Chapter 2
Numerical Methods for Fluids

Abstract After reading this chapter, you will have insight into a number of other fluid simulation methods and their advantages and disadvantages. These methods are divided into two categories. First, conventional numerical methods based on discretising the equations of fluid mechanics, such as finite difference, finite volume, and finite element methods. Second, methods that are based on microscopic, mesoscopic, or macroscopic particles, such as molecular dynamics, lattice gas models, and multi-particle collision dynamics. You will know where the particle-based lattice Boltzmann method fits in the landscape of fluid simulation methods, and you will have an understanding of the advantages and disadvantages of the lattice Boltzmann method compared to other methods.

While the equations of fluid mechanics described in Sect. 1.1 may *look* relatively simple, the behaviour of their solutions is so complex that analytical flow solutions are only available in certain limits and for a small number of geometries. In particular, the equations' non-linearity and the presence of boundary conditions of complex shape make it extremely difficult or even impossible to find analytical solutions. In most cases, we have to solve the equations numerically on a computer to find the flow field. The field of computational fluid dynamics (CFD) started soon after the advent of electronic computers, although numerical solution of difficult equations is a much older topic.[1]

At this point, a wide variety of methods for finding fluid flow solutions have been invented. Some of these methods are general-purpose methods, usable for any partial difference equation (PDE), which have been applied to fluids with minor adaptations. Other methods are more tailored for finding fluid flow solutions.

While the lattice Boltzmann method is the topic of this book, it is simply one of the many, many methods available today. Each of these methods has its own advantages and disadvantages, and the LB method is no exception. Therefore, this chapter briefly covers the most similar methods and relevant alternatives to LB, in order to give some perspective on where LB fits in the wider landscape of methods, and to give some idea of the cases for which LB can be better than other methods.

[1] Before electronic computers, numerical solutions were performed manually by people whose job title was "computer"!

T. Krüger et al., *The Lattice Boltzmann Method*, Graduate Texts in Physics,
DOI 10.1007/978-3-319-44649-3_2

One concept that we will be referring to often in this chapter is **order of accuracy**, which will be covered in more depth in Sect. 4.5.1. It is tied to *truncation errors*, i.e. the inherent errors when solving a PDE numerically. A numerical solution is always an approximation of the "true" solution, and typically deviates from it by truncation error terms proportional to resolution parameters like the time step Δt and spatial step Δx. For example, one particular numerical method could deviate from the "true" solution by terms $O(\Delta t^2) + O(\Delta x^4)$, so that as the resolution is made finer the deviation would decrease with the second power of Δt and the fourth power of Δx. This method is said to have a *second order* accuracy in time, and a *fourth order* accuracy in space. Section 2.1.1 shows where the truncation error comes from in the case of finite difference approximations.

Section 2.1 covers "conventional" Navier-Stokes solvers, i.e. "top-down" methods where the macroscopic fluid equations are directly discretised and solved by the aforementioned general-purpose methods. Section 2.2 covers particle-based methods, typically "bottom-up" methods based on microscopic or mesoscopic fluid descriptions. The LB method falls into the latter category. In Sect. 2.3 we summarise the two main categories of methods, and in Sect. 2.4 we explain where the lattice Boltzmann method fits in and give a brief overview of its advantages and disadvantages.

2.1 Conventional Navier-Stokes Solvers

Conventional numerical methods work by taking the equation (or coupled system of equations) of interest and directly solving them by a particular method of approximation. In the case of CFD, the basic equations to be solved are the continuity equation and the Navier-Stokes equation (or their incompressible counterparts). Additional equations, such as an energy equation and an equation of state, may augment these; the choice of such additional equations depends on the physics to be simulated and the approximations used.

