2

Direct numerical simulations of finite Reynolds number flows

In this chapter and the following three, we discuss numerical methods that have been used for direct numerical simulations of multiphase flow. Although direct numerical simulations, or DNS, mean slightly different things to different people, we shall use the term to refer to computations of complex unsteady flows where all continuum length and time scales are fully resolved. Thus, there are no modeling issues beyond the continuum hypothesis. The flow within each phase and the interactions between different phases at the interface between them are found by solving the governing conservation equations, using grids that are finer and time steps that are shorter than any physical length and time scale in the problem.

The detailed flow field produced by direct numerical simulations allows us to explore the mechanisms governing multiphase flows and to extract information not available in any other way. For a single bubble, drop, or particle, we can obtain integrated quantities such as lift and drag and explore how they are affected by free stream turbulence, the presence of walls, and the unsteadiness of the flow. In these situations it is possible to take advantage of the relatively simple geometry to obtain extremely accurate solutions over a wide range of operating conditions. The interactions of a few bubbles, drops, or particles is a more challenging computation, but can be carried out using relatively modest computational resources. Such simulations yield information about, for example, how bubbles collide or whether a pair of buoyant particles, rising freely through a quiescent liquid, orient themselves in a preferred way. Computations of one particle can be used to obtain information pertinent to modeling of dilute multiphase flows, and studies of a few particles allow us to assess the importance of rare collisions. It is, however, the possibility of conducting DNS of thousands of freely interacting particles that holds the greatest promise. Such simulations can yield data for not only the collective lift and drag of dense systems, but also about how the particles are distributed and what impact the formation of structures and clusters has on the overall behavior of the flow. Most industrial size systems, such as fluidized bed reactors or bubble columns, will remain out of reach of direct numerical simulations for the foreseeable future (and even if they were possible, DNS is unlikely to be used for routine design). However, the size of systems that *can* be studied is growing rapidly. It is realistic today to conduct DNS of fully three-dimensional systems resolved by several hundred grid points in each spatial direction. If we assume that a single bubble can be adequately resolved by 25 grid points (sufficient for clean bubbles at relatively modest Reynolds numbers), that the bubbles are, on the average, one bubble diameter apart (a void fraction of slightly over 6%), and that we have a uniform grid with 1000^3 grid points, then we would be able to accommodate 8000 bubbles. High Reynolds numbers and solid particles or drops generally require higher resolution. Furthermore, the number of bubbles that we can simulate on a given grid obviously depends strongly on the void fraction. It is clear, however, that DNS has opened up completely new possibilities in the studies of multiphase flows which we have only started to explore.

In addition to relying on explosive growth in available computer power, progress in DNS of multiphase flows has also been made possible by the development of numerical methods. Advecting the phase boundary poses unique challenges and we will give a brief overview of such methods below, followed by a more detailed description in the next few chapters. In most cases, however, it is also necessary to solve the governing equations for the fluid flow. For body-fitted and unstructured grids, these are exactly the same as for flows without moving interfaces. For the “one-fluid” approach introduced in Chapter 3, we need to deal with density and viscosity fields that change abruptly across the interface and singular forces at the interface, but otherwise the computations are the same as for single-phase flow. Methods developed for single-phase flows can therefore generally be used to solve the fluid equations. After we briefly review the different ways of computing multiphase flows, we will therefore outline in this chapter a relatively simple method to compute single-phase flows using a regular structured grid.

2.1 Overview

Many methods have been developed for direct numerical simulations of multiphase flows. The oldest approach is to use one stationary, structured grid for the whole computational domain and to identify the different fluids by markers or a marker function. The equations expressing conservation of mass, momentum and energy hold, of course, for any fluid, even when density and viscosity change abruptly and the main challenge in this approach is to accurately advect the phase boundary and to compute terms concentrated at the interface, such as surface tension. In the marker-and-cell (MAC) method of Harlow and collaborators at Los Alamos (Harlow and Welch, 1965) each fluid is represented by marker points distributed over the region that it occupies. Although the MAC method was used to produce some spectacular results, the distributed marker particles were not particularly good at representing fluid interfaces. The Los Alamos group thus replaced the markers by a marker function that is a constant in each fluid and is advected by a scheme specifically written for a function that changes abruptly from one cell to the next. In one dimension this is particularly straightforward and one simply has to ensure that each cell fills completely before the marker function is advected into the next cell. Extended to two and three dimensions, this approach results in the volume-of-fluid (VOF) method.

The use of a single structured grid leads to relatively simple as well as efficient methods, but early difficulties experienced with the volume-of-fluid method have given rise to several other methods to advect a marker function. These include the level-set method, originally introduced by Osher and Sethian (1988) but first used for multiphase flow simulations by Sussman, Smereka, and Osher (1994), the CIP method of Yabe and collaborators (Takewaki, Nishiguchi and Yabe, 1985; Takewaki and Yabe, 1987), and the phase field method used by Jacqmin (1997). Instead of advecting a marker function and inferring the location of the interface from its gradient, it is also possible to mark the interface using points moving with the flow and reconstruct a marker function from the interface location. Surface markers have been used extensively for boundary integral methods for potential flows and Stokes flows, but their first use in Navier–Stokes computations was by Daly (1969a,b) who used them to calculate surface tension effects with the MAC method. The use of marker points was further advanced by the introduction of the immersed boundary method by Peskin (1977), who used connected marker points to follow the motion of elastic boundaries immersed in homogeneous fluids, and by Unverdi and Tryggvason (1992) who used connected marker points to advect the boundary between two different fluids and to compute surface tension from the geometry of the interface. Methods based on using a single structured grid, identifying the interface either by a marker function or connected marker points, are discussed in some detail in Chapter 3 of this book.

