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 $(i\pm 1/2, j)$ and $(i, j\pm 1/2)$. The pressure

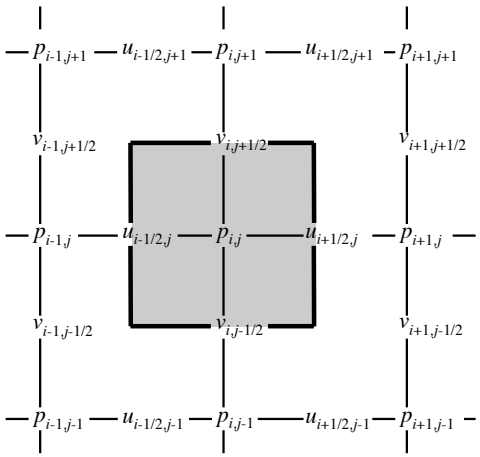

Fig. 2.1. The notation used for a standard staggered MAC mesh. The pressure control volume, centered at the (i, j) node, is outlined.

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

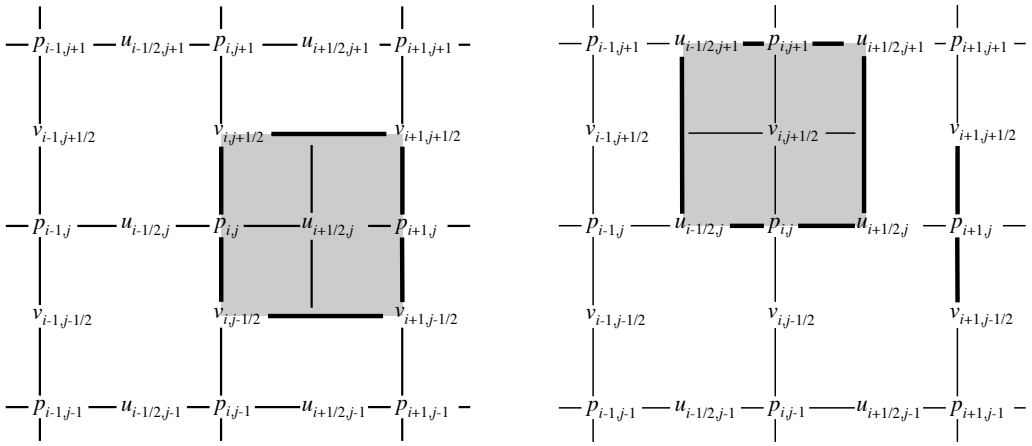

Fig. 2.2. The control volumes for the velocities on a staggered grid. The u-velocity control volume, centered at $(i+1/2, j)$, is shown on the left and the v-velocity control volume, centered at $(i, j+1/2)$, is shown on the right.

$(i+1/2, j)$ 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:

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

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 overrelaxation (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

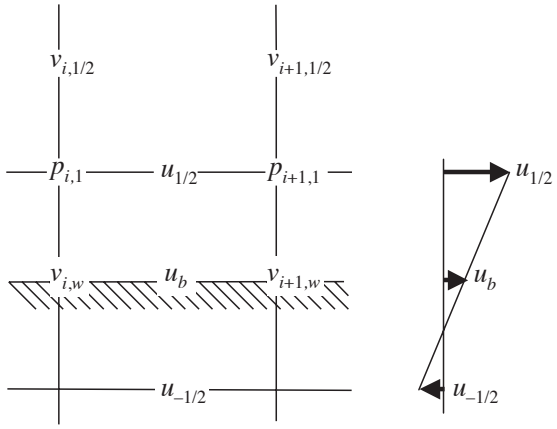

Fig. 2.3. A cell next to a solid wall. To impose the tangent velocity u_w, a "ghost" point is introduced half a grid cell outside the boundary.

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: $\nabla\phi^{n+1} \cdot \mathbf{n} = 0$.

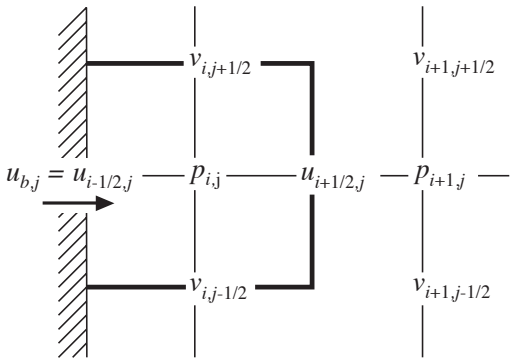

Fig. 2.4. A cell next to an inflow boundary. The normal velocity $u_{\mathrm{b},j}$ is given.

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.

3

Immersed boundary methods for fluid interfaces

Nearly half a century of computational fluid dynamics has shown that it is very hard to beat uniform structured grids in terms of ease of implementation and computational efficiency. It is therefore not surprising that a large fraction of the most popular methods for finite Reynolds number multiphase flows today are methods where the governing equations are solved on such grids. The possibility of writing one set of governing equations for the whole flow field, frequently referred to as the "one-fluid" formulation, has been known since the beginning of large-scale computational studies of multiphase flows. It was, in particular, used by researchers at the Los Alamos National Laboratory in the early 1960s for the marker-and-cell (MAC) method, which permitted the first successful simulation of the finite Reynolds number motion of free surfaces and fluid interfaces. This approach was based on using marker particles distributed uniformly in each fluid to identify the different fluids. The material properties were reconstructed from the marker particles and sometimes separate surface markers were also introduced to facilitate the computation of the surface tension. While the historical importance of the MAC method for multiphase flow simulations cannot be overstated, it is now obsolete. In current usage, the term "MAC method" usually refers to a projection method using a staggered grid.

When the governing equations are solved on a fixed grid, the different fluids must be identified by a marker function that is advected by the flow. Several methods have been developed for that purpose. The volume-of-fluid (VOF) method is the oldest and, after many improvements and innovations, continues to be widely used. Other marker function methods include the level-set method, the phase-field method, and the constrained interpolated propagation (CIP) method[1]. Instead of advecting the marker function

[1] While the initials CIP have stayed constant since the introduction of the method the actual name of the method has evolved. CIP initially stood for *cubic interpolated pseudoparticle*

directly, the boundary between the different fluids can be tracked using marker points, and the marker function then reconstructed from the location of the interface. Methods using marker points are generally referred to as "front-tracking" methods, and have been developed by a number of investigators. In addition to the difference in the way the phase boundary is advected, the inclusion of surface tension is fundamentally different in front-tracking and marker-function methods.

In this chapter we will discuss solution techniques based on the "one-fluid" formulation. In essentially all cases the Navier–Stokes equations are solved on a fixed grid using a projection method. Once the density and viscosity fields have been determined, the various methods are the same, although different implementations generally rely on slightly different flow solvers. The advection of the fluid interface, or – equivalently – the density and viscosity fields, is what distinguishes one method from another. We first present numerical solution techniques for the Navier–Stokes equations for flows with discontinuous density and viscosity fields, leaving unspecified how these fields are updated. Once we have discussed the integration of the momentum equations, we will describe in detail how the phase boundary is advected using the different methods. But first we must introduce the "one-fluid" formulation of the governing equations.

3.1 The "one-fluid" approach

When multiphase flow is simulated by solving a single set of equations for the whole flow field, it is necessary to account for differences in the material properties of the different fluids and to add appropriate interface terms for interfacial phenomena, such as surface tension. Since these terms are concentrated at the boundary between the different fluids, they are represented by delta (δ) functions. When the equations are discretized, the δ-functions must be approximated along with the rest of the equations. The material properties and the flow field are, in general, also discontinuous across the interface and all variables must therefore be interpreted in terms of generalized functions.

The various fluids can be identified by a step (Heaviside) function H which is 1 where a particular fluid is located and 0 elsewhere. The interface itself is marked by a nonzero value of the gradient of the step function. To find

then for *cubic interpolated propagation* and most recently for *constrained interpolation profile*.

the gradient it is most convenient to express H in terms of the integral over multidimensional δ-functions. For a two-dimensional field

$$H(x,y) = \int_A \delta(x-x')\delta(y-y')\,da', \tag{3.1}$$

where the integral is over an area A bounded by a contour S. H is obviously 1 if the point (x,y) is enclosed by S and 0 otherwise. To find the gradient of H, first note that since the gradient is with respect to the unprimed variables, the gradient operator can be put under the integral sign. Since the gradient of the δ-function is antisymmetric with respect to the primed and unprimed variables, the gradient with respect to the unprimed variables can be replaced by the gradient with respect to the primed variables. The resulting area (or volume, in three dimensions) integral can be transformed into a line (surface) integral by a variation of the divergence theorem for gradients. Symbolically:

$$\begin{aligned} \nabla H &= \int_A \nabla \left[\delta(x-x')\delta(y-y')\right] da' \\ &= -\int_A \nabla' \left[\delta(x-x')\delta(y-y')\right]\, da' \\ &= -\oint_S \delta(x-x')\delta(y-y')\mathbf{n}'\,ds'. \end{aligned} \tag{3.2}$$

Here the prime on the gradient symbol denotes the gradient with respect to the primed variables and $\mathbf{n}'$ is the outward unit normal vector to the interface. Although we have used that S is a closed contour, the contribution of most of the integral is zero. We can therefore replace the integral by one over a part of the contour and drop the circle on the integral:

$$\nabla H = -\int_S \delta(x-x')\delta(y-y')\mathbf{n}'\,ds'. \tag{3.3}$$

By introducing local coordinates tangent (s) and normal (n) to the front, we can write

$$\delta(x-x')\delta(y-y') = \delta(s)\delta(n), \tag{3.4}$$

and evaluate the integral as

$$-\int_S \delta(x-x')\delta(y-y')\mathbf{n}'\,ds' = -\int_S \delta(s')\delta(n')\mathbf{n}'\,ds' = -\delta(n)\mathbf{n}. \tag{3.5}$$

This allows us to use a one-dimensional delta function of the normal variable, instead of the two-dimensional one in equation (3.2). Although we have assumed a two-dimensional flow in the discussion above, the same arguments apply to three-dimensional space.

If the density of each phase is assumed to be constant, the density at each point in the domain can be represented by the constant densities and the Heaviside function:

$$\rho(x,y) = \rho_1 H(x,y) + \rho_0[1 - H(x,y)]. \tag{3.6}$$

Here, ρ_1 is the density of the fluid in which $H = 1$ and ρ_0 is the density where $H = 0$. The gradient of the density is given by

$$\begin{aligned} \nabla\rho &= \rho_1 \nabla H - \rho_0 \nabla H = (\rho_1 - \rho_0)\nabla H \\ &= \Delta\rho \int \delta(x - x')\delta(y - y')\mathbf{n}'\, ds' = \Delta\rho\delta(n)\mathbf{n}, \end{aligned} \tag{3.7}$$

where $\Delta\rho = \rho_0 - \rho_1$. Similar equations can be derived for other material properties.

The Navier–Stokes equations, as derived in Chapter 1, allow for arbitrary changes in the material properties of the fluids. While the differential form does not, strictly speaking, allow discontinuous material properties, numerical methods based on the finite-volume method are equivalent to working with the integral form of the governing equations, where no smoothness assumption is made. Furthermore, if we work with generalized functions, then we can use the equations in their original form. This is the approach that we take here. The equations as written in Chapter 1 do not, however, account for interface effects such as surface tension. Surface tension acts only at the interface and we can add this force to the Navier–Stokes equations as a singular interface term by using a δ-function,

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\nabla\cdot\mathbf{u}\mathbf{u} = -\nabla p + \rho\mathbf{f} + \nabla\cdot\mu(\nabla\mathbf{u} + \nabla^T\mathbf{u}) + \gamma\kappa\delta(n)\mathbf{n}. \tag{3.8}$$

Here, κ is the curvature for two-dimensional flows and twice the mean curvature in three-dimensions, and $\mathbf{n}$ is a properly oriented unit vector normal to the front. n is a normal coordinate to the interface, with $n = 0$ at the interface. As in Chapter 1, $\mathbf{u}$ is the velocity, p is the pressure, and $\mathbf{f}$ is a body force. With the singular term added, this equation is valid for the whole flow field, including flows with interfaces across which ρ, and the viscosity field, μ, change discontinuously. This fact justifies the "one-fluid" denomination.

For incompressible flows, the mass conservation equation is the same as for a single-phase flow

$$\nabla\cdot\mathbf{u} = 0, \tag{3.9}$$

showing that volume is conserved. For a single-phase flow where the density is constant, there is no need to follow the motion of individual fluid particles. If the density varies from one particle to another, but remains constant for each particle as it moves (as it must do for an incompressible flow), it is necessary to follow the motion of each fluid particle. This can be done by integrating the equation

$$\frac{D\rho}{Dt} = 0. \tag{3.10}$$

For multiphase flows with well-defined interfaces, where the density of each phase is a constant, we only need to find H and then construct the density directly from H as discussed above. The same arguments hold for the viscosity and other properties of the fluid.