The **derivatives** in these equations are always **discretised** in some form so that the equations may be solved approximately on a computer. One simple example is the **Euler approximation of a time derivative**. By definition, a variable's derivative is its slope over an infinitesimal interval Δt, and this can

(continued)

be approximated using a finite interval Δt as

$$\frac{\partial y(t)}{\partial t} = \lim_{\Delta t \to 0} \frac{y(t+\Delta t) - y(t)}{\Delta t} \approx \frac{y(t+\Delta t) - y(t)}{\Delta t}. \tag{2.1}$$

Unsurprisingly, the accuracy of this approximation increases as Δt is made smaller and thus closer to its infinitesimal ideal. Additionally, the accuracy depends on the solution itself; as a rule of thumb, a rapidly varying solution requires a smaller value of Δt to reach a good level of accuracy than a slowly varying solution does.

Example 2.1 The *forward Euler* method can be used to find a numerical solution to simple equations. Consider the equation $\partial y(t)/\partial t = -y(t)$, with $y(0) = 1$. If we did not know already that the answer is $y(t) = \mathrm{e}^{-t}$ we might want to solve it step-by-step for discrete time steps $t_n = n\Delta t$ as

$$y_{n+1} = y_n + \Delta t \left.\frac{\partial y}{\partial t}\right|_{y=y_n} = (1 - \Delta t) y_n. \tag{2.2}$$

Here, y_n is the numerical approximation to $y(t_n)$. In this way, we would find $y_1 = (1 - \Delta t)y_0 = (1 - \Delta t)$, $y_2 = (1 - \Delta t)y_1$, and so forth. The resulting solutions for various values of Δt are shown in Fig. 2.1.

Exercise 2.1 Write a script implementing (2.2) from $t = 0$ to $t = 3$. Try out different values of Δt and show that the difference between the numerical solution $y_k|_{t_k=3}$ and the analytical solution $y(3) = \mathrm{e}^{-3}$ varies linearly with Δt.

While the forward Euler method is the simplest and fastest method to step the solution forward in time, other methods such as the implicit backward Euler

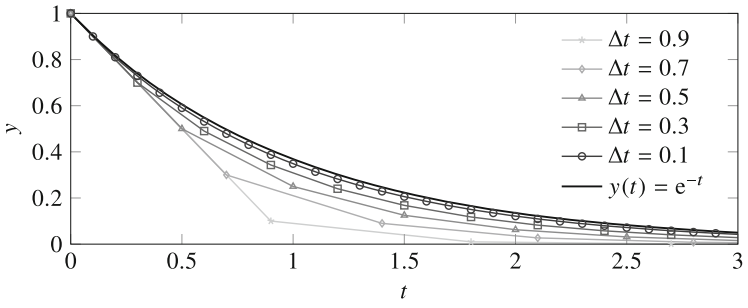

Fig. 2.1 Comparison of the analytical solution of $\partial y(t)/\partial t = -y(t)$ to forward Euler solutions with different values of Δt

method or Runge-Kutta methods beat it in stability and/or accuracy [1].[2] Typically, conventional methods for unsteady (i.e. time-dependent) CFD can use any of these methods in order to determine the solution at the next time step from the solution at the current time step.

However, these conventional CFD methods are distinguished by the approach they use to *discretise* the solution, i.e. how they use a finite set of numbers to represent the solution in continuous physical space. All these methods must represent the solution variables, such as fluid velocity $\boldsymbol{u}$ and pressure p, in such a way that their spatial derivatives can be found throughout the entire domain.

For many if not most conventional methods, this process of discretisation leads to matrix equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A}$ is a sparse matrix that relates the unknown discretised solution variables in the vector $\boldsymbol{x}$, and $\boldsymbol{b}$ represents the influence of boundary conditions and source terms. Solving such matrix equations by inverting $\boldsymbol{A}$ to find $\boldsymbol{x}$ is a linear algebra problem that lies at the heart of these methods, and finding efficient solution methods for such problems has been the topic of much research. Another common challenge of conventional incompressible Navier-Stokes solvers is to obtain a solution for the pressure Poisson equation.

In the following sections we will take a brief look at the basics of some of these methods, namely the finite difference, finite volume, and finite element methods. We will not cover the boundary-element method (BEM) [2], which is often used for creeping flows in complex geometries, or spectral methods for fluid dynamics [3].