The attraction of methods based on the use the "one-fluid" formulation on stationary grids is their simplicity and efficiency. Since the interface is, however, represented on the grid as a rapid change in the material properties, their formal accuracy is generally limited to first order. Furthermore, the difficulty that the early implementations of the "one-fluid" approach experienced, inspired several attempts to develop methods where the grid lines were aligned with the interface. These attempts fall, loosely, into three categories. Body-fitted grids, where a structured grid is deformed in such a way that the interface always coincides with a grid line; unstructured grids where the fluid is resolved by elements or control volumes that move with it in such a way that the interface coincides with the edge of an element; and what has most recently become known as sharp interface methods, where a regular structured grid is used but something special is done at the interface to allow it to stay sharp.

Body-fitted grids that conform to the phase boundaries greatly simplify the specification of the interaction of the phases across the interface. Furthermore, numerical methods on curvilinear grids are well developed and a high level of accuracy can be maintained both within the different phases and along their interfaces. Such grids were introduced by Hirt, Cook, and Butler (1970) for free surface flows, but their use by Ryskin and Leal (1983, 1984) to find the steady state shape of axisymmetric buoyant bubbles brought their utility to the attention of the wider fluid dynamics community. Although body-fitted curvilinear grids hold enormous potential for obtaining accurate solutions for relatively simple systems such as one or two spherical particles, generally their use is prohibitively complex as the number of particles increases. These methods are briefly discussed in Chapter 4. Unstructured grids, consisting usually of triangular (in two-dimensions) and tetrahedral (in three-dimensions) shaped elements offer extreme flexibility, both because it is possible to align grid lines to complex boundaries and also because it is possible to use different resolution in different parts of the computational domain. Early applications include simulations of the breakup of drops by Fritts, Fyre, and Oran (1983) but more recently unstructured moving grids have been used for simulations of multiphase particulate systems, as discussed in Chapter 5. Since body-fitted grids are usually limited to relatively simple geometries and methods based on unstructured grids are complex to implement and computationally expensive, several authors have sought to combine the advantages of the single-fluid approach and methods based on a more accurate representation of the interface. This approach was pioneered by Glimm and collaborators many years ago (Glimm, 1982; Glimm and McBryan, 1985) but has recently re-emerged in methods that can be referred to collectively as "sharp interface" methods. In these methods the fluid domain is resolved by a structured grid, but the interface treatment is improved by, for example, introducing special difference formulas that incorporate the jump across the interface (Lee and LeVeque, 2003), using "ghost points" across the interface (Fedkiw *et al.*, 1999), or restructuring the control volumes next to the interface so that the face of the control volume is aligned with the interface (Udaykumar *et al.*, 1997). While promising, for the most part these methods have yet to prove that they introduce fundamentally new capabilities and that the extra complication justifies the increased accuracy. We will briefly discuss "sharp interface methods" for simulations of the motion of fluid interfaces in Chapter 3 and in slightly more detail for fluid–solid interactions in Chapter 4.

2.2 Integrating the Navier–Stokes equations in time

For a large class of multiphase flow problems, including most of the systems discussed in this book, the flow speeds are relatively low and it is appropriate to treat the flow as incompressible. The unique role played by the pressure for incompressible flows, where it is not a thermodynamic variable, but takes on whatever value is needed to enforce a divergence-free velocity field, requires us to pay careful attention to the order in which the equations are solved. There is, in particular, no explicit equation for the pressure and therefore such an equation has to be found as a part of the solution process. The standard way to integrate the Navier–Stokes equations is by the so called "projection method," introduced by Chorin (1968) and Yanenko (1971). In this approach, the velocity is first advanced without accounting for the pressure, resulting in a field that is in general not divergence-free. The pressure necessary to make the velocity field divergence-free is then found and the velocity field corrected by adding the pressure gradient.

We shall first work out the details for a simple first-order explicit time integration scheme and then see how it can be modified to generate a higher order scheme. To integrate equations (1.2) and (1.7) (or 1.8) in time, we write

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}+\mathbf{A}_{\mathrm{h}}(\mathbf{u}^{n})=-\frac{1}{\rho}\nabla_{\mathrm{h}}p+\nu\mathbf{D}_{\mathrm{h}}(\mathbf{u}^{n})+\mathbf{f}_{\mathrm{b}}^{n} \tag{2.1}$$

$$\nabla_{\mathrm{h}}\cdot\mathbf{u}^{n+1}=0. \tag{2.2}$$

The superscript n denotes the variable at the beginning of a time step of length Δt and $n+1$ denotes the new value at the end of the step. $\mathbf{A}_{\mathrm{h}}$ is a numerical approximation to the advection term, $\mathbf{D}_{\mathrm{h}}$ is a numerical approximation to the diffusion term, and $\mathbf{f}_{\mathrm{b}}$ is a numerical approximation to any other force acting on the fluid. ∇_{h} means a numerical approximation to the divergence or the gradient operator.

