{\text{P}}(t)]\Delta x = \int_t^{t+\Delta t} \left(\Gamma_{\text{e}} \frac{\phi_{\text{E}} - \phi_{\text{P}}}{\Delta x} - \Gamma_{\text{w}} \frac{\phi_{\text{P}} - \phi_{\text{W}}}{\Delta x} \right) \mathrm{d}t. \tag{3.56}$$

To continue, ϕ on the right-hand side of Eq. (3.56) is evaluated at each time step. ϕ changes in time, so there are at least three different ways to get values for ϕ: (1) let ϕ be equal to $\phi(t)$, (2) let ϕ be equal to $\phi(t + \Delta t)$, and (3) let ϕ be a mixture of the two.

We now define the weight factor θ:

$$\int_t^{t+\Delta t} \phi \, \mathrm{d}t = [\theta\phi(t + \Delta t) + (1 - \theta)\phi(t)]\Delta t. \tag{3.57}$$

The case $\theta = 0$ means that only the old value of ϕ is used when evaluating Eq. (3.56). This leads to so-called *explicit discretization* of this equation. The other extreme, $\theta = 1$, corresponds to a discretization where only the new value of ϕ is used in the discretization, the so-called *(fully) implicit discretization*. Of course, θ can take values also between zero and one; e.g. the *Crank–Nicolson scheme* for $\theta = 0.5$.

Using Eq. (3.57) to express ϕ_i in Eq. (3.56),

$$\rho\left[\phi_{\mathrm{P}} - \phi_{\mathrm{P}}^{\mathrm{o}}\right]\Delta x = \left[\theta\left(\Gamma_{\mathrm{e}}\frac{\phi_{\mathrm{E}} - \phi_{\mathrm{P}}}{\Delta x} - \Gamma_{\mathrm{w}}\frac{\phi_{\mathrm{P}} - \phi_{\mathrm{W}}}{\Delta x}\right)\right. \\ \left. + (1-\theta)\left(\Gamma_{\mathrm{e}}\frac{\phi_{\mathrm{E}}^{\mathrm{o}} - \phi_{\mathrm{P}}^{\mathrm{o}}}{\Delta x} - \Gamma_{\mathrm{w}}\frac{\phi_{\mathrm{P}}^{\mathrm{o}} - \phi_{\mathrm{W}}^{\mathrm{o}}}{\Delta x}\right)\right]\Delta t. \tag{3.58}$$

Here, ϕ_i^{o} means the value for cell i at time t. ϕ_i means the value for cell i at time $t + \Delta t$.

To examine the properties of Eq. (3.58), it is rewritten as

$$\left[\rho\frac{\Delta x}{\Delta t} + \theta\left(\frac{\Gamma_{\mathrm{e}}}{\Delta x} + \frac{\Gamma_{\mathrm{w}}}{\Delta x}\right)\right]\phi_{\mathrm{P}} \\ = \frac{\Gamma_{\mathrm{e}}}{\Delta x}\left[\theta\phi_{\mathrm{E}} + (1-\theta)\phi_{\mathrm{E}}^{\mathrm{o}}\right] + \frac{\Gamma_{\mathrm{w}}}{\Delta x}\left[\theta\phi_{\mathrm{W}} + (1-\theta)\phi_{\mathrm{W}}^{\mathrm{o}}\right] \\ + \left[\rho\frac{\Delta x}{\Delta t} - (1-\theta)\frac{\Gamma_{\mathrm{e}}}{\Delta x} - (1-\theta)\frac{\Gamma_{\mathrm{w}}}{\Delta x}\right]\phi_{\mathrm{P}}^{\mathrm{o}}. \tag{3.59}$$