The "one-fluid" equations are an exact rewrite of the Navier–Stokes equations for the fluid in each phase and the interface boundary conditions. The governing equations as listed above assume that the only complication in multifluid flows is the presence of a moving phase boundary with a constant surface tension. While these equations can be used to describe many problems of practical interest, additional complications quickly emerge. The presence of a surfactant or contaminants at the interface between the fluids is perhaps the most common one, but other effects such as phase changes, are common too. We will not address these more complex systems in this book.

There is, however, one issue that needs to be addressed here. While the Navier–Stokes equations, with the appropriate interface conditions, govern the evolution of a system of two fluids separated by a sharp interface, the topology of the interface can change by processes that are not included in the continuum description. Topology changes are common in multiphase flow, such as when drops or bubbles break up or coalesce. These changes can be divided into two broad classes: films that rupture and threads that break. If a large drop approaches another drop or a flat surface, the fluid in between must be "squeezed" out before the drops are sufficiently close so that the film between them becomes unstable to attractive forces and ruptures. A long, thin cylinder of one fluid will, on the other hand, generally break by Rayleigh instability where one part of the cylinder becomes sufficiently thin so that surface tension "pinches" it in two. The exact mechanisms of how threads snap and films break are still being actively investigated. There are, however, good reasons to believe that threads can become infinitely thin in a finite time and that their breaking is "almost" described by the Navier–Stokes equations (Eggers, 1995). Films, on the other hand, are generally

believed to rupture due to short-range attractive forces, once they are a few hundred angstroms thick (Edwards, Brenner, and Wasan, 1991). At the moment, most numerical simulations of topological changes treat the process in a very *ad hoc* manner and simply fuse interfaces together when they come close enough and allow threads to snap when they are thin enough. "Enough" in this context is generally about a grid spacing and it should be clear that this approach can, when the exact time of rupture is important, as it sometimes is for thin films, lead to results that do not converge under grid refinement. The pinching of threads, on the other hand, appears to be much less sensitive to the exact numerical treatment and results that are essentially independent of the grid can be obtained. How to incorporate more exact models for the rupture of films into simulations of multiphase flows is a challenging problem that is, as of this writing, mostly unsolved. Current efforts on multiscale computing, however, hold great promise.

The one-fluid formulation of the Navier–Stokes equations allows multiphase flow to be treated in more or less the same way as homogeneous flows and any standard algorithm based on fixed grids can, in principle, be used to integrate the discrete Navier–Stokes equations in time. The main difference between a Navier–Stokes solver for multiphase flows and the simple method outlined in Chapter 2 is that we must allow for variable density and viscosity and add the surface tension as a body force. For the simple first-order method of Chapter 2, the equation for the predicted velocity (equation 2.3) is modified by explicitly identifying at which time the density is computed and by adding the surface tension:

$$\mathbf{u}^* = \mathbf{u}^n + \Delta t\Big(-\mathbf{A}_\mathrm{h}(\mathbf{u}^n) + \frac{1}{\rho^n}\mathbf{D}_\mathrm{h}(\mathbf{u}^n) + \frac{1}{\rho^n}\mathbf{f}_\mathrm{b}^n + \frac{1}{\rho^n}\mathbf{F}_\gamma^n\Big). \tag{3.11}$$

The exact calculation of the surface tension term $\mathbf{F}_\gamma$ generally depends on how the marker function is advected, which is discussed below. The Poisson equation for the pressure is the same, except that the density is no longer a constant and must be explicitly included under the divergence operator

$$\nabla_\mathrm{h}\frac{1}{\rho^n}\cdot\nabla_\mathrm{h}p = \frac{1}{\Delta t}\nabla_\mathrm{h}\cdot\tilde{\mathbf{u}}. \tag{3.12}$$

This simple-looking change has rather profound implications for the solution of the pressure equation, since highly developed methods for separable equations cannot be used. Considerable progress has, however, been made in the development of efficient methods for elliptic equations with variable coefficients like equation (3.12). We refer the reader to Wesseling (2004) for a discussion and additional references. The projected velocity at time step $n+1$ is found, as before, from equation (2.4) from Chapter 2, but with the

density computed at time n:

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} - \frac{\Delta t}{\rho^n} \nabla_{\mathrm{h}} P. \tag{3.13}$$

Several variations of this algorithm are possible. Here we have used the momentum equations in the non-conservative form. If the conservative form is used, the density must be available at time $n+1$ so the density must be advected at the beginning of the time step. The conservative form can, however, lead to unrealistically large velocities around the interface, so the non-conservative form is generally recommended. For serious simulations, higher order time integration is usually used. The spatial discretization of the advection terms is the same as discussed in Chapter 2, but we must use the full deformation tensor for the viscous terms, since the viscosity is generally not constant. Although the viscous fluxes are usually computed by simply using the linearly interpolated value of viscosity for the boundaries of the control volumes, the resulting approximation does not ensure the proper continuity of the viscous fluxes when the viscosity changes rapidly. Many authors have found that working with the inverse of the viscosities improves the results (Patankar, 1980; Ferziger, 2003).

3.2 Advecting a marker function

When a fluid interface is captured on a fixed grid, a marker function f is introduced such that $f = 1$ in one fluid and $f = 0$ in the other fluid. The marker function is advected by the fluid and once the fluid velocity is known, f can, in principle, be updated by integrating

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = 0 \tag{3.14}$$

in time. Here, f is essentially the Heaviside function H introduced by equation (3.1), but in numerical implementations f can have a smooth transition zone between one value and the other. We will, therefore, use f to denote a marker function introduced for computational reasons. Integrating equation (3.14) for discontinuous data is, in spite of its apparent simplicity, one of the hard problems in computational fluid dynamics.

The importance of understanding the difficulties in advecting a discontinuous marker function justifies devoting space to examining it in some detail. To simplify the discussion, we focus on the one-dimensional advection equation, with constant advection velocity, $U > 0$:

$$\frac{\partial f}{\partial t} + U \frac{\partial f}{\partial x} = 0. \tag{3.15}$$

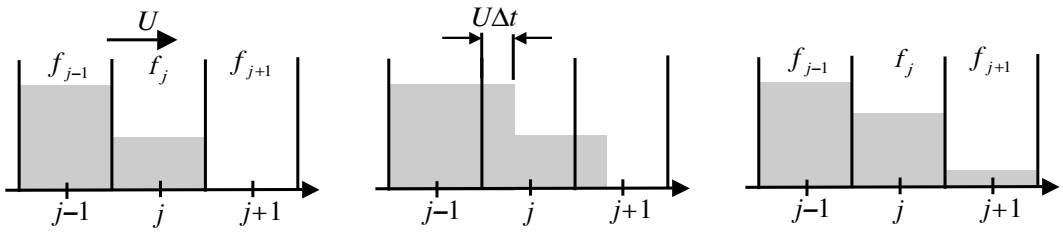

Fig. 3.1. The advection of a scalar with a constant velocity U, using the first-order upwind method. The function f is assumed to be a constant in each cell, equal to the cell average f_j^n (first frame). After the function has been moved a distance $U\Delta t$ with the velocity (middle frame), the function is averaged in each cell, giving a new cell average f_j^{n+1}.

We introduce the flux $F = Uf$ and write

$$\frac{\partial f}{\partial t} + \frac{\partial F}{\partial x} = 0. \tag{3.16}$$

To discretize this equation we use a spatial grid with evenly spaced grid points denoted by $j-1$, j, $j+1$ and so on. The grid is sketched in Fig. 3.1. The grid points represent the centers of cells or control volumes whose boundaries are located half-way between them. The boundaries of cell j are denoted by $j \pm 1/2$ and the average value of f in cell j is defined by

$$f_j = \frac{1}{h} \int_{j-1/2}^{j+1/2} f(x)\, dx, \tag{3.17}$$

where h is the grid spacing. Integrating equation (3.16) in space and time yields

$$f_j^{n+1} - f_j^n = \int_t^{t+\Delta t} F_{j+1/2}\, dt - \int_t^{t+\Delta t} F_{j-1/2}\, dt. \tag{3.18}$$

It is important to appreciate that this is an exact result. The physical meaning of equation (3.18) is that the average value of f_j increases by the difference in the flux of f in and out of the control volume. Thus, the quality of the results depends solely on how we compute F and its time integral at each cell boundary.

Since the velocity U is constant, the f profile is advected by simply shifting it to the right and the time integral of the flux across the boundary can be computed if the exact shape of f is known. Since we only store the average value f_j, we have to make some assumptions about the distribution of f in each cell. If we take f to be a constant in each cell, equal to the average f_j,

then the time integral of the flux can be computed exactly as

$$\int_t^{t+\Delta t} F_{j+1/2}\, dt = \Delta t U f_j. \tag{3.19}$$

The average value of f_j is then updated by:

$$f_j^{n+1} = f_j^n - \frac{\Delta t U}{h}(f_j^n - f_{j-1}^n). \tag{3.20}$$

Although we have introduced this method for advecting the solution by geometric considerations, it is also easily derived by approximating equation (3.15) using a first-order *upwind* (or one-sided, using the value of f at grid points j and $j-1$) finite difference approximation for the spatial derivative and a first-order forward approximation for the time derivative.

We can think of the advection of f as a two-step process. First the solution is simply shifted by $U\Delta t$ to the right. Then it is averaged to give a new uniform distribution of f in each cell. These steps in updating f_j are shown in Fig. 3.1. The initial distribution of f is shown on the left-hand side. The solution moves to the right with velocity U and the shaded area in the middle frame shows the exact solution after time Δt. However, only the average value is stored in each cell and the shaded area in the frame on the right represents the distribution after the averaging. This distribution is then used to take the next time step. The main problem with this approach is that by assuming that f in each cell is equal to the average value and by using the average value to compute the fluxes, f starts flowing out of cell j long before it is full. Or, in other words, cell $j+1$ acquires f before cell j is full. This leads to the very rapid *artificial diffusion* that the first-order advection scheme is so notorious for.

The poor performance of the method described above is entirely due to the assumption that f was uniform in each cell. The advection itself, after all, was exact. This suggests that we should try to use a more sophisticated representation for the distribution of f in each cell. If we assume that f is distributed linearly, it is still relatively easy to compute the advection exactly. There are, however, several ways to select the slopes of the line describing how f is distributed. If we assume that the slope in cell j is given by $s_j = (f_{j+1} - f_j)/h$, we obtain the centered second-order Lax–Wendroff scheme where the average is updated by:

$$f_j^{n+1} = f_j^n - \frac{\Delta t U}{2h}(f_{j+1}^n - f_{j-1}^n) + \frac{\Delta t^2 U^2}{2h^2}(f_{j+1}^n - 2f_j^n + f_{j-1}^n). \tag{3.21}$$

The performance of the two approaches outlined above is shown in Fig. 3.2 where the results of advecting a discontinuous function by both the low-order

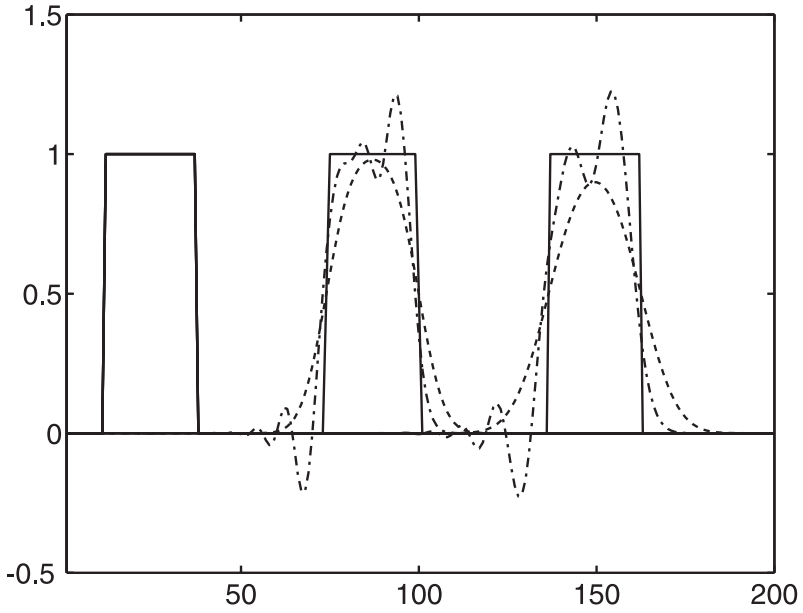

Fig. 3.2. The advection of a "blob" of f using a first-order upwind method (dashed line) and the second-order Lax–Wendroff method (dash-dotted line) in a constant velocity field $U = 1$. The exact solution is shown by a solid line. The domain has length 4 and has been resolved by 201 grid points. The time step is $\Delta t = 0.5h$. The initial conditions are shown on the left, and the solution is then shown after 125 and 250 time steps.

upwind scheme (equation 3.20) and the Lax–Wendroff scheme (equation 3.21) are presented. The initial data is the rectangle on the left. The solutions are then shown after 125 and 250 steps. Results for the upwind scheme are shown by the dashed line and the results for the higher order scheme are shown by the dash-dot line. The exact solution is the initial condition translated without a change of shape as shown by the solid line. Obviously, neither method performs very well. For the low-order scheme the discontinuity is rapidly smeared so that the sharp interface disappears, and the high-order scheme produces unacceptable wiggles or oscillations. The numerical solutions also deteriorate in time, whereas the exact solution propagates unchanged. Selecting the slopes in each cell differently results in different high-order schemes. While there are differences in their behavior, they all show oscillations near the discontinuity.