In the projection method the momentum equation is split into two parts by introducing a temporary velocity $\mathbf{u}^*$ such that $\mathbf{u}^{n+1} - \mathbf{u}^n = \mathbf{u}^{n+1} - \mathbf{u}^* + \mathbf{u}^* - \mathbf{u}^n$. The first part is a predictor step, where the temporary velocity field is found by ignoring the effect of the pressure:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\mathbf{A}_{\mathrm{h}}(\mathbf{u}^n) + \nu \mathbf{D}_{\mathrm{h}}(\mathbf{u}^n) + \mathbf{f}_{\mathrm{b}}^n. \tag{2.3}$$

In the second step – the projection step – the pressure gradient is added to yield the final velocity at the new time step:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\rho} \nabla_{\mathrm{h}} p^{n+1}. \tag{2.4}$$

Adding the two equations yields exactly equation (2.1).

To find the pressure, we use equation (2.2) to eliminate $\mathbf{u}^{n+1}$ from equation (2.4), resulting in Poisson's equation:

$$\frac{1}{\rho} \nabla_{\mathrm{h}}^2 p^{n+1} = \frac{1}{\Delta t} \nabla_{\mathrm{h}} \cdot \mathbf{u}^* \tag{2.5}$$

since the density ρ is constant. Once the pressure has been found, equation (2.4) is used to find the projected velocity at time step $n+1$. We note that we do not assume that $\nabla_{\mathrm{h}} \cdot \mathbf{u}^n = 0$. Usually, the velocity field at time step n is not exactly divergence-free but we strive to make the divergence of the new velocity field, at $n+1$, zero.

As the algorithm described above is completely explicit, it is subject to relatively stringent time-step limitations. If we use standard centered second-order approximations for the spatial derivatives, as done below, stability analysis considering only the viscous terms requires the step size Δt to be bounded by

$$\Delta t < C_\nu \frac{h^2}{\nu} \tag{2.6}$$

where $C_\nu = 1/4$ and $1/6$ for two- and three-dimensional flows, respectively, and h is the grid spacing. The advection scheme is unstable by itself, but it is stabilized by viscosity if the step size is limited by

$$\Delta t < \frac{2\nu}{q^2}, \tag{2.7}$$

where $q^2 = \mathbf{u} \cdot \mathbf{u}$. More sophisticated methods for the advection terms, which are stable in the absence of viscosity and can therefore also be used to integrate the Euler equations in time, are generally subject to the Courant–Friedrichs–Lewy (CFL) condition[1]. For one-dimensional flow,

$$\Delta t < \frac{h}{(|u|)}. \tag{2.8}$$

Many advection schemes are implemented by splitting, where the flow is sequentially advected in each coordinate direction. In these cases the one-dimensional CFL condition applies separately to each step. For fully multidimensional schemes, however, the stability analysis results in further reduction of the size of the time step. General discussions of the stability of different schemes and the resulting maximum time step can be found in standard textbooks, such as Hirsch (1988), Wesseling (2001), or Ferziger and Perić (2002). In an unsteady flow, the CFL condition on the time step is usually not very severe, since accuracy requires the time step to be sufficiently small to resolve all relevant time scales. The limitation due to the viscous diffusion, equation (2.6), can be more stringent, particularly for slow flow, and the viscous terms are frequently treated implicitly, as discussed below. For problems where additional physics must be accounted for, other stability restrictions may apply. When surface tension is important, it is generally found, for example, that it is necessary to limit the time step in such a way that a capillary wave travels less than a grid space in one time step.

The simple explicit forward-in-time algorithm described above is only first-order accurate. For most problems it is desirable to employ at least a second-order accurate time integration method. In such methods the nonlinear advection terms can usually be treated explicitly, but the viscous terms are often handled implicitly, for both accuracy and stability. If we use a second-order Adams–Bashforth scheme for the advection terms and a second-order Crank–Nicholson scheme for the viscous term, the predictor step is (Wesseling, 2001)

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\frac{3}{2}\mathbf{A}_\mathrm{h}(\mathbf{u}^n) + \frac{1}{2}\mathbf{A}_\mathrm{h}(\mathbf{u}^{n-1}) + \frac{\nu}{2}\Big(\mathbf{D}_\mathrm{h}(\mathbf{u}^n) + \mathbf{D}_\mathrm{h}(\mathbf{u}^*)\Big), \tag{2.9}$$

and the correction step is

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\nabla_\mathrm{h}\phi^{n+1}. \tag{2.10}$$

[1] Lewy is often spelled Levy. This is incorrect. Hans Lewy (1904–1988) was a well-known mathematician.

Here, ϕ is not exactly equal to the pressure, since the viscous term is not computed at the new time level but at the intermediate step. It is easily seen that

$$-\nabla \phi^{n+1} = -\nabla p^{n+1} + \frac{\nu}{2}\Big(\mathbf{D}_\mathrm{h}(\mathbf{u}^{n+1}) - \mathbf{D}_\mathrm{h}(\mathbf{u}^*)\Big). \tag{2.11}$$

The intermediate velocity $\mathbf{u}^*$ does not satisfy the divergence-free condition, and a Poisson equation for the pseudo-pressure is obtained as before from Eq. (2.10) as

$$\nabla_\mathrm{h}^2 \phi^{n+1} = \frac{\nabla_\mathrm{h} \cdot \mathbf{u}^*}{\Delta t}. \tag{2.12}$$