This leads to (cf. Eq. (3.19))

$$a_{\mathrm{P}}\phi_{\mathrm{P}} = a_{\mathrm{W}}\left[\theta\phi_{\mathrm{W}} + (1-\theta)\,\phi_{\mathrm{W}}^{\mathrm{o}}\right] + a_{\mathrm{E}}\left[\theta\phi_{\mathrm{E}} + (1-\theta)\,\phi_{\mathrm{E}}^{\mathrm{o}}\right] \\ + \left[a_{\mathrm{P}}^{o} - (1-\theta)\,a_{\mathrm{W}} - (1-\theta)\,a_{\mathrm{E}}\right]\phi_{\mathrm{P}}^{\mathrm{o}}, \tag{3.60}$$

where

$$\begin{aligned} a_{\mathrm{P}} &= \theta(a_{\mathrm{W}} + a_{\mathrm{E}}) + a_{\mathrm{P}}^{\mathrm{o}}, \\ a_{\mathrm{P}}^{\mathrm{o}} &= \rho\frac{\Delta x}{\Delta t}, \\ a_{\mathrm{W}} &= \frac{\Gamma_{\mathrm{w}}}{\Delta x}, \\ a_{\mathrm{E}} &= \frac{\Gamma_{\mathrm{e}}}{\Delta x}. \end{aligned} \tag{3.61}$$

An unsteady problem must be solved in a different way from a steady problem. This is as a consequence of having to solve the set of equations for many different times. Evidently, Eq. (3.60) must be satisfied within each time step, thus it is necessary to sub-iterate within each time step. When a convergent solution has been obtained, move forward one time step and repeat the sub-iterations. This is continued until the appropriate time has been reached. The starting guess for the next time step is the solution to the previous time step. In that way, the number of sub-iterations required within each time step usually decreases with time, provided that the time step is sufficiently small. The presence of a starting solution instead of a starting guess is required for the initial time step.

The example is finally solved using the different discretization methods.

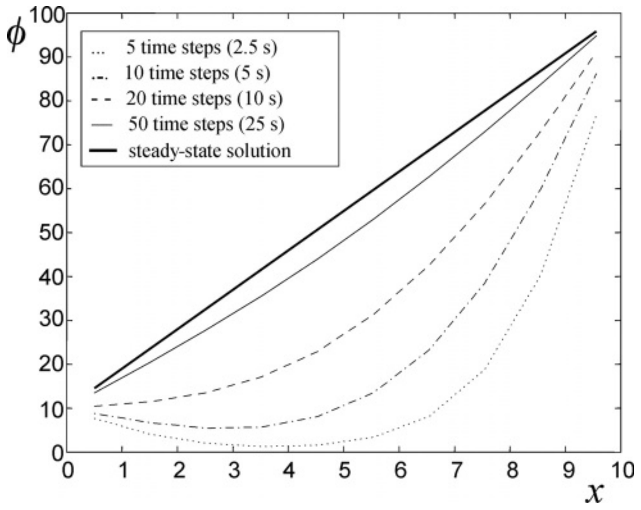

Figure 3.13 A plot of molar concentration as a function of the distance at different times. The time step is 0.4 s and the discretization method is explicit. The initial condition was $c = 0$ everywhere.

Explicit method

With $\theta = 0$, Eq. (3.60) gives

$$\phi_P = \frac{a_W\phi_W^o + a_E\phi_E^o + \left(a_P^o - a_W - a_E\right)\phi_P^o}{a_P} \tag{3.62}$$

and the coefficients

$$\begin{aligned} a_P &= a_P^o, \\ a_P^o &= \rho\frac{\Delta x}{\Delta t}, \\ a_W &= \frac{\Gamma_w}{\Delta x}, \\ a_E &= \frac{\Gamma_e}{\Delta x}. \end{aligned} \tag{3.63}$$

The next question is what time step to choose. A time step is chosen in order to be able to evaluate ϕ_P. As earlier, the scheme needs to be bounded, and therefore the time step should be chosen so that $a_P^o - a_W - a_E > 0$. This implies

$$\Delta t < \frac{\rho(\Delta x)^2}{\Gamma_e + \Gamma_w}. \tag{3.64}$$

In the example, this means that the time step cannot be larger than 0.5 seconds. Solving Eq. (3.62) with the coefficients from Eq. (3.63) using Δt from Eq. (3.64) gives a plot of the molar concentration at different times, see Figure 3.13.