2.1.1 Finite Difference Method

In the finite difference (FD) method, physical space is divided into a regular grid of nodes. In one dimension, these nodes are placed at the position $x_j = j\Delta x$. On each of these nodes, the solution variables are represented by a number; for a general quantity $\lambda(x)$, the exact solution $\lambda(x_j)$ is approximated by a discretised counterpart, denoted as λ_j.

2.1.1.1 Finite Difference Approximations of Derivatives

At the base of the finite difference method, derivatives of λ are approximated by linear combinations ("finite differences") of λ_j. To find these differences, we

[2] Stability and accuracy, especially in terms of the lattice Boltzmann method, are later covered in more detail in Sects. 4.4 and 4.5, respectively.

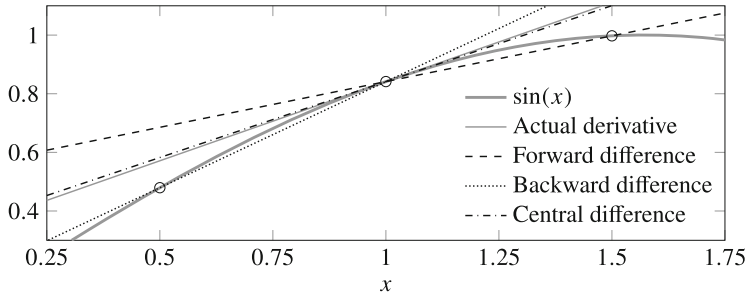

Fig. 2.2 Approximations of the derivative of $\sin(x)$ at $x = 1$, with $\Delta x = 0.5$

consider the Taylor series of $\lambda(x)$ about x_j:

$$\begin{aligned}\lambda(x_j + n\Delta x) &= \lambda(x_j) + (n\Delta x)\frac{\partial \lambda(x_j)}{\partial x} + \frac{(n\Delta x)^2}{2}\frac{\partial^2 \lambda(x_j)}{\partial x^2} + \dots \\ &= \sum_{m=0}^{\infty} \frac{(n\Delta x)^m}{m!}\frac{\partial^m \lambda(x_j)}{\partial x^m}.\end{aligned} \tag{2.3}$$

From this we can find three simple approximations for the first-order derivative,

$$\left.\frac{\partial \lambda}{\partial x}\right|_{x_j} \approx \frac{\lambda_{j+1} - \lambda_j}{\Delta x}, \quad \left.\frac{\partial \lambda}{\partial x}\right|_{x_j} \approx \frac{\lambda_{j+1} - \lambda_{j-1}}{2\Delta x}, \quad \left.\frac{\partial \lambda}{\partial x}\right|_{x_j} \approx \frac{\lambda_j - \lambda_{j-1}}{\Delta x}. \tag{2.4}$$

These three approximations are called the *forward* difference,[3] the *central* difference, and the *backward* difference approximations, respectively.

Exercise 2.2 Prove that the four approximations in (2.4) are valid by letting e.g. $\lambda_{j+1} \to \lambda(x_{j+1})$ and inserting (2.3), and show that the truncation error of the forward and backward difference approximations are $O(\Delta x)$ while that of the central difference approximation is $O(\Delta x^2)$.

The comparison in Fig. 2.2 indicates that central differences approximate the first derivative better, which is typically true and which can also be seen from its smaller $O(\Delta x^2)$ truncation error.

We can also find an approximation for the second-order derivative with a $O(\Delta x^2)$ truncation error:

$$\left.\frac{\partial^2 \lambda}{\partial x^2}\right|_{x_j} \approx \frac{\lambda_{j+1} - 2\lambda_j + \lambda_{j-1}}{\Delta x}. \tag{2.5}$$

[3]The forward difference approximation corresponds to the forward Euler approximation for time discretisation, shown in (2.1).

From any such given finite difference scheme, it is possible to insert the Taylor expansion in order to determine not only what the scheme approximates, but also the truncation error of the approximation. This is detailed further in Sect. 4.5.1.