This multidimensional Poisson's equation must be solved before the final velocity, $\mathbf{u}^{n+1}$, can be obtained. Since the viscous terms are treated implicitly, we must rearrange equation (2.9) to yield a Helmholtz equation for the intermediate velocity $\mathbf{u}^*$

$$\mathbf{D}_\mathrm{h}(\mathbf{u}^*) - \frac{2\nu}{\Delta t}\mathbf{u}^* = \frac{3\Delta t}{2}\mathbf{A}_\mathrm{h}(\mathbf{u}^n) - \frac{\Delta t}{2}\mathbf{A}_\mathrm{h}(\mathbf{u}^{n-1}) - \mathbf{D}_\mathrm{h}(\mathbf{u}^n) - \frac{2\nu}{\Delta t}\mathbf{u}^n = \mathrm{RHS} \tag{2.13}$$

which needs to be solved with the appropriate boundary conditions. Considerable effort has been devoted to the solution of Poisson's and Helmholtz's equations and several packages are available (see `www.mgnet.org`, for example), particularly for rectangular grids. For curvilinear body-fitted grids the solution of Helmholtz's equation can sometimes be simplified by using implicit time advancement of the viscous term selectively only along certain directions. The viscous term in the other directions can be treated explicitly along with the nonlinear terms. Usually, the wall-normal viscous term is treated implicitly, while the tangential viscous terms can be treated explicitly (Mittal, 1999; Bagchi and Balachandar, 2003b). The resulting viscous time-step limitation, arising only from the tangential contributions to (2.13), is usually not very stringent.

The above two-step formulation of the time-splitting scheme is not unique; another variant is presented in Chapter 9. We refer the reader to standard textbooks, such as Ferziger and Perić (2002) and Wesseling (2001) for further discussions.

2.3 Spatial discretization

Just as there are many possible time integration schemes, the spatial discretization of the Navier–Stokes equations – where continuous variables are replaced by discrete representation of the fields and derivatives are approximated by relations between the discrete values – can be accomplished in many ways. Here we use the finite-volume method and discretize the governing equations by dividing the computational domain into small control volumes of finite size. In the finite-volume method we work with the average velocity in each control volume, and approximate each term in equations (2.3) and (2.4) by its average value over the control volume. To derive numerical approximations to the advection and the viscous terms, we first find the averages over each control volume:

$$\mathbf{A}(\mathbf{u}^n) = \frac{1}{\Delta V}\int_V \nabla\cdot(\mathbf{u}^n\mathbf{u}^n)\,dv = \frac{1}{\Delta V}\oint_S \mathbf{u}^n(\mathbf{u}^n\cdot\mathbf{n})\,ds \tag{2.14}$$

and

$$\mathbf{D}(\mathbf{u}^n) = \frac{1}{\Delta V}\int_V \nabla^2\mathbf{u}^n\,dv. \tag{2.15}$$

Here, ΔV is the volume of the control volume, S is the surface of the control volume, and we have used the divergence theorem to convert the volume integral of the advection term to a surface integral. It is also possible to rewrite the viscous term as a surface integral, but for constant-viscosity fluids it is generally simpler to work with the volume integral. Numerical approximations to these terms, $\mathbf{A}_\mathrm{h}$ and $\mathbf{D}_\mathrm{h}$, are found by evaluating the integrals numerically.

Similarly, a numerical approximation to the continuity equation (2.2), $\nabla_\mathrm{h}\cdot\mathbf{u}^{n+1} = 0$, is found by first integrating over the control volume and then rewriting the integral as a surface integral:

$$\frac{1}{\Delta V}\int_V \nabla\cdot\mathbf{u}^{n+1}\,dv = \frac{1}{\Delta V}\oint_S \mathbf{u}^{n+1}\cdot\mathbf{n}\,ds. \tag{2.16}$$

The surface integral is then evaluated numerically. While we started here with the differential form of the governing equations, averaging over each cell, the discrete approximations could just as well have been obtained by starting with the conservation principles in integral form.

To carry out the actual computations, we must specify the control volumes to be used, and how the surface and volume integrals are approximated. Here, we will take the simplest approach and use square or cubic control volumes, defined by a regular array of grid points, separated by a distance h. For simplicity, we assume a two-dimensional flow as the extension to three dimensions involves no new concepts. We start by picking a control volume around a point where the pressure is stored and identify it by the integer pair (i, j). The control volumes to the left and the right are given by $(i-1, j)$ and $(i+1, j)$, respectively. Similarly, $(i, j-1)$ and $(i, j+1)$ refer to the control volumes below and above. The locations of the edges are identified by half-indices $(i\pm1/2, j)$ and $(i, j\pm1/2)$. The pressure

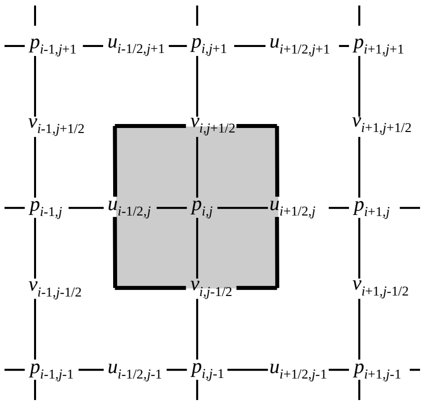

Fig. 2.1. The notation used for a standard staggered MAC mesh. The pressure control volume, centered at the (i, j) node, is outlined.

control volume, centered at (i, j), is shown by a thick line in Fig. 2.1. To derive a discrete approximation for the incompressibility condition, we evaluate equation (2.16) for the pressure control volume centered at (i, j). The integrals along the edges of the control volume are approximated by the midpoint rule, using the normal velocities at the edges. The normal velocities on the right and the left boundaries are $u_{i+1/2,j}$ and $u_{i-1/2,j}$, respectively. Similarly, $v_{i,j+1/2}$ and $v_{i,j-1/2}$ are the normal velocities on the top and the bottom boundary. The discrete approximation to the incompressibility condition is therefore:

$$u^{n+1}_{i+1/2,j} - u^{n+1}_{i-1/2,j} + v^{n+1}_{i,j+1/2} - v^{n+1}_{i,j-1/2} = 0 \tag{2.17}$$

since the grid spacing is the same in both directions.