The results look reasonable. Initially there is no species A in the system, and the more time elapses, the more of the species has diffused from the walls. The explicit method will be discussed more later on.

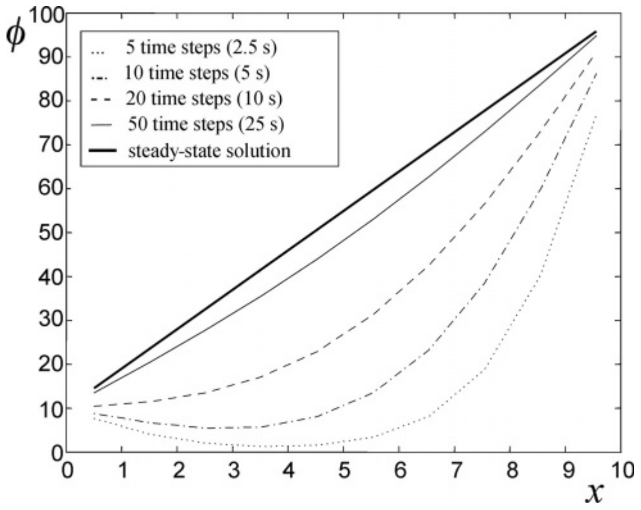

Figure 3.14 A plot of molar concentration as a function of the distance at different times. The time step is 0.5 s and the discretization method is fully implicit. The initial condition was $c = 0$ everywhere.

An advantage with the explicit method is that there is no need for sub-iterations. Looking at Eq. (3.62) explains why; all values are taken from 'old' time steps. Thus, its name *explicit*. The method is also called *global time stepping*.

Fully implicit method

As defined before, in the fully implicit method the new values of ϕ are used as an estimate of ϕ during the whole step of integration, i.e. $\theta = 1$ in this method. Equation (3.60) is then reduced to

$$a_P\phi_P = a_W\phi_W + a_E\phi_E + a_P^o\phi_P^o \tag{3.65}$$

with the following coefficients:

$$\begin{aligned} a_P &= a_W + a_E + a_P^o, \\ a_P^o &= \rho\frac{\Delta x}{\Delta t}, \\ a_W &= \frac{\Gamma_w}{\Delta x}, \\ a_E &= \frac{\Gamma_e}{\Delta x}. \end{aligned} \tag{3.66}$$

An obvious advantage with the fully implicit method is that the coefficients in Eq. (3.66) are always positive, giving unconditional boundedness. Thus, there is no strict upper limit in the choice of the appropriate time step. However, there is usually an optimal number of iterations to be used in each time step, which will restrict its value. Solving Eq. (3.65) with the coefficients in Eq. (3.66) with the time step $\Delta t = 0.5$ s gives a similar plot to that in the explicit case, see Figure 3.14. As in the explicit case,

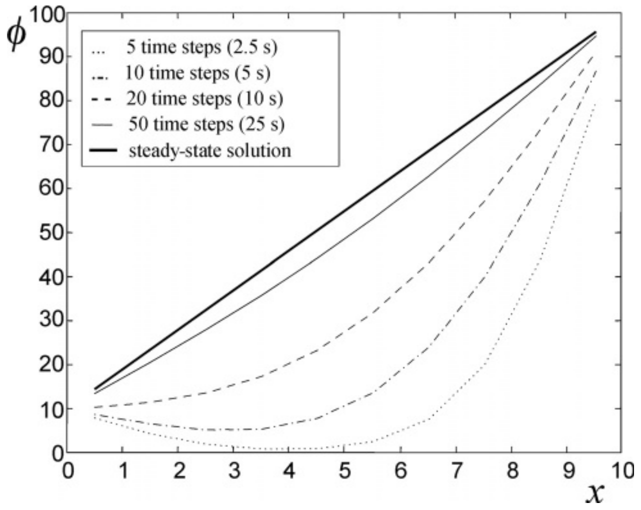

Figure 3.15 A plot of molar concentration as a function of the distance at different times. The time step is 0.5 s and the discretization method is Crank–Nicolson. The initial condition was $c = 0$ everywhere.

the results are reasonable. Diffusion takes care of the transport of species A into the domain.