Example 2.2 A finite difference approximation of the heat equation $\partial T/\partial t = \kappa \partial^2 T/\partial x^2$, where $T(x,t)$ is the temperature and κ is the thermal diffusivity, is

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = \kappa \frac{T_{j+1}^n - 2T_j^n + T_{j-1}^n}{\Delta x^2}. \tag{2.6}$$

Here the superscripts indicate the time step and the subscripts the spatial position, e.g. $T(x_j, t_n) \approx T_j^n$. We have used the forward Euler approximation from (2.1) to discretise the time derivative and (2.5) for the spatial second derivative. If we know the value of the solution at every point x_j at time t_n, along with the values at the edges of the system at all times, we can from these values determine the temperature at t_{n+1} for every point.

2.1.1.2 Finite Difference Methods for CFD

The finite difference method is simple in principle; just take a set of equations and replace the derivatives by finite difference approximations. However, this simple approach is often not sufficient in practice, and special techniques may be required for the set of equations in question. We will now touch on some problems and techniques of finding FD solutions of the Navier-Stokes equation, all of which are covered in more depth in the straightforward finite difference CFD book by Patankar [4].

We found above that the central difference scheme for first derivatives is typically more accurate than forward or backward schemes. However, in the advection term $\partial(\rho u_\alpha u_\beta)/\partial x_\beta$, information comes only from the opposite direction of the fluid flow, i.e. *upstream* or *upwind*.[4] Since the central difference scheme looks both upwind and downwind, it is possible to improve on it by using an *upwind scheme*, where either a forward or a backward scheme is used depending on the direction of fluid flow.

An issue requiring special treatment is the problem of *checkerboard instabilities*, where patterns of alternatingly high and low values emerge, patterns which in 2D are reminiscent of the black-and-white pattern on a checkerboard. A 1D example is shown in Fig. 2.3. In short, the reason behind this pattern is that a central difference scheme would report the first derivative as being zero, so that the rapidly varying field is felt as being uniform. Thus, there is nothing to stop the pattern from emerging.

A remedy to this problem is using a staggered grid as shown in Fig. 2.4, where different grids of nodes are used for different variables. These different grids are

[4] As a practical example, a deer can smell a hunter who is upwind of it, since the wind blows the hunter's scent towards the deer.

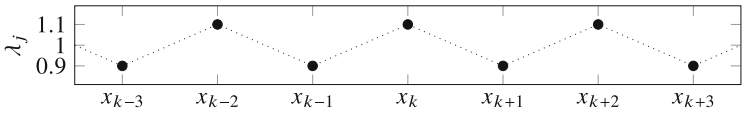

Fig. 2.3 A one-dimensional "checkerboard" field around x_k

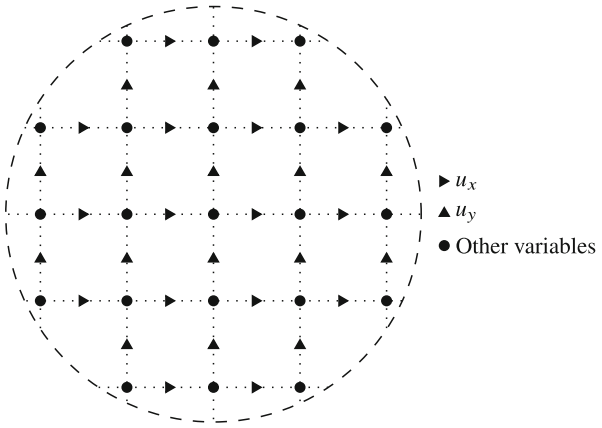

Fig. 2.4 A cutout of a staggered grid, where u_x and u_y are each stored in their own shifted grid of nodes

shifted relative to each other. Thus, when evaluating e.g. $\partial u_x/\partial x$ in one of the p nodes or $\partial p/\partial x$ in one of the u_x nodes, we can use a central difference scheme where only *adjacent* nodes are used, instead of having to skip a central node like the central difference scheme in (2.4) implies. Thus, a field like that shown in Fig. 2.3 is no longer felt as being uniform, and checkerboard instabilities cannot emerge.

The Navier-Stokes equation is nonlinear because of its advection term. Nonlinear equations are typically handled by iterating a series of "guesses" for the nonlinear quantity. This is additionally complicated by having to couple a simultaneous set of equations. In the classic FD algorithms for incompressible flow called SIMPLE and SIMPLER, guesses for the pressure field and the velocity field are coupled and successively iterated using equations tailored for the purpose. More information on these somewhat complex algorithms can be found elsewhere [4].