The higher order methods can be improved considerably by the use of nonlinear filters that prevent the formation of oscillations. Such "monotone" schemes have been developed to a very high degree of sophistication for gas dynamics applications where they have been used to capture a shock in essentially one grid cell. These high-order advection schemes – designed to keep shocks sharp and monotonic – are intended to deal with general

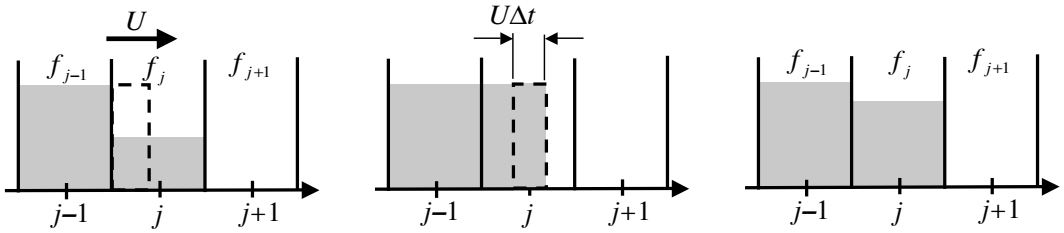

Fig. 3.3. VOF "reconstruction" for the advection of a scalar f in one dimension. Given the average value of f in each cell, f_j^n, and which neighboring cell is full and which is empty, the exact location of the interface in a half-full cell can be determined (left frame). The interface is then moved a distance $U\Delta t$ (middle frame) and a new cell averaged value f_j^{n+1} found by averaging (right frame).

hyperbolic systems such as the Euler equations. Further aspects of these schemes are discussed in Chapter 10, Section 10.2. Our equation, as well as the initial data, permit a simpler approach. The velocity field is given (or computed separately) so the equation is linear and f_j is simply 1 or 0 in most of the domain. In a partially full cell, where the value of f_j is somewhere between 0 and 1 we must have $f = 1$ in a part of the cell and $f = 0$ in the remainder. Thus, given the cell average f_j and information about whether it is the $j + 1$ or the $j - 1$ cell that is full, we can reconstruct f in cell j exactly. The advection of f_j, using this reconstruction, is shown in Fig. 3.3 where we have assumed that the full cell is on the left. On the left side of cell j, $f = 1$ and on the right hand side, $f = 0$ (dashed line in the first frame). This distribution is then advected a distance $U\Delta t$ (middle frame) and the advected distribution used to find the new average. In this example the interface remained in cell j, and no f flowed into cell $j + 1$. If the j cell is nearly full, however, then obviously some f must flow into the next cell. To account for both cases, we write the time integral of the fluxes as

$$\int_t^{t+\Delta t} F_{j+1/2}\, dt = \begin{cases} 0, & \Delta t \le (1 - f_j)h/U, \\ h - (f_j + U\Delta t), & \Delta t > (1 - f_j)h/U. \end{cases} \tag{3.22}$$

Thus, given f_j, we can compute the fluxes exactly. The advection of f_j, using this approach, is shown schematically in Fig. 3.3 and the exact solution in Fig. 3.2 (the solid line) was computed using this algorithm.

The ease by which we can advect f_j exactly in one dimenension is misleading. For two- and three-dimensional flow fields the advection becomes much more complex and the difficulties encountered in higher dimensions have stimulated the development of several methods to advect a discontinuous marker function. We will survey a few of the more common and/or successful ones below.

3.3 The volume-of-fluid (VOF) method

The volume-of-fluid or VOF method is an attempt to generalize the advection scheme described by equation (3.22) to two- and three-dimensional flows. The simplest approach, introduced by Noh and Woodward (1976), is to apply the one-dimensional algorithm directly by splitting. In this approach – usually referred to as simple line interface calculation or SLIC – the interface separating the $f = 1$ part from the $f = 0$ part of the cell is approximated by a straight line parallel to the y-axis for advection in the x-direction and parallel to the x-axis for y-advection (for two-dimensional flow). Hirt and Nichols (1981) proposed a slightly different method where the interface was still approximated by straight lines parallel to the coordinate axis, but the same orientation was used for both the x- and the y-directions. To determine whether the interface should be horizontal or vertical, Hirt and Nichols found the normal to the interface, using values in the neighboring cells and selected the orientation of the interface depending on whether the normal was more closely aligned with the x- or the y-axis. Although perhaps more appealing than the original SLIC method, tests by Rudman (1997) suggest that the Hirt–Nichols method is not significantly more accurate. In addition to distorting the interfaces, both methods generate a considerable amount of "floatsam" and "jetsam" where pieces of the interface break away in an unphysical way.

Although the method of Hirt and Nichols perhaps did not improve significantly on the SLIC approach, it nevertheless suggested that the key to improving the behavior of the advection scheme was the *reconstruction* of the interface in each cell, using the values of the marker function in the neighboring cells. In the method introduced by Youngs (1982), the interface was approximated by a straight-line segment in each cell, but the line could be oriented arbitrarily with respect to the coordinate axis. The orientation of the line is determined by the normal to the interface, which is found by considering the average value of f in both the cell under consideration, as well as neighboring cells. The results of the advection depend on the accuracy of the interface reconstruction and finding the normal accurately; therefore these operations are critical for piecewise linear interface calculation (PLIC) methods. Several methods have been proposed. Rudman (1997) recommends using

$$n^x_{i,j} = \frac{1}{h}(f_{i+1,j+1} + 2f_{i+1,j} + f_{i+1,j-1} - f_{i-1,j+1} - 2f_{i-1,j} - f_{i-1,j-1}) \quad (3.23)$$

$$n^y_{i,j} = \frac{1}{h}(f_{i+1,j+1} + 2f_{i,j+1} + f_{i-1,j+1} - f_{i+1,j-1} - 2f_{i,j-1} - f_{i-1,j-1}) \quad (3.24)$$

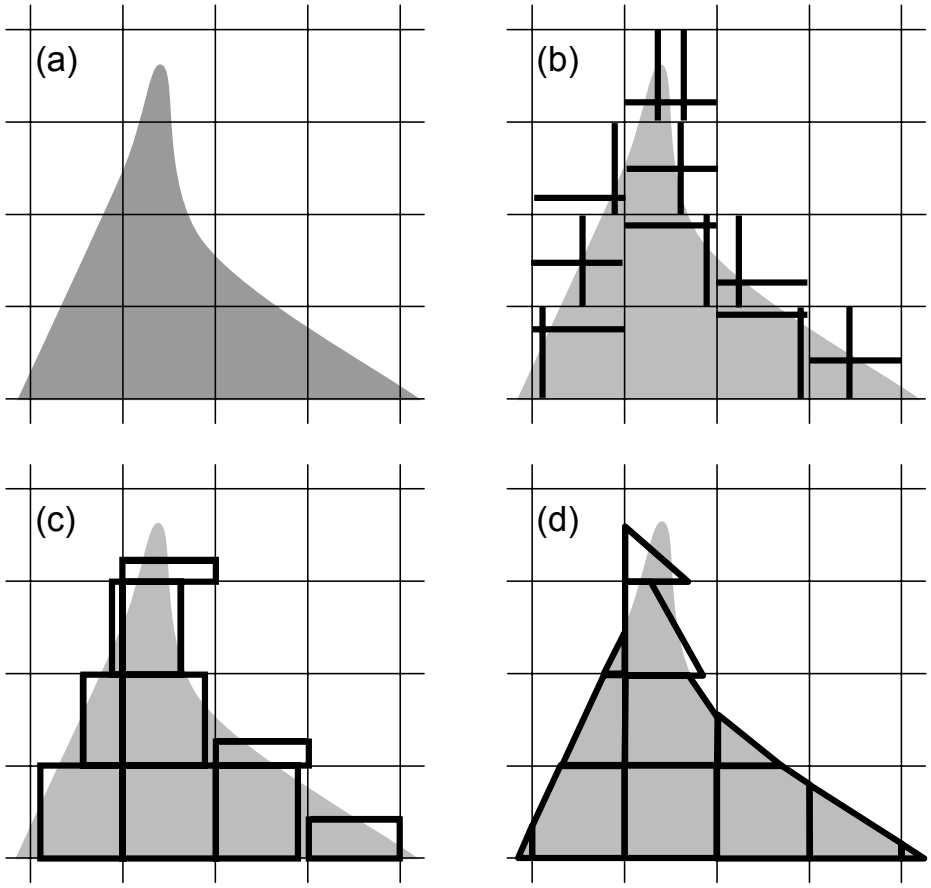

Fig. 3.4. VOF reconstruction of the solution for advection in two-dimensions. (a) The original interface. (b) The original SLIC reconstruction where the interface is always perpendicular to the advection direction. (c) The Hirt–Nichols reconstruction where the interface is parallel to one grid line. (d) PLIC reconstruction using straight lines with "optimal" orientation. Figure adapted from Rudman (1997).

introduced by Kothe, Mjolsness, and Torrey (1991). Once the normal and the average value of f in each cell, $f_{i,j}$, is known, the exact location of the interface can be determined. In two-dimensions the line segment can cross any of two adjacent or opposite cell faces, so there are four basic interface configurations. In three-dimensions there is a considerably larger number of possible configurations, adding to the complexity of the method. Figure 3.4, adapted from Rudman (1997), shows the main difference between the interface reconstruction using SLIC, the Hirt–Nichols VOF method, and PLIC. While the linear reconstruction captures slowly varying sections of the interface very well, it usually does less well for segments that are changing rapidly. Notice that the line segments are not continuous across cell boundaries.

Once the interface in each cell has been constructed, the fluxes from one cell to another are computed by geometric considerations. While fully multidimensional VOF schemes have been developed, generally it is found that split schemes, where the advection is first done in one direction and then the other, work better.

In addition to the advection of the interface, the computation of surface tension has been greatly improved in recent versions of the VOF method. In many early implementations of the VOF method the surface tension was simply ignored. However, in a widely cited paper, Brackbill, Kothe, and Zemach (1992) pointed out that the curvature of an interface represented by a marker function can be computed by taking the divergence of a normal field computed in a region extended off the interface. This extension usually exists naturally, particularly if the marker function is smoothed slightly. Once the normal field $\mathbf{n}$ is found, the mean curvature is given by

$$\kappa = \nabla \cdot \mathbf{n}. \tag{3.25}$$

This approach has become known as the continuous surface force (CSF) method.

Equation (3.25) can be implemented in several slightly different ways and generally it is found that smoothing the marker function to spread the transition zone before computing the normal field improves the curvature computations. It is also found that computing the normal in equation (3.8) using the same discretization used for the pressure gradient helps.

While the reconstruction of the interface in three dimensions is generally a fairly complex task, VOF remains one of the most widely used interface tracking method and impressive results have been obtained using the most advanced implementations. Many commercial CFD codes now include the option of simulating free-surface or multiphase flows using the VOF method. A review of the VOF method can be found in Scardovelli and Zaleski (1999).