The Crank–Nicolson method

In the Crank–Nicolson method, both the old value and the new value of ϕ are used to estimate the integral value of ϕ over a time step $\theta = 0.5$. Equation (3.60) then becomes

$$a_{\mathrm{P}}\phi_{\mathrm{P}} = a_{\mathrm{W}}\left[\frac{1}{2}\phi_{\mathrm{W}} + \frac{1}{2}\phi_{\mathrm{W}}^{\mathrm{o}}\right] + a_{\mathrm{E}}\left[\frac{1}{2}\phi_{\mathrm{E}} + \frac{1}{2}\phi_{\mathrm{E}}^{\mathrm{o}}\right] + \left[a_{\mathrm{P}}^{\mathrm{o}} - \frac{1}{2}a_{\mathrm{W}} - \frac{1}{2}a_{\mathrm{E}}\right]\phi_{\mathrm{P}}^{\mathrm{o}} \tag{3.67}$$

with the coefficients

$$\begin{aligned} a_{\mathrm{P}} &= \frac{1}{2}(a_{\mathrm{W}} + a_{\mathrm{E}}) + a_{\mathrm{P}}^{\mathrm{o}}, \\ a_{\mathrm{P}}^{\mathrm{o}} &= \rho\frac{\Delta x}{\Delta t}, \\ a_{\mathrm{W}} &= \frac{\Gamma_{\mathrm{w}}}{\Delta x}, \\ a_{\mathrm{E}} &= \frac{\Gamma_{\mathrm{e}}}{\Delta x}. \end{aligned} \tag{3.68}$$

Solving Eq. (3.67) together with its coefficients in Eq. (3.68) yields with $\Delta t = 0.5$ s the results shown in Figure 3.15.

Just as in the previous cases, the results look well shaped. Before moving on, we note that

$$\Delta t < \frac{\rho(\Delta x)^2}{\Gamma} \tag{3.69}$$

must be fulfilled in order to avoid unbounded solutions.

3.11.2 Conclusions on the different time discretization methods

As when the different discretization methods for spatial discretization were presented and evaluated in earlier sections of this chapter, accuracy, boundedness and transportiveness will be discussed here. The explicit method as well as the fully implicit method are first-order accurate, which we state here, without proof,

$$\phi = \phi^{\text{ex}} + O(\Delta t)\,,$$
$$\phi = \phi^{\text{im}} + O(\Delta t)\,.$$

This follows from the fact that the value of ϕ in each time step is taken from *either* the old value *or* the new one. The Crank–Nicolson method is second-order accurate, or in mathematical terms,

$$\phi = \phi^{\text{C-N}} + O\left[(\Delta t)^2\right]\,.$$

The Crank–Nicolson method has major resemblances to the central-differencing scheme for spatial discretization, cf. Eqs. (3.13) and (3.57) with $\theta = 0.5$. A reduction of the time-step size thus gives a more significant improvement in the results if the Crank–Nicolson method is used than for first-order methods. There also exist many higher-order numerical schemes for time discretization. Figure 3.15 shows a simulation using the Crank–Nicolson method.

As concluded earlier, the explicit method is only conditionally bounded. This imposes a constraint on the choice of the time step, cf. Eq. (3.64). If this criterion is violated, the solution often diverges. On the other hand, being without the need to sub-iterate, the explicit method is *faster* than the other two.

The fully implicit method is unconditionally bounded. This is a great advantage with the method, and, in most commercial CFD codes, the fully implicit method is the default method for time discretization.

The Crank–Nicolson method is only conditionally bounded, cf. Eq. (3.69). This constraint is, however, not as restrictive as in the explicit method; it differs by a factor of 2.

The time step is determined by the Courant number (CFL). Since the time step is determined by the fluxes in each cell and the general rule is that the time step should be shorter than the time it takes to transport past the cell,

$$\Delta t < \text{CFL}\ \min\left(\frac{\rho(\Delta x)^2}{\Gamma}, \frac{\Delta x}{U}\right), \tag{3.70}$$

where CFL = 1 for an explicit solver and CFL = 5 or higher for a fully implicit solver. A fully implicit solver should be stable for all time steps, but, due to nonlinearities in the equations, it is recommended that one start with a low Courant number (5) and increase it during the iterations.