The velocities needed in equation (2.17) are the normal velocities to the boundary of the control volume. Although methods have been developed to allow us to use co-located grids (discussed below and in Section 10.3.4), where the velocities are stored at the same points as the pressures, here we proceed in a slightly different way and to define new control volumes, one for the u velocity, centered at $(i+1/2, j)$ and another one for the v velocity, centered at $(i, j+1/2)$. Such *staggered* grids result in a very robust numerical method that is – in spite of the complex looking indexing – relatively easily implemented. The control volume for $u_{i+1/2,j}$ is shown in the left frame of Fig. 2.2 and the control volume for $v_{i,j+1/2}$ in the right frame.

Continuing with the first-order method described above, the discrete forms of equations (2.3) and (2.4) for the u velocity in a control volume centered at

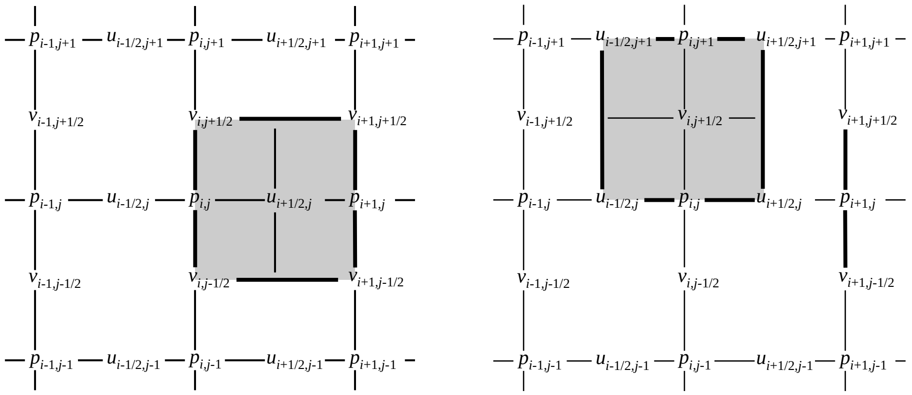

Fig. 2.2. The control volumes for the velocities on a staggered grid. The u-velocity control volume, centered at $(i+1/2, j)$, is shown on the left and the v-velocity control volume, centered at $(i, j+1/2)$, is shown on the right.

$(i+1/2, j)$ and the v velocity in a control volume centered at $(i, j+1/2)$ are:

$$\begin{aligned} u^*_{i+1/2,j} &= u^n_{i+1/2,j} + \Delta t\Big(-(A_x)^n_{i+1/2,j} + \nu(D_x)^n_{i+1/2,j} + (f_x)_{i+1/2,j}\Big) \\ v^*_{i,j+1/2} &= v^n_{i,j+1/2} + \Delta t(-(A_y)^n_{i,j+1/2} + \nu(D_y)^n_{i,j+1/2} + (f_y)_{i,j+1/2}\Big) \end{aligned} \tag{2.18}$$

for the predictor step, and

$$\begin{aligned} u^{n+1}_{i+1/2,j} &= u^*_{i+1/2,j} - \frac{1}{\rho}\frac{\Delta t}{h}(p^{n+1}_{i+1,j} - p^{n+1}_{i,j}) \\ v^{n+1}_{i,j+1/2} &= v^*_{i,j+1/2} - \frac{1}{\rho}\frac{\Delta t}{h}(p^{n+1}_{i,j+1} - p^{n+1}_{i,j}) \end{aligned} \tag{2.19}$$

for the projection step.

To find an equation for the pressure, we substitute the expression for the correction velocities at the edges of the pressure control volume, equations (2.19), into the continuity equation (2.17). When the density is constant, we get:

$$\frac{p^{n+1}_{i+1,j} + p^{n+1}_{i-1,j} + p^{n+1}_{i,j+1} + p^{n+1}_{i,j-1} - 4p^{n+1}_{i,j}}{h^2} = \frac{\rho}{\Delta t}\left(\frac{u^*_{i+1/2,j} - u^*_{i-1/2,j} + v^*_{i,j+1/2} - v^*_{i,j-1/2}}{h}\right). \tag{2.20}$$

The pressure Poisson equation, (2.20), can be solved by a wide variety of methods. The simplest one is iteration, where we isolate $p^{n+1}_{i,j}$ on the left-hand side and compute it by using already estimated values for the surrounding pressures. Once a new pressure is obtained everywhere, we repeat the process until the pressure does not change any more. This iteration (Jacobi iteration) is very robust but converges very slowly. It can be accelerated slightly by using the new values of the pressure as soon as they become available (Gauss–Seidel iteration) and even more by extrapolating toward the new value at every iteration. This is called successive over-relaxation (SOR) and was widely used for a while although now its value is mostly its simplicity during code development. For "production runs", much more efficient methods are available. For constant-density flows in regular geometries, solvers based on fast Fourier transforms or cyclic reduction (Sweet, 1974) have been used extensively for over two decades, but for more complex problems, such as variable-density or nonsimple domains or boundary conditions, newer methods such as multigrid iterative methods are needed (Wesseling, 2004). Solving the pressure equation is generally the most time-consuming part of any simulation involving incompressible flows.