3.4 Front tracking using marker points

Instead of updating the value of the marker function by finding the fluxes in and out of each cell, we can mark the boundary between the two fluids using connected marker points that are advected with the flow and then reconstruct the marker function from the location of the front. Marker points have been used extensively for boundary integral simulations of potential flows and Stokes flows, but the use of connected marker points to identify a boundary between two fluids governed by the full Navier–Stokes equations appears to originate with Daly (1969a,b), who used the points

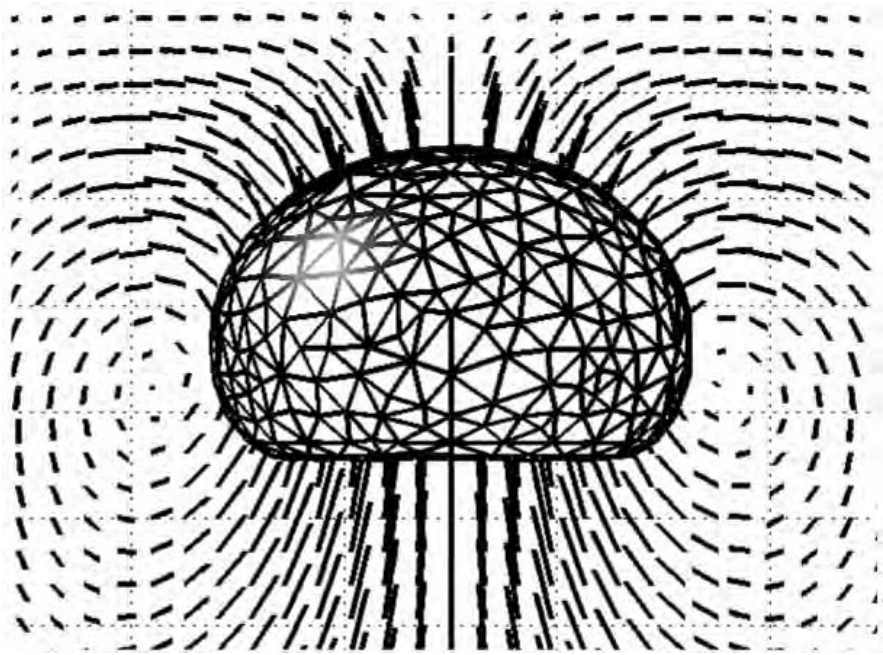

Fig. 3.5. Explicit tracking of a three-dimensional fluid interface. The interface is represented by an unstructured triangulated grid that moves with the fluid. As the interface stretches, points and elements are added and deleted.

to compute surface tension. The advection of the material properties was, however, done by the original MAC method where unconnected markers were distributed throughout the flow region. Further progress was made by Peskin (1977) who used marker points to represent elastic one-dimensional fibers in homogeneous flows and by Glimm and collaborators (Glimm, 1982, Glimm and McBryan, 1985) who tracked shocks in compressible flows by connected marker points. The first use of marker points to capture immiscible fluid interfaces in a flow governed by the full Navier–Stokes equations appears to be Unverdi and Tryggvason (1992). The method of Unverdi and Tryggvason and its derivatives have now been used to examine a large range of multiphase flows. A recent review of the method and its applications can be found in Tryggvason *et al.* (2001).

We describe the Unverdi and Tryggvason method in some detail here. Figure 3.5 shows a rising buoyant bubble whose surface is tracked by a triangular surface grid while the fluid velocity is found by solving the conservation equations on a fixed grid. The data structure forming the front consists of marker points whose coordinates are stored and triangular elements that connect the marker points. The elements also carry information about adjacent elements and material properties, in those cases where

they vary over the front. Both the points and the elements are stored in a linked list to facilitate restructuring of the front, including the addition and deletion of points and elements. Unlike methods based on following the motion of a marker function, where the marker function identifies where each fluid is, here it is necessary to construct a marker function from the location of the front. To integrate the Navier–Stokes equations in time to update the fluid velocity on the fixed grid, the surface tension must also be transferred from the front to the fixed grid. Thus, it is necessary to set up communications between the front grid and the fixed grid. Since the front represents a δ-function, the transfer corresponds to the construction of an approximation to this δ-function on the fixed grid. This "smoothing" can be done in several different ways, but it is always necessary to ensure that the quantity transferred is conserved. The interface variables, ψ_{f}, are usually expressed as a quantity per unit area (or length in two dimensions), but the grid value, ψ_{g}, should be given in terms of a quantity per unit volume. To ensure that the total value is conserved in the smoothing, we must require that:

$$\int_{\Delta s} \psi_{\mathrm{f}}(s)\, ds = \int_{\Delta v} \psi_{\mathrm{g}}(\mathbf{x})\, dv. \tag{3.26}$$

The discrete version of this condition is

$$\psi_{ijk} = \sum_{\mathrm{f}} \psi_{\mathrm{f}} w^{\mathrm{f}}_{ijk} \frac{\Delta s_{\mathrm{f}}}{h^3} \tag{3.27}$$

for a three-dimensional smoothing. Here ψ_{f} is a discrete approximation to the front value and ψ_{ijk} is an approximation to the grid value. Δs_{f} is the area of the front element. The weights, w^{f}_{ijk}, determine what fraction of the front value each grid points gets. They must satisfy

$$\sum_{ijk} w^{\mathrm{f}}_{ijk} = 1, \tag{3.28}$$

but can be selected in several different ways. The number of grid points used in the smoothing depends on the particular weighting function. Since the weights have finite support, there is a relatively small number of front elements that contribute to the value at each point of the fixed grid. Therefore, even though the summation in equation (3.27) is, in principle, over all of the front points, in practice it is sufficient to sum only over those points whose "range" includes the given grid point.

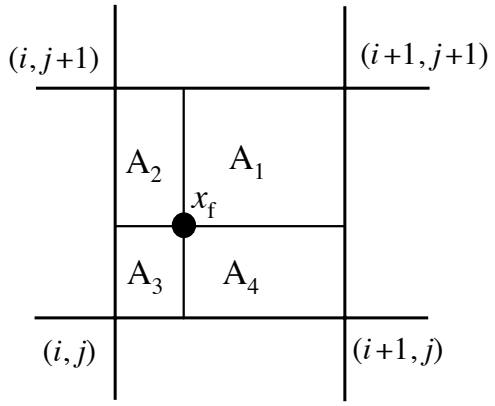

Fig. 3.6. Area weighting for smoothing and interpolation. For distribution of a front property at x_p, grid point (i, j) gets the fraction determined by A_1, grid point $(i+1, j)$ the fraction determined by A_2, and so on. When interpolating, the process is reversed and the grid values are added up at the front point, weighted by the area fractions.

We usually write the weighting functions as products of one-dimensional functions. In three dimensions, for example, the weight for the grid point (i, j, k) for smoothing a quantity from point $\mathbf{x}_\mathrm{f} = (x_\mathrm{f}, y_\mathrm{f}, z_\mathrm{f})$ is written as

$$w^\mathrm{f}_{ijk}\,(\mathbf{x}_\mathrm{f}) = d(x_\mathrm{f} - ih)\; d(y_\mathrm{f} - jh)\; d(z_\mathrm{f} - kh). \tag{3.29}$$

For two-dimensional interpolation, the third term is set to unity. $d(r)$ can be constructed in different ways. The simplest interpolation is the area (volume) weighting:

$$d(r) = \begin{cases} (h - r)/h, & 0 < r < h, \\ (h + r)/h, & -h < r < 0, \\ 0, & |r| \geq h\ . \end{cases} \tag{3.30}$$

Figure 3.6 gives a simple geometric interpretation of the weights. The weight for grid point (i, j) is the area fraction A_1, the weight for $(i + 1, j)$ is A_2, and so on. A smoother weighting function, distributing the front quantities to a larger number of grid points, was introduced by Peskin (1977) who suggested:

$$d(r) = \begin{cases} (1/4h)(1 + \cos(\pi r/2h)), & |r| < 2h, \\ 0, & |r| \geq 2h. \end{cases} \tag{3.31}$$

Other weighting functions can be found in the literature, see Lai and Peskin (2000), for example.

On the fixed grid the front is approximated by a transition zone from one fluid to the other and it is, in principle, desirable that this zone be as

thin as possible. This is achieved by spreading the front quantities to the grid points closest to the front. In the VOF method the transition between one fluid and the other takes place over one grid cell and for front tracking the area weighting function results in a similarly sharp transition. A very narrow transition zone does, however, usually result in increased small-scale anisotropy of the solution and the functions proposed by Peskin provide a smoother transition.

In many cases, particularly for high surface tension, the smoother functions improve the results considerably. Area weighting, however, involves only two grid points in each direction so it is more efficient in three dimensions where it requires values from eight grid points versus 27 for Peskin's interpolation functions. It also allows for simpler treatment of boundaries.

Since the fluid velocities are computed on the fixed grid and the front moves with the fluid, the velocity of the interface points must be found by interpolating from the fixed grid. The interpolation starts by identifying the grid points closest to the front point. The grid value is then interpolated by

$$\psi_{\mathrm{f}} = \sum_{ijk} w^{\mathrm{f}}_{ijk} \psi_{ijk}, \tag{3.32}$$

where the summation is over the points on the fixed grid that are close to the front point and ψ stands for one of the velocity components. It is generally desirable that the interpolated front value be bounded by the grid values and that the front value be the same as the grid value if a front point coincides with a grid point. Although it is not necessary to do so, the same weighting functions used to smooth front values onto the fixed grid are usually used to interpolate values to the front from the fixed grid. Once the velocity of each front point has been determined, its new position can be found by integration. If a simple first-order explicit Euler integration is used, the new location of the front points is given by

$$\mathbf{x}_{\mathrm{f}}^{n+1} = \mathbf{x}_{\mathrm{f}}^{n} + \mathbf{v}_{\mathrm{f}}^{n} \Delta t, \tag{3.33}$$

where $\mathbf{x}_{\mathrm{f}}^{n}$ is the front position at the beginning of the time step, $\mathbf{v}_{\mathrm{f}}$ is the front velocity, and Δt is the time step.

When the interface is tracked using marker particles, the fluid properties, such as the density, are not advected directly. Thus it is necessary to update these properties (or a marker function that is used to determine these properties) at every time step. There are several ways to do this. The simplest methods is, of course, to loop over the interface elements and set the value of the property on the fixed grid as a function of the shortest normal distance

from the interface. Since the interface is usually restricted to move less than the size of one fixed grid mesh, this update can be limited to the grid points in the immediate neighborhood of the interface. This straightforward approach fails, however, when two interfaces are very close to each other, or when an interface folds back on itself, such that two front segments are located between the same two fixed grid points. In these cases the proper value on the fixed grid depends on which interface segment is being considered. To overcome this problem, Unverdi and Tryggvason (1992) use the fact that the gradient of the marker function is a δ-function located at the interface. When this δ-function is smoothed onto a grid, the gradients for interfaces that are very close simply cancel. Once the grid gradient field has been constructed, the marker function must be recovered. Again, this can be done in several ways. The simplest approach is to integrate the grid gradient along grid lines from a point where the marker function is known. Since this can lead to slight grid dependence, depending upon which direction the integration is carried out, the recovery is often done by first taking the numerical divergence of the discrete grid gradient and then using this field as a source term for a Poisson equation

$$\nabla_{\mathrm{h}}^2 f = \nabla_{\mathrm{h}} \cdot (\nabla f)_{\mathrm{f}}. \tag{3.34}$$

Here $(\nabla f)_{\mathrm{f}}$ is the gradient of the marker function as constructed from the front. This equation can be solved by any standard method, but since the interface generally moves by less than a grid space, it is usually simplest to solve it by an iteration confined to a narrow band of points around the interface.

For interfaces identified by connected marker points, the computation of surface tension is, at least in principle, relatively straightforward. In most cases it is the total force on a small section of the front that is needed. In two dimensions we are generally working with a line element connecting two points and in three dimensions it is the force on a surface element connecting three points that is needed. Thus, the challenge is to find

$$\delta \mathbf{F}_\gamma = \int_{\Delta s} \gamma \kappa \mathbf{n} \, ds. \tag{3.35}$$

For a two-dimensional flow we use the definition of the curvature of a plane curve, $\kappa \mathbf{n} = \partial \mathbf{t}/\partial s$, to write equation (3.35) as

$$\delta \mathbf{F}_\gamma = \int_{\Delta s} \gamma \kappa \mathbf{n} \, ds = \gamma \int_{\Delta s} \frac{\partial \mathbf{t}}{\partial s} \, ds = \gamma(\mathbf{t}_2 - \mathbf{t}_1). \tag{3.36}$$

Therefore, instead of having to find the curvature, it is only necessary to find the tangents of the endpoints of each element. The simplest approach

is to fit a parabola to the interface points and differentiate to obtain the tangent vectors. For higher accuracy, a polynomial is fitted through more points, then differentiated to give the tangent vector.

For three-dimensional surfaces we use the fact that the mean curvature can be written as

$$\kappa \mathbf{n} = (\mathbf{n} \times \nabla) \times \mathbf{n}. \tag{3.37}$$

The force on a surface element is therefore,

$$\delta \mathbf{F}_\gamma = \gamma \int_{\delta A} \kappa \mathbf{n} \, da = \gamma \int_{\delta A} (\mathbf{n} \times \nabla) \times \mathbf{n} \, da = \gamma \oint_L \mathbf{m} \, dl, \tag{3.38}$$

where we have used the Stokes theorem to convert the area integral into a line integral along the edges of the element. Here, $\mathbf{m} = \mathbf{t} \times \mathbf{n}$, where $\mathbf{t}$ is a vector tangent to the edge of the element and $\mathbf{n}$ is a normal vector to the surface. The cross-product is a vector that is in the surface and is normal to the edge of the element. The surface tension coefficient times this vector gives the "pull" on the edge and the net "pull" is obtained by integrating around the edges. If the element is flat, the net force is zero, but if the element is curved, the net force is normal to it when the surface tension coefficient is constant. As in two dimensions, this formulation ensures that the net force on a closed surface is zero, as long as the force on the common edge of two elements is computed in the same way. Once the force on each element is known, it can be smoothed onto the fixed grid using equation (3.27).