Among the most recent transient solvers, the use of the *method of lines* is very common. This method reformulates the transient PDE into a transient ordinary differential equation

(ODE) by discretizing the spatial coordinates only. Then an ordinary ODE solver is used to solve the equations.

Questions

(1) The first term in Eq. (3.53) can be seen as a convective term for time. Why is there no diffusion term for time? What would such a term look like?
(2) What does Figure 3.12 tell us? What consequences does it enforce on the dependent variables?

3.12 Meshing

Finally, a few words will be said about meshing. So far, it has been assumed in all the examples that a proper mesh has been provided. However, this is generally not the case. Thus, *meshing* becomes an important part of the CFD engineer's work. Bad meshes often give numerical problems and bad results.

3.12.1 Mesh generation

There are several commercial software packets for mesh generation. To generate a mesh is in general a very complex procedure, and only some of the basics will be mentioned here.

Traditionally, grids have been divided into *structured grids* and *unstructured grids*. The structured grids are built up from quadrilateral elements, i.e. four-edged elements, but not necessarily rectangles, in 2D and hexahedra (elements with six faces) in 3D. Indexing and finding neighbouring cells is very easy with these elements. CFD programs using structured grids are usually faster and require less memory than programs using unstructured grids. Unstructured grids are built from different elements, quadrilateral and triangular elements in 2D and tetrahedra, hexahedra, pyramids, prisms and even dodecahedra in 3D. Usually, a structured grid will require less memory and have better numerical properties. However, it is not always possible to mesh complex geometries using structured mesh and today most solvers can handle both structured and unstructured grids. Figure 3.16 shows typical building elements for meshing in 2D and 3D.

Most meshing programs allow the user to draw the geometry in the program itself, but most commonly the geometry is imported from a general CAD program. The geometry is built from single points, lines or 2D and 3D shapes, e.g. rectangles, circles, boxes or cylinders. The points can be combined to make lines, and lines to 2D shapes etc., so any geometrical shape can be formed. It is then possible to unite, subtract or intersect these shapes and form new shapes.

Elements at the walls must be handled carefully. The angle between the wall and the grid line should be close to 90°. The easiest way to obtain this is to start the meshing by forming a regular mesh that is built from the surface. When a fine resolution is needed

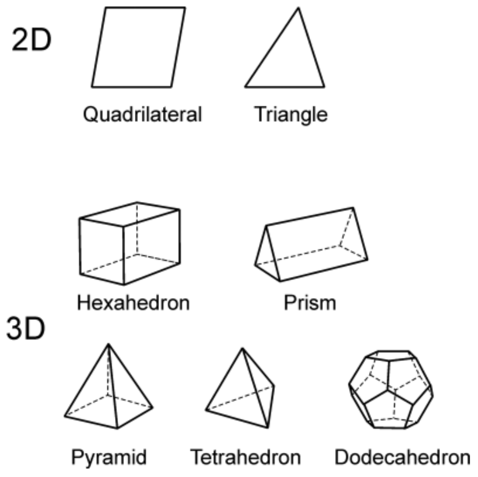

Figure 3.16 Building elements for meshing in 2D and 3D.

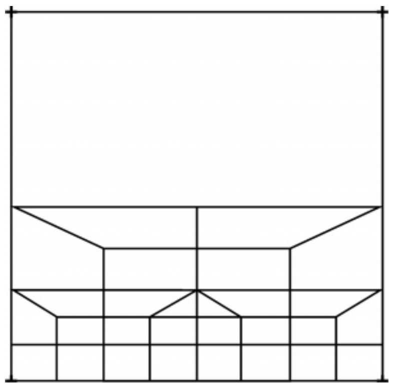

Figure 3.17 A boundary-layer mesh.

on the boundary, a mesh that increases in size with distance from the wall can be formed as shown in Figure 3.17.