Now that we have determined the layout of the control volumes, we can write explicit formulas for the advection and diffusion terms. The simplest approach is to use the midpoint rule to approximate the integral over each edge in equation (2.14) and to use a linear interpolation for the velocities at those points where they are not defined. This results in:

$$
\begin{aligned}
(A_x)^n_{i+1/2,j} = \frac{1}{h}\Bigg\{ & \left(\frac{u^n_{i+3/2,j}+u^n_{i+1/2,j}}{2}\right)^2 - \left(\frac{u^n_{i+1/2,j}+u^n_{i-1/2,j}}{2}\right)^2 \\
& + \left(\frac{u^n_{i+1/2,j+1}+u^n_{i+1/2,j}}{2}\right)\left(\frac{v^n_{i+1,j+1/2}+v^n_{i,j+1/2}}{2}\right) \\
& - \left(\frac{u^n_{i+1/2,j}+u^n_{i+1/2,j-1}}{2}\right)\left(\frac{v^n_{i+1,j-1/2}+v^n_{i,j-1/2}}{2}\right)\Bigg\}
\end{aligned}
$$

$$
\begin{aligned}
(A_y)^n_{i,j+1/2} = \frac{1}{h}\Bigg\{ & \left(\frac{u^n_{i+1/2,j}+u^n_{i+1/2,j+1}}{2}\right)\left(\frac{v^n_{i,j+1/2}+v^n_{i+1,j+1/2}}{2}\right) \\
& - \left(\frac{u^n_{i-1/2,j+1}+u^n_{i-1/2,j}}{2}\right)\left(\frac{v^n_{i,j+1/2}+v^n_{i-1,j+1/2}}{2}\right) \\
& + \left(\frac{v^n_{i,j+3/2}+v^n_{i,j+1/2}}{2}\right)^2 - \left(\frac{v^n_{i,j+1/2}+v^n_{i,j-1/2}}{2}\right)^2\Bigg\}.
\end{aligned}
$$

The viscous term, equation (2.15), is approximated by the Laplacian at the center of the control volume, found using centered differences and by assuming that the average velocities coincide with the velocity at the center of the control volume:

$$(D_x)^n_{i+1/2,j} = \frac{u^n_{i+3/2,j} + u^n_{i-1/2,j} + u^n_{i+1/2,j+1} + u^n_{i+1/2,j-1} - 4u^n_{i+1/2,j}}{h^2}$$
$$(D_y)^n_{i,j+1/2} = \frac{v^n_{i+1,j+1/2} + v^n_{i-1,j+1/2} + v^n_{i,j+3/2} + v^n_{i,j-1/2} - 4v^n_{i,j+1/2}}{h^2}. \quad (2.21)$$

The staggered grid used above results in a very robust method, where pressure and velocities are tightly coupled. The grid does, however, require fairly elaborate bookkeeping of where each variable is and, for body-fitted or unstructured grids, the staggered grid arrangement can be cumbersome. Considerable effort has therefore been devoted to the development of methods using *co-located* grids, where all the variables are stored at the same spatial locations. The simplest approach is probably the one due to Rhie and Chow (1983), where we derive the pressure equation using the predicted velocities interpolated to the edges of the basic control volume. Thus, after we find the temporary velocities at the pressure points, $u^*_{i,j}$ and $v^*_{i,j}$, we interpolate to find

$$u^*_{i+1/2,j} = \frac{1}{2}(u^*_{i+1,j} + u^*_{i,j})$$
$$v^*_{i,j+1/2} = \frac{1}{2}(v^*_{i,j+1} + v^*_{i,j}). \quad (2.22)$$

We then “pretend” that we are working with staggered grids and that the edge velocities at the new time step are given by

$$u^{n+1}_{i+1/2,j} = u^*_{i+1/2,j} - \frac{\Delta t}{\rho h}(p^{n+1}_{i+1,j} - p^{n+1}_{i,j})$$
$$v^{n+1}_{i,j+1/2} = v^*_{i,j+1/2} - \frac{\Delta t}{\rho h}(p^{n+1}_{i,j+1} - p^{n+1}_{i,j}), \quad (2.23)$$

which is identical to equation (2.19), except that the temporary edge velocities are found by interpolation (equation 2.22).

The continuity equation for the pressure control volume is still given by equation (2.17), and substitution of the velocities given by equation (2.23) yields exactly equation (2.20), but with the velocities on the right-hand side now given by equation (2.22), rather than equations (2.19). Although we enforce incompressibility using the corrected edge velocities so that the pressure gradient is estimated using pressure values next to each other, the new cell-centered velocities are found using the average pressure at the edges. Thus,

$$u_{i,j}^{n+1} = u_{i,j}^{*} - \frac{\Delta t}{\rho h}(p_{i+1,j}^{n+1} - p_{i-1,j}^{n+1})$$

$$v_{i,j}^{n+1} = v_{i,j}^{*} - \frac{\Delta t}{\rho h}(p_{i,j+1}^{n+1} - p_{i,j-1}^{n+1}). \tag{2.24}$$

The Rhie–Chow method has been successfully implemented by a number of authors and applied to a range of problems. For regular structured grids it offers no advantage over the staggered grid, but for more complex grid layouts and for cut-cell methods (discussed in Chapter 4) it is easier to implement. A different approach for the solution of the fluid equations on co-located grids is described in Section 10.3.4.