One of the major differences between methods based on advecting a marker function and methods based on tracking the interface using marker particles is their treatment of interfaces that come close together. Physically, thin filaments of one fluid embedded in another fluid can snap and thin films can rupture. When a marker-function is advected directly, the interface is simply the place where the marker-function changes from one value to the other. If two interfaces come together, they therefore fuse together when the cells between them fill up (or empty out). Thus, topological changes (snapping of filaments and breaking of films) take place automatically. Interfaces marked by connected marker particles, on the other hand, do not reconnect unless something special is done. Which behavior is preferred? It depends. Multiphase flows obviously do undergo topological changes and in many cases marker-function methods reproduce such processes in a physically realistic way. In other cases, the topological change, particularly the rupturing of films, depends sensitively on how rapidly the thin film drains, and fusing the interfaces together once the film thickness is comparable to the grid size

may not be the right thing to do. It can, in particular, lead to solutions that do not converge under grid refinement, since the time of rupture depends directly on the grid size. In these cases, using connected marker particles gives the user more control over when the rupture takes place. Front tracking does not, however, introduce any new physics and the accurate modeling of thin films and the incorporation of appropriate models to capture their rupture remains a difficulty that is still not solved. We should note that while the "natural" behavior for the marker-function method is always to fuse interfaces together and for the front-tracking method never to do so, each method can be modified to behave like the other.

3.5 The level-set method

The level-set method, introduced by Osher and Sethian (1988) and further developed by Sussman, Smereka, and Osher (1994) for multiphase flow simulations, has emerged as the main alternative to the volume-of-fluid method for the direct advection of a marker function. As with VOF and other marker-function advection methods, level-set methods make no assumption about the connectivity of the interface. For example, if the interface separating liquid and gas should undergo a topological transition (where a drop breaks into two or more smaller drops or two drops coalesce), or form a sharp corner or cusp, there is no user intervention or extra coding necessary in order to continue the computation. Besides being robust, level-set methods allow one to accurately represent interfacial quantities such as the interfacial normal and curvature. Unlike interfaces in the VOF method, where the transition from one fluid to the next takes place over one grid cell, the level-set function transitions smoothly across the liquid–gas interface; therefore standard discretizations of the normal and the curvature using the level-set function can be as accurate as needed (e.g. second-order accurate or higher). In this section, we describe the equations and their accompanying discretizations for advecting and "reinitializing" the level-set function. We also describe the discretization details associated with coupling the level-set representation with a "one-fluid" multiphase Navier–Stokes solver.

In the level-set method, the interface separating two phases is represented using the level-set function ϕ. For a gas–liquid system, we can define ϕ to be positive in the liquid and negative in the gas. In other words,

$$\phi(\mathbf{x}, t) = \begin{cases} +d, & \mathbf{x} \text{ in the liquid,} \\ -d, & \mathbf{x} \text{ in the gas,} \end{cases}$$

where d represents the normal distance to the interface at time t.

Fig. 3.7. Level-set representation of an interface. The interface consists of two distinct circles on the left, but on the right the circles are closer and form one interface. The level-set function is constructed as a distance function.

Figure 3.7 shows the level-set function ϕ initialized as a distance function. On the left the two contours represent two distinct circular fluid blobs, but on the right the fluid blobs have merged into one.

The level-set function ϕ is advected by

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0, \tag{3.39}$$

where $\mathbf{u}$ is the fluid velocity. The level-set equation is derived using the fact that the level-set function should be constant along particle paths. In other words,

$$\frac{d\phi(\mathbf{x}(t), t)}{dt} = 0. \tag{3.40}$$

Equation (3.40), together with $d\mathbf{x}(t)/dt \equiv \mathbf{u}$, implies that

$$\frac{d\mathbf{x}(t)}{dt} \cdot \nabla \phi(\mathbf{x}, t) + \frac{\partial \phi(\mathbf{x}, t)}{\partial t} = 0,$$

which leads to equation (3.39).

In principle, the level-set function can be advected by any sufficiently accurate scheme for hyperbolic equations. In practice, however, the following simple second-order Runge–Kutta/ENO (Shu and Osher, 1989) numerical discretization of equation (3.39) has often been used. We first rewrite equation (3.39) as $\phi_t = L\phi$. Assuming that $\phi^n_{i,j}$ and $\mathbf{u}^n_{i,j}$ are valid discrete values defined at $t = t^n$, $x = x_i$, and $y = y_j$, we advance the solution to $t = t^{n+1}$ by first finding a predicted value for the level-set function

$$\phi^*_{i,j} = \phi^n_{i,j} + \Delta t L\phi^n \tag{3.41}$$

and then correcting the solution by

$$\phi_{i,j}^{n+1} = \phi_{i,j}^{n} + \frac{\Delta t}{2}(L\phi^n + L\phi^*). \tag{3.42}$$

The operator $L\phi$ is discretized by

$$L\phi = -u_{i,j}\frac{\phi_{i+1/2,j} - \phi_{i-1/2,j}}{h} - v_{i,j}\frac{\phi_{i,j+1/2} - \phi_{i,j-1/2}}{h}, \tag{3.43}$$

where the value of ϕ at the cell boundaries is found from

$$\phi_{i+1/2,j} = \begin{cases} \phi_{i,j} + \frac{1}{2}M(D_x^+\phi_{i,j}, D_x^-\phi_{i,j}), & \frac{1}{2}(u_{i+1,j} + u_{i,j}) > 0 \\ \phi_{i+1,j} - \frac{1}{2}M(D_x^+\phi_{i+1,j}, D_x^-\phi_{i+1,j}), & \frac{1}{2}(u_{i+1,j} + u_{i,j}) < 0. \end{cases} \tag{3.44}$$

M is a switch defined by

$$M(a,b) = \begin{cases} a, & |a| < |b| \\ b, & |b| \leq |a| \end{cases} \tag{3.45}$$

and the differences are defined as

$$D_x^+\phi_{i,j} = \phi_{i+1,j} - \phi_{i,j} \qquad D_x^-\phi_{i,j} = \phi_{i,j} - \phi_{i-1,j}. \tag{3.46}$$

The equation for $\phi_{i,j+1/2}$ is similar.

While the $\phi = 0$ contour is accurately advected in this way, the level-set function off the interface will, in general, not remain a distance function. Since preserving the level-set function as a distance function (at least in the vicinity of the interface) is important for the coupling with the fluid equations in simulations of multiphase flows, it is necessary to *reinitialize* the level-set function. Without reinitialization, the magnitude of the gradient of the level-set function, $|\nabla\phi|$, can become very large or very small near the zero level-set of ϕ. The large gradients in the level-set function lead to an overall loss of accuracy of equation (3.39) and in variables (e.g. velocity) that may depend on ϕ. In the reinitialization process ϕ is converted into a new level-set function ϕ_d in which ϕ and ϕ_d share the same zero level-set, and ϕ_d satisfies

$$|\nabla\phi_d| = 1 \tag{3.47}$$

for values (x, y) within M cells of the zero level-set,

$$|\phi_d| < Mh.$$

A level-set function that satisfies equation (3.47) is called a distance function because $\phi_d(x, y)$ is the signed normal distance to the zero level-set of ϕ_d. For example, the level-set functions,

$$\phi(x, y) = x^2 + y^2 - 1$$

and

$$\phi_d(x, y) = \sqrt{x^2 + y^2} - 1$$

share the same zero level-set, but ϕ_d is a distance function and ϕ is not.

Sussman, Smereka, and Osher (1994) introduced an iterative approach to reinitialize ϕ. The advantage of an iterative approach is that if the level-set function ϕ is already close to a distance function, then only a few iterations are necessary to turn ϕ into the valid distance function ϕ_d. The reinitialization step is achieved by solving the following partial differential equation,

$$\frac{\partial \phi_d}{\partial \tau} = \text{sgn}(\phi)(1 - |\nabla \phi_d|), \tag{3.48}$$

with initial conditions,

$$\phi_d(\mathbf{x}, 0) = \phi(\mathbf{x}),$$

where

$$\text{sgn}(\phi) = \begin{cases} -1, & \phi < 0 \\ 0, & \phi = 0 \\ 1, & \phi > 0 \end{cases}$$

and τ is an artificial time. The steady solutions of equation (3.48) are distance functions. Furthermore, since $\text{sgn}(0) = 0$, then $\phi_d(\mathbf{x}, \tau)$ has the same zero level-set as $\phi(\mathbf{x})$. A nice feature of this reinitialization procedure is that the level-set function is reinitialized near the front first. To see this we rewrite equation (3.48) as,

$$\frac{\partial \phi_d}{\partial \tau} + \mathbf{w} \cdot \nabla \phi_d = \text{sgn}(\phi_d), \tag{3.49}$$

where

$$\mathbf{w} = \text{sgn}(\phi) \frac{\nabla \phi_d}{|\nabla \phi_d|}.$$

It is evident that equation (3.49) is a nonlinear hyperbolic equation with the characteristic velocities pointing *outwards* from the interface in the direction of the normal. This means that ϕ_d will be reinitialized to $|\nabla \phi_d| = 1$ near the interface first. Since we only need the level-set function to be a distance function near the interface, it is not necessary to solve equation (3.49) to steady state. We may use a fixed number of iterations in order to ensure the distance function property near the interface. For example, if the iteration step size is $\Delta\tau = h/2$, and the interface is spread over a thickness $2Mh$, then we can stop the iteration process after $2M$ time steps.

The same simple second-order Runge–Kutta/ENO numerical discretization used to advance the level-set function (equation 3.39) in time can be used for the reinitialization, where the new distance function, ϕ_d, satisfies equation (3.47) for $|\phi_d| < Mh$ (Sussman, Smereka, and Osher, 1994). We first rewrite equation (3.49) as $\partial\phi_d/\partial\tau = L\phi_d$. We solve equation (3.49) for $\tau = 0, \ldots, Mh$, assuming that $\Delta\tau = h/2$; therefore we take $2M$ fictitious time steps. $(\phi_d)^n_{i,j}$ are discrete values defined at $\tau = \tau^n$, $x = x_i$ and $y = y_j$. As before, we first find a predicted value by an Euler step $(\phi_d)^*_{i,j} = (\phi_d)^n_{i,j} + \Delta\tau L\phi^n_d$ and then correct the solution using the midpoint rule, $(\phi_d)^{n+1}_{i,j} = (\phi_d)^n_{i,j} + (\Delta\tau/2)(L\phi^n_d + L\phi^*_d)$. For the discretization of $L\phi$ we have

$$L\phi = \mathrm{sgn}_{Mh}(\phi)\left(1 - \sqrt{\left(\frac{\tilde{D}_x}{h}\right)^2 + \left(\frac{\tilde{D}_y}{h}\right)^2}\right), \tag{3.50}$$

where

$$\tilde{D}_x = \begin{cases} \tilde{D}^+_x, & \mathrm{sgn}(\phi)D^+_x(\phi_d)_{i,j} < 0 \ \text{ and} \\ & \mathrm{sgn}(\phi)D^-_x(\phi_d)_{i,j} < -\mathrm{sgn}(\phi)D^+_x(\phi_d)_{i,j} \\ \tilde{D}^-_x & \mathrm{sgn}(\phi)D^-_x(\phi_d)_{i,j} > 0 \ \text{ and} \\ & \mathrm{sgn}(\phi)D^+_x(\phi_d)_{i,j} > -\mathrm{sgn}(\phi)D^-_x(\phi_d)_{i,j} \\ \frac{1}{2}(\tilde{D}^+_x + \tilde{D}^-_x) & \text{otherwise} \end{cases} \tag{3.51}$$

and the modified differences are given by

$$\begin{aligned} \tilde{D}^+_x &= D^+_x(\phi_d)_{i,j} - \frac{1}{2}M(D^+_xD^-_x(\phi_d)_{i,j}, D^+_xD^-_x(\phi_d)_{i+1,j}) \\ \tilde{D}^-_x &= D^-_x(\phi_d)_{i,j} + \frac{1}{2}M(D^+_xD^-_x(\phi_d)_{i,j}, D^+_xD^-_x(\phi_d)_{i-1,j}). \end{aligned} \tag{3.52}$$

Here, $M(a, b)$ is defined by equation (3.45) and $D^+_x\phi_{i,j}$ and $D^-_x\phi_{i,j}$ by equations (3.46). The smoothed sign function in equation (3.50) is defined by

$$\mathrm{sgn}_{Mh}(\phi) = \begin{cases} 1 & \phi \geq Mh \\ -1 & \phi \leq -Mh \\ (\phi/Mh) - \frac{1}{\pi}\sin(\pi\phi/Mh) & \text{otherwise} \end{cases} \tag{3.53}$$

To couple the level-set method with a "one-fluid" multiphase flow algorithm, we need to use the level-set function to determine the density ρ, viscosity μ, and surface tension term $\gamma\kappa\delta(n)\mathbf{n}$. From the level-set function, we generate a marker function f which is, in effect, a smoothed Heaviside function,

$$f(\phi) = \begin{cases} 0, & \text{if } \phi < -Mh \\ \frac{1}{2}(1 + \frac{\phi}{Mh} + \frac{1}{\pi}\sin(\pi\frac{\phi}{Mh})), & \text{if } |\phi| \leq Mh \\ 1 & \text{if } \phi > Mh. \end{cases} \tag{3.54}$$