There are many other properties of the mesh that should be taken into account in order to produce an accurate solution. For example, it can be shown that, if adjacent cells are very different in size (or volume), the numerical error will increase. It can also be shown that a high skewness of the cells can lead to instabilities and lower accuracy. The optimal situation is to have cells that have 90° corners and edges of equal length. For example, optimal quadrilateral meshes will have vertex angles close to 90°, while triangular meshes should preferably have angles close to 60° and have all angles less than 90°. In tetrahedral meshes the angles should be kept between 40° and 140°. One way to decrease the number of cells and enhance the convergence rate is to stretch the cells along a coordinate axis. This will increase their aspect ratio, but this is usually acceptable as long as the aspect ratio is kept below 5. For example, in long thin channels,

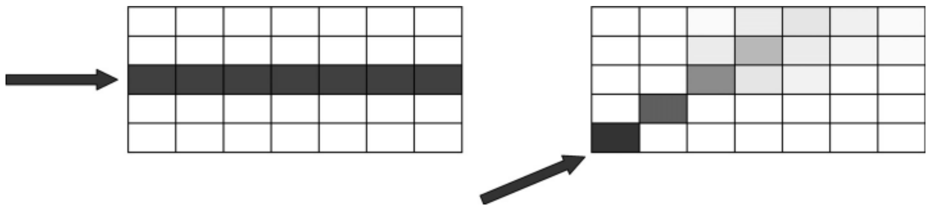

Figure 3.18 The effect of numerical diffusion. In the cases shown, the grid is aligned with the flow (left), and has inclined alignment (right).

it could be a good idea to use long cells in the streamwise direction since there are hardly any gradients in that direction.

3.12.2 Adaptation

It has already been concluded that the distance between adjacent cells, namely the grid spacing, Δx, plays an important role in determining the accuracy of the solution. The denser the mesh, the more accurate the solution. If consistency is upheld, the solution to the discretized problem tends to the exact solution of the set of PDEs as the grid spacing is reduced infinitely. It has also been shown that the required grid spacing is related to the order of the discretization scheme.

In many cases, a good idea could be to use different grid spacings in different regions of the grid. In this way, it is possible to resolve some areas to a very large extent, while other areas are not resolved any more than is necessary in order to avoid divergence. Boundary-layer flows, for example, generally require a very dense mesh close to the wall, while the flow far from the wall does not have to be resolved in detail. This is a consequence of the fact that areas with large gradients normally contain larger errors and thus have to be better resolved. In order to know where the largest gradients are, a simulation has to be performed or the user has to trust his or her intuition and previous experience. Then grid refinement/coarsening is done in the appropriate areas. A new simulation is then performed. If necessary, further refinement/coarsening can be done. Most CFD software has a built-in grid adaptation/coarsening algorithm.

3.12.3 Numerical diffusion

Care must be taken when choosing the directions of the cell axes. In a structured grid it is common to dispose the cells in a manner parallel to the flow. To illustrate what can happen if the cells are disposed in an inclined manner, consider the following (see also Figure 3.18).

A liquid stream with a non-diffusing species is flowing with a constant velocity. If the cells are parallel to the flow, there will be no transfer of the species in the wall-normal direction. This would of course be the case in reality as well. However, if the cells are not parallel to the flow, there will be a transport of species due to the fact that the value of the species entity is the same throughout the cell. Hence, there will be transport due to the discretization. This is called *numerical*, or *false*, *diffusion*. By refining the mesh it is possible to reduce the effect of numerical diffusion. The use of higher-order discretization schemes will also reduce this effect.

3.13 Summary

The basics of the numerics behind a CFD simulation have now been presented. It is important to keep in mind that this has only been a short introduction. At this stage the reader should be able to understand the fundamentals of commercial CFD software. The reader should also be able to understand some of the problems that can occur and how they can be avoided.

Questions

(1) Explain what is meant by adaptation.
(2) Describe the principle of multigrid methods.
(3) What is meant by the Courant number?
(4) Describe the difference between implicit and explicit methods.
(5) What is numerical diffusion and how can it be minimized?