The method described above works well for moderate Reynolds number flows and short enough computational times, but for serious computational studies, particularly at high Reynolds numbers, it usually needs to be refined in various ways. In early computations the centered difference scheme for the advection terms was sometimes replaced by the upwind scheme. In this approach, the momentum is advected through the left boundary of the $(i + 1/2, j)$ control volume in Fig. 2.2 using $u_{i-1/2,j}$ if the flow is from the left to right and using $u_{i+1/2,j}$ if the flow is from the right to left. While this resulted in much improved stability properties, the large numerical diffusion of the scheme and the low accuracy has all but eliminated it from current usage. By using more sophisticated time integration schemes, such as the Adams–Bashforth method described above or Runge–Kutta schemes, it is possible to produce stable schemes based on centered differences for the advection terms that are subject to the time-step limit given by (2.8), rather than equation (2.7). However, for situations where the velocity changes rapidly over a grid cell this approach can produce unphysical oscillations. These oscillations do not always render the results unusable, and the problem only shows up in regions of high gradients. However, as the Reynolds number becomes higher the problem becomes more serious. To overcome these problems, many authors have resorted to the use of higher order upwind methods. These methods are almost as accurate as centered difference schemes in regions of fully resolved smooth flows and much more robust in regions where the solution changes rapidly and the resolution is marginal. The best-known approach is the quadratic upstream interpolation for convective kinematics (QUICK) method of Leonard (1979), where values at the cell edges are interpolated by upstream biased third-order polynomials. Thus, for example, the u velocity at (i, j) in Fig. 2.2 is found from

$$u_{i,j} = \begin{cases} (1/8)\big(3u_{i+1/2,j} + 6u_{i-1/2,j} - u_{i-3/2,j}\big), & \text{if } u_{i,j} > 0; \\ (1/8)\big(3u_{i-1/2,j} + 6u_{i+1/2,j} - u_{i+3/2,j}\big), & \text{if } u_{i,j} < 0. \end{cases} \tag{2.25}$$

While QUICK and its variants are not completely free of "wiggles" for steep enough gradients, in practice they are much more robust than the centered difference scheme and much more accurate than first-order upwind. Other authors have used the second-order ENO (essentially nonoscillatory) scheme described in Chapter 3 (for the advection of the level set function) to find the edge velocities.

2.4 Boundary conditions

At solid walls the boundary conditions for the Navier–Stokes equations are well defined. As we saw in Section 1.3, the velocity is simply equal to the wall velocity. When staggered grids are used to resolve a rectangular geometry, the grid is arranged in such a way that the boundaries coincide with the location of the normal velocities. In the discrete version of the equations, the relative normal velocity is then simply zero on the side of the control volume that coincides with the wall. Since the location of the tangent velocity component is, however, half a grid space away from the wall, imposing the tangent wall velocity is slightly more complicated. Usually, this is done by the introduction of "ghost points" on the other side of the wall, half a grid space away from the wall. The tangent velocity at this point is specified in such a way that linear interpolation gives the correct wall velocity. Thus, if the wall velocity is u_{b} and the velocity at the first point inside the domain (half a cell from the wall) is $u_{1/2}$, the velocity at the ghost point is $u_{-1/2} = 2u_{\mathrm{b}} - u_{1/2}$. Figure 2.3 shows a ghost point near a solid boundary.

While the enforcement of the tangential velocity boundary conditions is perhaps a little kludgy, it is in the implementation of the boundary conditions for the pressure where the true elegance of the staggered grid manifests itself best. As seen before, the pressure equation is derived by substituting the discrete equations for the correction velocities into the discrete continuum equation. For cells next to the boundary the normal velocity at the wall is known and there is no need to substitute the correction velocity for the boundary edge. The pressure equation for the cell will therefore only contain pressures for nodes inside the domain. This has sometimes led to declarations to the effect that on staggered grids no boundary conditions are needed for the pressure. This is, of course, not correct. The discrete

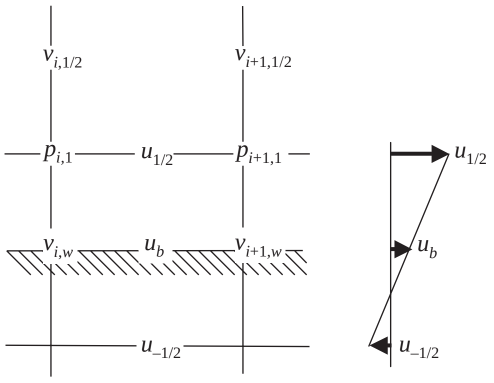

Fig. 2.3. A cell next to a solid wall. To impose the tangent velocity u_w, a "ghost" point is introduced half a grid cell outside the boundary.

pressure equation for cells next to the boundary is different than for interior nodes since the boundary conditions are incorporated during the derivation of the equation. For the pressure cell in Fig. 2.4 the inflow on the left is given ($u_{\mathrm{b},j}$), so that we only substitute equations (2.19) for the right, top, and bottom boundaries, yielding:

$$\frac{p^{n+1}_{i+1,j} + p^{n+1}_{i,j+1} + p^{n+1}_{i,j-1} - 3p^{n+1}_{i,j}}{h^2} = \frac{\rho}{\Delta t}\left\{\frac{u^*_{i+1/2,j} - u_{\mathrm{b},j} + v^*_{i,j+1/2} - v^*_{i,j-1/2}}{h}\right\} \tag{2.26}$$

which does not include a pressure to the left of the control volume. Similar equations are used for the other boundaries. At corner nodes, the velocity is known at two edges of the control volume. This expression can also be derived by substituting equation (2.19), written for $u^*_{i-1/2,j}$, into equation (2.20) and using that $u^{n+1}_{i-1/2,j} = u_{\mathrm{b},j}$.