The density ρ and viscosity μ are given by,

$$\rho(\phi) = \rho_1 f(\phi) + \rho_0(1 - f(\phi))$$

and

$$\mu(\phi) = \mu_1 f(\phi) + \mu_0(1 - f(\phi)).$$

The surface tension term is given by

$$\gamma\kappa\delta(n)\mathbf{n} = \gamma\kappa(\phi)\frac{df(\phi)}{d\phi}\nabla(\phi), \tag{3.55}$$

where the curvature is given by

$$\kappa(\phi) = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}. \tag{3.56}$$

Since the level-set function is naturally a smooth function, the discretization of the curvature, equation (3.56), can be done using standard central differences:

$$\begin{aligned} 2h\kappa(\phi) \approx & \left(\frac{\phi_x}{|\nabla\phi|}\right)_{i+1/2,j+1/2} + \left(\frac{\phi_x}{|\nabla\phi|}\right)_{i+1/2,j-1/2} \\ & - \left(\frac{\phi_x}{|\nabla\phi|}\right)_{i-1/2,j+1/2} - \left(\frac{\phi_x}{|\nabla\phi|}\right)_{i-1/2,j-1/2} \\ & + \left(\frac{\phi_y}{|\nabla\phi|}\right)_{i+1/2,j+1/2} - \left(\frac{\phi_y}{|\nabla\phi|}\right)_{i+1/2,j-1/2} \\ & + \left(\frac{\phi_y}{|\nabla\phi|}\right)_{i-1/2,j+1/2} - \left(\frac{\phi_y}{|\nabla\phi|}\right)_{i-1/2,j-1/2} \end{aligned}$$

where

$$(\phi_x)_{i+1/2,j+1/2} \approx (\phi_{i+1,j+1} + \phi_{i+1,j} - \phi_{i,j+1} - \phi_{i,j})/(2h)$$

and

$$(\phi_y)_{i+1/2,j+1/2} \approx (\phi_{i+1,j+1} - \phi_{i+1,j} + \phi_{i,j+1} - \phi_{i,j})/(2h).$$

We note that an appropriately smoothed Heaviside function (equation 3.54) is critical and without it the solutions may not converge under grid

refinement; we recommend using an interfacial thickness of six cells ($M = 3$). In order for f(ϕ) to be a valid "smoothed" Heaviside function (equation 3.54), ϕ must be reinitialized every time step so that ϕ is a distance function within Mh of the zero level-set.

We point out that other "field" approaches, e.g. phase-field methods, or VOF methods, can also treat coalescence and breakup without any special treatment. The level-set method is, however, significantly simpler than advanced VOF methods and extends to three-dimensional flow in a straightforward way. For more detailed discussions of the use of level-set functions to capture fluid interfaces, including more accurate/efficient discretizations, we refer the reader to Osher and Fedkiew (2003), Sethian (1999), Sussman and Fatemi (1999), Chang *et al.*, (1996), and Smereka (1996). To emphasize the ease of implementation/flexibility of the level-set approach, we point out that the level-set method can seamlessly be coupled with finite element methods or spectral element methods; see, e.g. Sussman and Hussaini (2003).

3.6 Other methods to advect the marker function

In addition to the volume-of-fluid, the level-set, and the front-tracking methods discussed above, several other methods have been proposed to simulate multiphase flows, using the "one-fluid" formulation of the governing equations. In some cases the difference is only in the way the marker function, or the fluid interface, is advected, but in other cases the differences are in how the governing fluid equations are treated.

In the CIP (constrained interpolated propagation) method, equation (3.14) is supplemented by equations for the derivatives of f, obtained by differentiating equation (3.14). Generally the equation for the derivative will be the same as equation (3.14), except that the right-hand side is not zero. However, the derivatives are first advected in the same way as f, and a correction for the effect of the nonzero right-hand side then added. To do the advection, a cubic polynomial is fitted to f and its derivatives and the solution profile obtained is translated by $U\Delta t$ (in one dimension) to give the new nodal values of f and its derivatives. Although a fully multidimensional version of the method has been developed, splitting, where each coordinate direction is done separately, works well. The original method generally shows slight oscillations near a sharp interface but these can be reduced by the use of rational polynomials (the RCIP method). For a recent review of the method and the various extensions, as well as a few applications, see Yabe, Xiao, and Utsumi (2001).

While the CIP scheme is simply another method to solve equation (3.14), the phase-field method is based on modifying the governing equations by

incorporating some of the physical effects that are believed to govern the structure of a thin interface. Although the smoothed region between the different fluids is described in a thermodynamically consistent way, in actual implementation the thickness of the transition is much larger than it is in real systems and it is not clear whether keeping the correct thermodynamics in an artificially thick interface has any advantages over methods that model the behavior of the transition zone in other ways. The phase function, which identifies the different fluids, is updated by nonlinear advection–diffusion known as the Cahn–Hilliard equation. The diffusion terms smear an interface that is becoming thin due to straining, but an antidiffusive part prevents the interface from becoming too thick if the interface is being compressed. The Navier–Stokes equations are also modified by adding a term that results in surface tension in the interface zone. The key to the modification is the introduction of a properly selected free-energy function, ensuring that the thickness of the interface remains of the same order as the grid spacing. The phase-field approach has found widespread use in simulation of solidification, but its use for fluid dynamic simulations is relatively limited. Jacqmin (1999) analyzed the method in some detail and showed examples of computations of a two-dimensional Rayleigh–Taylor instability in the Boussinesq limit. Other applications of the phase-field method to simulations of the motion of a fluid interface include Jamet *et al.* (2001) who discussed using it for flows with phase change, Verschueren, van de Vosse, and Meijer (2001) who examined thermocapillary flow instabilities in a Hele–Shaw cell, and Jacqmin (2000) who studied contact-line dynamics of a diffuse fluid interface.

Although a comprehensive comparison of the accuracy of the phase-field method for simulations of the motion of a fluid interface has not been done, the results that have been produced suggest that the method is comparable to other methods that use fixed grids and the one-fluid formulation of the governing equations. The smoothing of the transition zone and the use of front capturing (rather than explicit tracking) is likely to make the phase-field method similar to the level-set approach.

In the CIP and the phase-field method, like VOF, level-set, and front-tracking methods, the fluid interface is treated just like the rest of the fluid domain when the Navier–Stokes equations are solved. This requires replacing the sharp interface with a smooth transition zone where the transition from one fluid to the other takes place over a few grid cells. This is done both for the material properties (density and viscosity) as well as for surface tension (in the VOF method the interface is confined to one grid cell but for the surface tension calculation additional smoothing is usually necessary).

The success of the various methods based on the one-fluid formulation has demonstrated beyond a doubt the power of this approach. It has, however, also become clear that the smoothing of the interface over several grid cells limits the obtainable accuracy and often makes it difficult to obtain detailed information about the solution at the interface. Several methods have recently been proposed to improve the treatment of the interface while retaining most of the advantages of the "one-fluid" treatment. Such "sharp interface" methods originated with the work of Glimm *et al.* (1982, 1985, 1986), but more recent contributions include the "ghost fluid" method of Fedkiw *et al.* (1999) who assign fictitious values to grid points on the other side of a fluid discontinuity, the immersed interface method of Lee and LeVeque (2003), where the finite difference approximations are modified near the interface to account for the discontinuity, and the method of Udaykumar, Mittal, and Shyy (1999), where the grid cells near the interface are distorted in such a way that their faces coincide with the interface. We will discuss some of these methods in slightly greater detail in Chapter 4, in the context of simulations of flows over solid particles. None of the "sharp interface" methods have yet been used as extensively as the ones we have described above and the jury is still out as to whether the added complexity is necessary for many of the physical problems that are currently being simulated. The development of methods for multiphase flows is, however, currently an active area and it seems likely that new and improved methods will continue to emerge.

3.7 Computational examples

The observation that no matter how promising an algorithm may be theoretically, its utility can only be assessed by actual tests, is so fundamental to algorithm development that it probably deserves the status of an axiom or a law in scientific computing. In this section we will first discuss tests aimed at evaluating the performance of the various methods outlined above and then show a few examples of applications to real physical systems.

3.7.1 Validation problems

The two main challenges in following accurately the motion of the interface between two immiscible fluids are the accurate advection of the interface and the computation of the surface tension. To address the performance of the various advection schemes that have been developed, several authors have examined the passive advection of a patch of one fluid by a given velocity field. Two tests have emerged as the most popular ones. In the first one, originally introduced by Zalesak (1979), a circular blob, with a rectangular

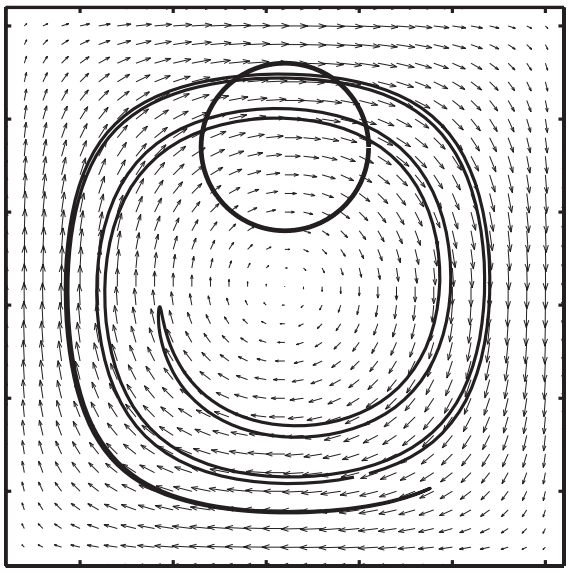

Fig. 3.8. The advection test case introduced by Rider and Kothe (1995). The circular blob near the top is deformed in a vortical flow field, until time $T = 16$ and then rotated back into its original position. The domain is given by $[0, 1]\times[0, 1]$. The undeformed blob is circular, with radius 0.15 and is initially located at $(0.5, 0.75)$.

notch almost cutting through the circle, is rotated with a fluid velocity corresponding to a solid body rotation. The circle should stay intact as it rotates, but generally it deteriorates and the degree of deterioration after a given number of rotations is used to distinguish between the different advection schemes. While this test assesses the ability of the advection scheme to preserve the sharp corners in the initial conditions, it does not test how well the scheme performs for shear flow. To evaluate the performance of various methods in such flows, another test case, introduced by Rider and Kothe (1995), has become essentially *the* standard advection test. Here, an initially circular blob of one fluid is advected in a vortex given by

$$U(x, y) = \cos \pi x \times \sin \pi y, \qquad V(x, y) = -\sin \pi x \times \cos \pi y \tag{3.57}$$

in a $[0, 1] \times [0, 1]$ domain. The radius of the blob is 0.15 and it is initially located off the vortex center so it is deformed by the flow. The computations are generally run until the blob has deformed significantly and then the flow is reversed until the blob should have returned to its original shape.

Figure 3.8 shows the boundaries of the initial circular blob, the boundary after time $T = 16$, and the boundary after the velocity field has been reversed. The computations are done using connected marker particles, the velocity is given on a 32^2 grid, and area weighting is used to interpolate the velocities. The time integration is done using a second-order Runge–Kutta

scheme with $\Delta t = 0.005$. Here, a large number of marker particles were used, so the circle returns to essentially exactly its original position when the sign of the velocity field is reversed.

The vortical flow shown in Fig. 3.8 was used by Rider and Kothe (1995) and Rudman (1997) to study the performance of several advection schemes. The methods examined included various implementations of the VOF method (the original SLIC method, the Hirt–Nichols method, and a PLIC method), the first-order upwind method, the piecewise parabolic method of Colella and Woodward (1984), and level-sets. Rudman (1997) used the original Youngs' version of the PLIC method but Rider and Kothe (1995) used a modified version of a least-square reconstruction technique introduced by Pilliod and Puckett (1997). Both authors found that the best results were obtained by the PLIC method. Rider and Kothe (1995) found that the level-set method ranked next and then SLIC. Rudman (1997) found that the Hirt and Nichols version of the method offered little improvement over the original SLIC. Rider and Kothe (1995) also tested a method based on uniformly distributed marker particles used to represent the blob. While more expensive than the other methods tested, the markers generally performed better than the marker functions. Neither of these authors used connected marker particles, but as Fig. 3.8 shows, even on a relatively coarse grid this method produces essentially the exact solution.

In addition to advection, the accurate computation of surface tension poses a challenge to methods designed for simulations of multiphase flows. To test the performance of surface tension calculations three tests have emerged as the most common ones. The first is simply a static test where the pressure inside a circular or a spherical drop is compared with the predictions of Laplace's law. Every method in current use passes this test for moderate values of the surface tension coefficient. The second test is a comparison of the oscillation frequency of a circular or spherical drop (see Lamb, 1932, for example) with analytical results for small-amplitude oscillations. In addition to the oscillation frequency, the viscous decay of the drop is sometimes also compared to theoretical predictions. The third test is essentially the same as the first one, but we look at the velocity field, which should be zero for a circular or spherical drop. For high surface tension, all methods based on the one-fluid formulation on fixed grids generally suffer from artificial currents induced by a mismatch between the resolution of the pressure and the surface tension terms. In early implementations of surface tension in VOF methods using the continuous surface force (CSF) (Brackbill, Kothe, and Zemach, 1992; Lafaurie *et al.*, 1994) these parasitic currents posed a serious limitation on the range of problems that could be

solved. While the front-tracking method of Unverdi and Tryggvason (1992) also shows parasitic currents at high surface tension, these are generally smaller. The origin of these currents is now reasonably well understood and remedies are emerging. The key seems to be that the normal vector to the interface should be found as the gradient of the indicator function and discretized in exactly the same way as the pressure gradient. If the curvature is constant this results in no parasitic currents. If the curvature is not constant it must be found from information about the geometry of the interface. In the PROST-VOF method of Renardy and Renardy (2002) the curvature is found by fitting a parabolic surface to the interface, resulting in a significant reduction of the parasitic currents. For phase-field methods, Jamet, Torres, and Brackbill (2002) have developed an energy-conserving discretization of the surface tension terms and shown that the parasitic currents are greatly reduced by its use. A similar remedy should work for level-set methods. For front-tracking methods, Shin *et al.* (2005) introduced a hybrid technique where the normal vector is found by differentiating the reconstructed marker function but a grid curvature is found from the front. This approach seems to essentially eliminate any spurious currents.

In addition to the problems with high surface tension, other difficulties emerge as the various governing parameters take on larger values. It is, in particular, generally true that more care has to be taken when the ratios of the material properties of the different fluids become large. It is therefore important to test each method under those circumstances, and although different authors have used different tests, no *standard* tests have emerged.

High density ratios cause two problems. At high Reynolds numbers, the appearance of high and irregular velocities near the interface can sometimes destroy the solution when the fully conservative form of the advection terms is used. These high velocities can be traced to an incompatibility in the advection of mass and momentum and can be overcome simply by using the nonconservative form of the advection terms (see Esmaeeli and Tryggvason, 2005, for a discussion). This is slightly counterintuitive since the conservative form is usually preferred for computations of shocks in compressible flows. The other problem is the solution of the elliptic equation for the pressure. Iterative solvers generally take a long time to converge, but – as long as the density is positive – a simple iterative technique like SOR will always converge with the proper selection of the over-relaxation constant. We should note, however, that for a density ratio of 1000, say, a very minor error – relative to the density in the heavier liquid – can lead to negative density values. More efficient multigrid solvers may, on the other hand, fail to converge. While some progress has been made, the development of

efficient pressure solvers for high-density ratios remains high on the wish-list of nearly everybody engaged in the use of one-fluid methods for multiphase flow simulations. In many applications, relatively modest values of the density and viscosity ratios can be used as approximations for larger differences, without influencing the solution too much. This is particularly true when the dynamics of the low viscosity/density fluid is largely controlled by what the high viscosity/density fluid is doing. In other cases, such as for wind-generated breakup and droplet suspensions, it is more important to use the correct properties of both fluids.

For large viscosity ratios there are generally inaccuracies around the interface due to the smoothing of the transition zone. As the error is generally "pushed" to the less viscous side, it tends to be small when the flow in the less viscous liquid has relatively minor impact on the global behavior. This is usually the case for bubbles, and for short-time evolution of drops, such as for drop collisions and splashing. For viscous drops suspended in a less viscous liquid, the errors in the viscous terms are more serious and the convergence rate is worse. It has been known for quite some time, in the context of VOF methods, that taking the harmonic mean of the viscosities at points where the viscosity is not defined, instead of the arithmetic average, ameliorated the discontinuity of the viscous stresses (Patankar, 1980). Ferziger (2003) expanded this approach to interfaces smoothed over several grid points.

3.7.2 Applications to physical problems

The "one-fluid" approach has been used to examine multifluid problems since the beginning of computational fluid mechanics. Using the marker-and-cell (MAC) method, Harlow and Welch (1965) studied gravity currents by looking at the motion of fluids caused by the sudden release of a dam and Harlow and Welch (1966) examined the nonlinear evolution of the Rayleigh–Taylor instability where a heavy fluid falls into a lighter one. Both the breaking dam and the Rayleigh–Taylor problems have since become standard test problems for multifluid codes. The MAC method, modified in several ways, has been used to study a number of other free-surface and interface problems, including splashing of drops on a liquid layer, collision of drops, free surface waves and cavitation. While many of these results were in excellent agreement with experiments, limited computer power restricted the available grid size and confined most of the studies to two-dimensional (or axisymmetric) systems.

During the last decade a large number of computations have been carried out for a large number of systems and it would be impractical to provide a complete review of these simulations and what has been learned from the results. We will, therefore, simply show a few examples of recent results, mostly for bubbly flows, demonstrating the capabilities of current methods.

Homogeneous bubbly flow with many buoyant bubbles rising together in an initially quiescent fluid is perhaps one of the simplest examples of dispersed flows. Such flows can be simulated using periodic domains where the bubbles in each period interact freely, but the configuration is repeated infinitely many times in each coordinate direction. In the simplest case there is only one bubble per period so the configuration of the bubbles does not change as they rise. While such regular arrays are unlikely to be seen in an experiment, they provide a useful reference configuration for a freely evolving array. As the number of bubbles in each period is increased, the regular array becomes unstable and the bubbles generally rise unsteadily, repeatedly undergoing close interactions with other bubbles. The behavior is, however, statistically steady and the average motion (averaged over long enough time) does not change. While the number of bubbles clearly influences the average motion for a small enough number of bubbles per period, the hope is that once the size of the system is large enough, information obtained by averaging over each period will be representative of a truly homogeneous bubbly flow.

Tryggvason and collaborators have examined the motion of nearly spherical bubbles in periodic domains in a number of papers, using improved versions of the front-tracking method originally described by Unverdi and Tryggvason (1992). Esmaeeli and Tryggvason (1998) examined a case where the average rise Reynolds number of the bubbles remained relatively small (1–2) and Esmaeeli and Tryggvason (1999) looked at another case where the Reynolds number was 20–30. Bunner and Tryggvason (2002a,b) simulated a much larger number of three-dimensional bubbles using a parallel version of the method used by Esmaeeli and Tryggvason. Their largest simulations followed the motion of 216 three-dimensional buoyant bubbles per periodic domain for a relatively long time. Simulations of freely evolving arrays were compared with regular arrays and it was found that while freely evolving bubbles at low Reynolds numbers rise faster than a regular array (in agreement with Stokes flow results), at higher Reynolds numbers the trend is reversed and the freely moving bubbles rise slower. The time averages of the two-dimensional simulations were generally well converged, but exhibited a dependency on the size of the system. This dependency was stronger for the low Reynolds number case than the moderate Reynolds number one.

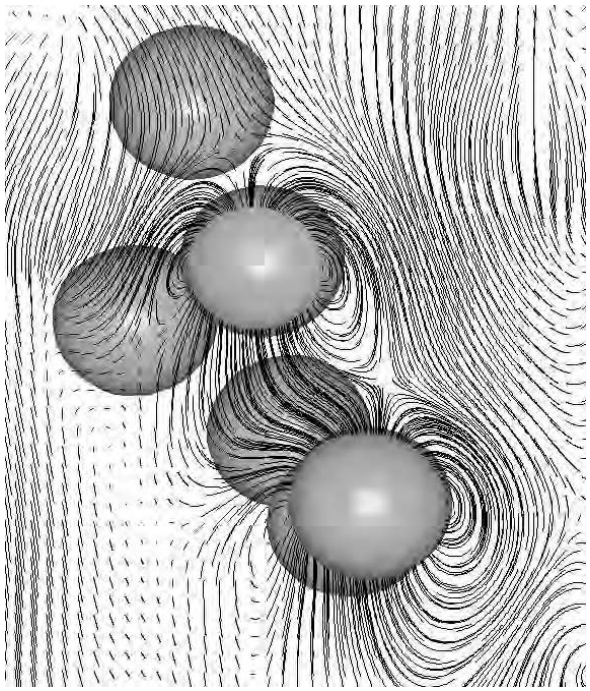

Fig. 3.9. A close-up of the flow field around a few bubbles, from a simulation of the buoyant rise of 91 bubbles in a periodic domain. Here $Eo = 1$ and $M = 1.234 \times 10^{-6}$ and the void fraction is 6%. The simulation was done using a 192^3 grid. From Bunner (2000).

The effect of deformability was studied by Bunner and Tryggvason (2003) who found that relatively modest deformability could lead to a streaming state where bubbles gathered in a "stream" or a "chimney." For extensive discussion of the results, and the insight generated by these studies, we refer the reader to the original papers. Figures 3.9 and 3.10 show two examples from the simulations of Bunner and Tryggvason. In Fig. 3.9 a close-up of the flow field around a few nearly spherical bubbles is shown. The streamlines are plotted in a plane cutting through a few of the bubbles and it is clear that the flow field is well resolved. In Fig. 3.10, one frame from a simulation of 27 deformable bubbles, while the bubbles are still relatively uniformly distributed, is shown. In addition to the bubbles and the velocity vectors in a plane cutting through the domain, the vorticity is shown by gray-scale contours. The vorticity is highest (light gray) in the wake of the bubbles, as expected.

While bubbles in initially quiescent liquid in fully periodic domains are a natural starting point for direct numerical simulation of bubbly flows, many of the important questions that need to be addressed for bubbly flows involve walls. Figure 3.11, from Lu, Fernandez, and Tryggvason (2005), shows one frame from a simulation of 16 bubbles in the so-called "minimum

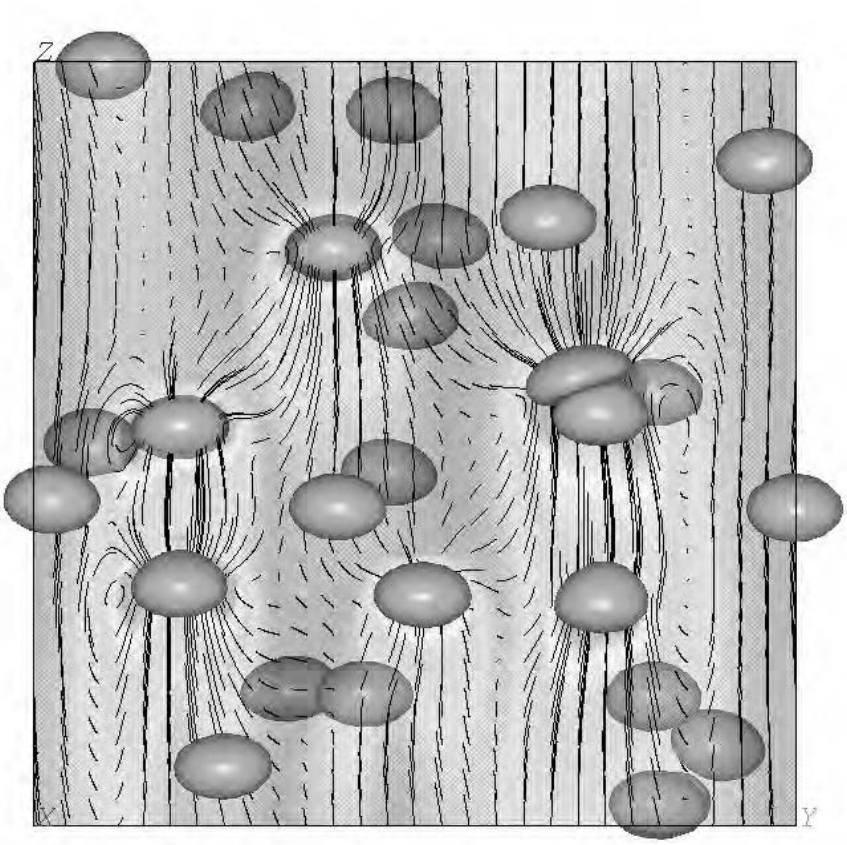

Fig. 3.10. One frame from a simulation of 27 deformable bubbles in a periodic domain. Here $Eo = 5$ and $M = 1.543 \times 10^{-4}$ and the void fraction is 2%. The simulation was done using a 128^3 grid. See also color plate. From Bunner (2000).

turbulent channel" of Jimenez and Moin (1991). In addition to the bubbles, isocontours of spanwise vorticity are shown, with different color indicating positive and negative vorticity. The contours on the bottom wall show the local wall shear. The goal of this simulation is to cast some light on the mechanisms underlying drag reduction due to the injection of bubbles near the wall and to provide data that may be useful for the modeling of such flows. Experimental studies (see Merkle and Deutsch, 1990; Kato *et al.*, 1995, and Kodama *et al.*, 2003, for a review) show that the injection of a relatively small number of bubbles into a turbulent boundary layer can result in a significant drag reduction. The flow rate in this simulation is kept constant so the total wall drag changes as the flow evolves. At the time shown, total wall shear has been reduced by about 15%. While drag reduction is usually found for deformable bubbles, nearly spherical bubbles generally lead to drag increase since they come closer to the wall and are