For co-located grids, it is necessary to specify boundary conditions for the intermediate velocity in terms of the desired boundary conditions to be satisfied by the velocity at the end of the time step. For the simple case where a Dirichlet boundary condition on velocity, $\mathbf{u}_{n+1} = \mathbf{u}_{\mathrm{b}}$, is specified, the boundary condition for the intermediate velocity can be expressed as

$$\mathbf{u}^* \cdot \mathbf{n} = \mathbf{u}_{\mathrm{b}} \cdot \mathbf{n} \qquad \text{and} \qquad \mathbf{u}^* \cdot \mathbf{t} = \mathbf{u}_{\mathrm{b}} \cdot \mathbf{t} + \Delta t(2\nabla\phi^n - \nabla\phi^{n-1}) \cdot \mathbf{t}\,, \tag{2.27}$$

where $\mathbf{n}$ is the unit vector normal to the surface on which the boundary condition is applied and $\mathbf{t}$ represents the unit vector(s) parallel to the surface. From equation (2.10) the corresponding boundary condition for ϕ^{n+1} is the homogeneous Neumann condition: $\nabla\phi^{n+1} \cdot \mathbf{n} = 0$.

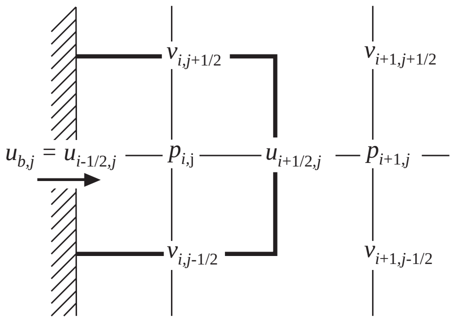

Fig. 2.4. A cell next to an inflow boundary. The normal velocity $u_{\mathrm{b},j}$ is given.

While the discrete boundary conditions for solid walls are easily obtained, the situation is quite different for inflow and outflow boundaries. Generally the challenges are to obtain as uniform (or at least well-defined) inflow as possible and to design outflow boundary conditions that have as little effect on the upstream flow as possible. Thus, the numerical problem is essentially the same as the one faced by the experimentalist and just as the experimentalist installs screens and flow straighteners, the computationalist must use *ad hoc* means to achieve the same effect.

The inclusion of a well-defined inflow, such as when a flow from a long pipe enters a chamber, is relatively easy, particularly in computations using staggered grids. Usually the normal and the tangential velocities are specified and once the normal velocity is given, the pressure equation can be derived in the same way as for rigid walls. Although a given velocity is easily implemented numerically, the inflow must be carefully selected and placed sufficiently far away from the region of interest in order to avoid imposing artificial constraints on the upstream propagation of flow disturbances. In many cases, however, such as for flow over a body, the upstream boundary condition does not represent a well-defined inflow, but rather a place in the flow where the modeler has decided that the influence of the body is sufficiently small so that the disturbance can be ignored there. For multiphase flow such situations arise frequently in studies of the flow over a single particle and we shall defer the discussion of how these are handled to Chapter 4. Although the specification of the inflow velocity conditions is by far the most common approach to simulations of internal flow, many practical situations call for the specification of the overall pressure drop. For discussions of how to handle such problems we refer the reader to standard references such as Wesseling (2001).

The outflow is a much more difficult problem. Ideally, we want the domain to be sufficiently long so that the outflow boundary has essentially no effect on the flow region of interest. Practical considerations, however, often force us to use relatively short computational domains. In reality the flow continues beyond the boundary and we accommodate this fact by assuming that the flow is relatively smooth. Thus, the trick is to do as little as possible to disturb the flow. Below we list a few of the proposals that have been put forward in the literature to allow the flow to exit the computational domain as gently as possible:

Convective (e.g. Kim and Choi, 2002):

$$\frac{\partial \mathbf{u}^*}{\partial t} + c\frac{\partial \mathbf{u}^*}{\partial n} = 0 \tag{2.28}$$

Parabolic (e.g. Magnaudet, Rivero, and Fabre, 1995):

$$\frac{\partial^2 u_n^*}{\partial n^2} = 0, \quad \frac{\partial u_\tau^*}{\partial n} = 0, \quad \frac{\partial^2 p}{\partial n \partial \tau} = 0 \tag{2.29}$$

Zero gradient (e.g. Kim, Elghobashi, & Sirignano, 1998):

$$\frac{\partial \mathbf{u}^*}{\partial n} = 0 \tag{2.30}$$

Zero second gradient (e.g. Shirayama, 1992):

$$\frac{\partial^2 \mathbf{u}^*}{\partial n^2} = 0 \tag{2.31}$$

In the above, n and τ indicate directions normal and tangential to the outer boundary and c is a properly selected advection velocity. Many other boundary conditions have been proposed in the literature to account for outflows in a physically plausible, yet computationally efficient way. The outflow boundary conditions are generally imposed for the velocity and an equation for the pressure is then derived in the same way as for solid boundaries. While the straightforward treatment of pressure for solid walls on staggered grids carries over to both in and outflow boundaries, co-located grids generally require us to come up with explicit approximations for the pressure at in and outflow boundaries. A more extensive discussion of outflow boundary conditions can be found in Section 6.5 in Wesseling (2001) and Johansson (1993), for example. Spectral simulations, owing to their global nature, place a more stringent nonreflection requirement at the outflow boundary. A buffer domain or viscous sponge technique is often used to implement a nonreflecting outflow boundary condition as discussed in Chapter 4.