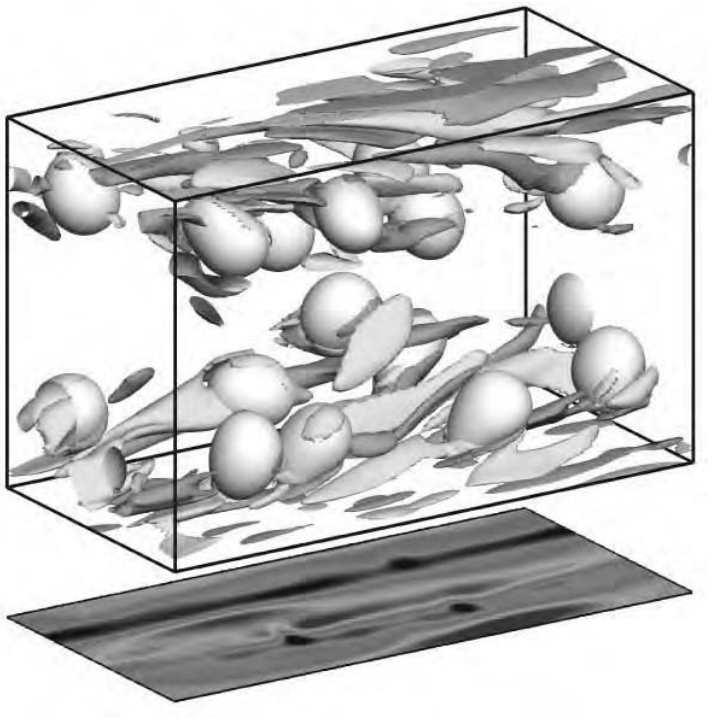

Fig. 3.11. One frame from a simulation of 16 bubbles in a turbulent channel flow. The wall Reynolds number is $Re^+ = 135$ and the initial flow, computed using a spectral code, is fully turbulent. The computations were done using a grid of $256 \times 128 \times 192$ grid points, uniformly spaced in the streamwise and the spanwise direction, but unevenly spaced in the wall normal direction. See also color plate. From Lu, Fernandez, and Tryggvason (2005).

slowed down by the viscous sublayer. It is unlikely that the subtle effect of bubble deformation on the wall drag found here could have been uncovered in any way other than by direct numerical simulations.

The results in Figs 3.9–3.11 were calculated using marker points to track the interface between the air in the bubbles and the ambient liquid. The remaining examples in this chapter have been computed using the level-set method. In Fig. 3.12 we show the evolution of a buoyant bubble where an air bubble rises in silicone oil (density ratio 642:1, viscosity ratio 27:1) computed using a 64^3 grid. According to the chart in Bhaga and Weber (1981), the bubble should rise unsteadily and "wobble." Although the computation is done using fully three-dimensional bubbles, the initial conditions are axisymmetric and the bubble therefore does not show the sidewise motion observed experimentally. The bubble does, however, oscillate. Two computations showing a fully three-dimensional motion are shown in Fig. 3.13 where the

Fig. 3.12. Numerical computation of a three-dimensional "wobbly" bubble. Results compare well with predicted behavior (Clift, Grace, and Weber, 1978). Parameters correspond to air inside the bubble and silicone oil surrounding the bubble. Density ratio and viscosity ratio are 642:1 and 27:1, respectively. See also color plate. From Ohta *et al.* (2004).

bubble on the left is rising along a spiral path, but the bubble on the right wobbles as it rises.

In Fig. 3.14 we show the evolution of a fully three-dimensional bubble collapse. Initially, a high-pressure spherical adiabatic bubble, with radius 1/2, is placed close to a solid wall in a quiescent liquid. Here, the bubble center is initially located a dimensionless distance of 3/2 from the wall. The overall size of the computational domain is $16 \times 16 \times 8$. For a similar simulation using an axisymmetric geometry, see Sussman (2003). As the bubble collapses, a thin jet, directed into the bubble, forms on the side of the bubble away from the wall and eventually penetrates through the bubble and impinges the solid wall. To capture this jet accurately, local patches of finer grids are used where needed. The overall coarse grid resolution is $64 \times 64 \times 32$ and three levels of adaptivity are used, giving an effective resolution of $512 \times 512 \times 256$ grid points. The emergence of localized small features is common in multiphase flow simulations and the use of adaptive mesh refinement (AMR) greatly increases the range of problems that can be solved in a reasonable amount of time. Other examples of the use of AMR in multifluid simulations include Agresar *et al.* (1998), Roma, Peskin,

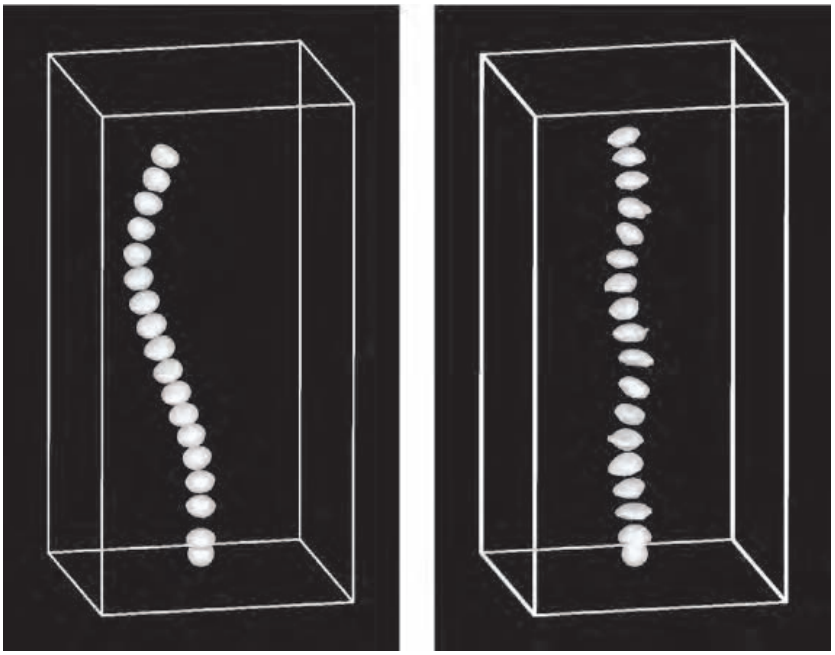

Fig. 3.13. The unsteady rise of a single bubble. Left frame: $Re = 700$, $Eo = 1.9$; Right frame: $Re = 1000$, $Eo = 7.5$. In both cases $M = 1.9 \times 10^{-10}$. Each bubble is shown at several times. From Ohta *et al.* (2004).

and Berger (1999), and Sussman *et al.* (1999). Simulations following an interface using a level-set on an adaptive unstructured mesh can be found in Zheng *et al.* (2005).

Our final example is a three-dimensional computation from Aleinov, Puckett, and Sussman (1999) of the ejection of ink by a microscale jetting devices, using a level-set method with adaptive grid refinement. Figure 3.15 shows three snapshots of the evolution after a drop has formed and as the thin filament trailing the drop breaks up into satellite drops due to surface tension. The density ratio in this computation corresponds to that for air/water. By adjusting parameters such as inlet pressure, geometry, etc. one gets different numbers/sizes of satellite drops which in turn affect the quality of the printing, once the drops land on the paper.

The level-set method has recently been used by several researchers to study various multiphase flow problems. These include Tran and Udaykumar (2004) (high-speed solid–fluid interaction), Carlson, Mucha, and Turk (2004) (low-speed solid-fluid interaction), Dhir (2001) (boiling), and Goktekin, Bargteil, and O'Brien (2004) (viscoelastic multiphase flows). Other applications can be found in the book by Osher and Fedkiw (2003).

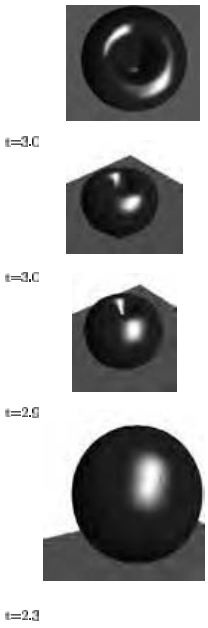

Fig. 3.14. Numerical computation of a three-dimensional collapsing bubble in the presence of a rigid boundary. Three levels of grid adaptivity are used, giving an effective fine grid resolution of $512 \times 512 \times 256$ grid points. See also color plate. From Sussman (2005).

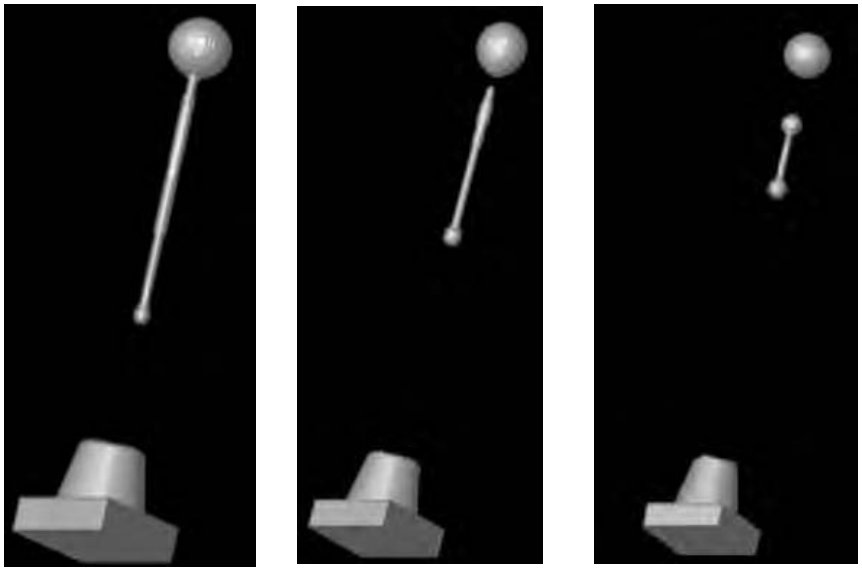

Fig. 3.15. Computation of jetting of ink using a fully three-dimensional code with adaptive grid refinement. Fluid is ejected from the solid block at the bottom. The jet is shown at three times, as it forms one large drop and several satellite drops. From Aleinov, Puckett, and Sussman (1999).

3.8 Conclusion

In this chapter we have discussed how to compute the motion of a fluid interface using fixed grids. We introduced the "one-fluid" formulation of the governing equations and discussed in some detail the various strategies that have been developed to update the marker function identifying the different fluids. Although the original idea of the "one-fluid" approach goes back to the very beginning of computational fluid dynamics, major progress has been made in the last decade. It is, in particular, safe to say that given enough computational resources we can now routinely solve a large range of problems that involve the motion of two fluids. It is possible, for example, to simulate accurately the evolution of finite Reynolds number disperse flows of several hundred bubbles and drops for sufficiently long time that reliable values can be obtained for various statistical quantities.

The "one-fluid" approach outlined in this chapter is likely to continue to remain an important strategy to handle multifluid and multiphase problems. This approach is relatively simple and enormously powerful. While the different methods of updating the marker function may at first appear very different – and there certainly are differences in the complexity and accuracy of these methods – their similarities are probably more important. There is, in particular, a considerable amount of migration of ideas developed in the context of one method to other methods. In the near future we are likely to see further convergence of the different methodologies, as well as the development of hybrid methods combining the best aspects of the different techniques.

4

Structured grid methods for solid particles

In this chapter we discuss numerical methods based on structured grids for direct numerical simulations of multiphase flows involving solid particles. We will focus on numerical approaches that are designed to solve the governing equations in the fluid and the interaction between the phases at their interfaces.

In methods employing structured grids, there are two distinct possibilities for handling the geometric complexities imposed by the phase boundaries. The first approach is to precisely define the phase boundaries and use a body-fitted grid in one or both phases, as necessary. The curvilinear grid that conforms to the phase boundaries greatly simplifies the specification of interaction processes that occur across the interface. Furthermore, numerical methods on curvilinear grids are well developed and a desired level of accuracy can be maintained both within the different phases and along their interfaces. The main difficulty with this approach is grid generation; for example, for the case of flow in a pipe with several hundred suspended particles, obtaining an appropriate structured body-fitted grid around all the particles is a nontrival task. There are several structured grid generation packages that are readily available. Nevertheless, the computational cost of grid generation increases rapidly as geometric complexity increases. This can be severely restrictive, particularly in cases where the phase boundaries are in time-dependent motion. The cost of grid generation can then overwhelm the cost of computing the flow.

The alternative to a body-fitted grid that is increasingly becoming popular are Cartesian grid methods. Here, irrespective of the complex nature of the internal and external boundaries between the phases, only a regular Cartesian mesh is employed and the governing equations are solved on this simple mesh. The interfaces between the different phases are treated as embedded boundaries within the regular Cartesian mesh. Although the