2.1.1.3 Advantages and Disadvantages

The crowning **advantage** of the finite difference method is that it is really quite **simple** in principle. For a number of simple equations it is not that much

(continued)

more difficult in practice, though some care must be taken in order to maintain stability and consistency [1].

However, fluids are governed by a complex set of coupled equations that contain several variables. Therefore, a number of special techniques need to be applied in order to use the FD method for CFD, which increases the amount of understanding and effort required to implement a FD CFD solver. Still, the FD method can be simple and effective compared to other conventional methods [5].

There are certain numerical **weaknesses** inherent to FD CFD. Unless special care is taken, the scheme is **not conservative**, meaning that the numerical errors cause the conservation of quantities like mass, momentum, and energy to not be perfectly respected [5]. Additionally, advective FD schemes are subject to **false diffusion**, where numerical errors cause the advected quantity to be diffused even in pure-advection cases that should have no diffusion [4]. Finally, since the FD method is based on a regular grid it **has issues with complex geometries** that do not conform to the grid itself [5]. (FD on irregular grids is in principle possible, but in practice it is hardly used [5].) The latter point is possibly the most important reason why other CFD methods have become more popular.

2.1.2 Finite Volume Method

In the finite volume (FV) method, space does not need to be divided into a *regular* grid. Instead, we subdivide the simulated volume V into many smaller *volumes* V_i, which may have different sizes and shapes to each other.[5] This allows for a better representation of complex geometries than e.g. the finite difference method, as illustrated in Fig. 2.5. In the middle of each finite volume V_i, there is a node where each solution variable $\lambda(x)$ is represented by its approximate average value $\bar{\lambda}_i$ within that volume.

2.1.2.1 Finite Volume Approximation of Conservation Equations

The FV method is not as general as the FD method which can in principle be used for any equation. Rather, the FV method is designed to solve *conservation equations*,

[5] We here use the term "volume" in a general sense, where a 2D volume is an area and a 1D volume is a line segment.

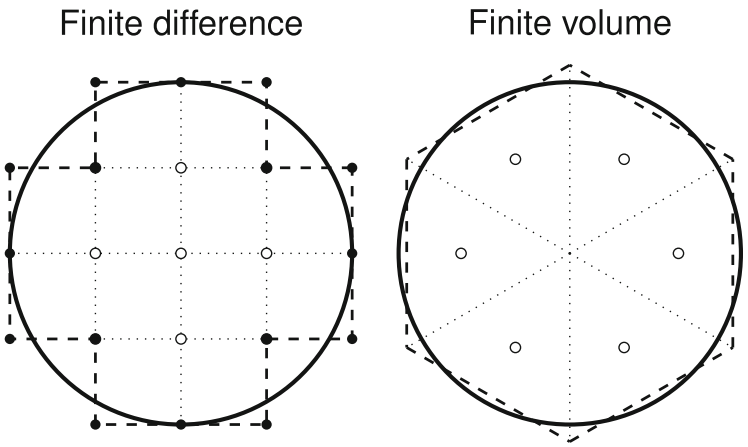

Fig. 2.5 Simple finite difference and finite volume discretisations of the volume inside a circular surface. The effective surface in each case is shown as *black dashed lines*, and interior nodes as *white circles*. To the right, the *dotted lines* show the finite volumes' interior edges

the type of equations which we typically find in e.g. fluid mechanics.[6] The FV method is *conservative* by design, which means that e.g. mass and momentum will always be conserved perfectly, unlike in the FD method.

To show the general principle of how FV approximates a conservation equation, we start with a steady advection-diffusion equation for a general quantity $\lambda(\boldsymbol{x}, t)$,

$$\nabla \cdot \left(\rho \lambda \boldsymbol{u}\right) = \nabla \cdot (D \nabla \lambda) + Q, \tag{2.7}$$

where the density ρ and the flow field $\boldsymbol{u}$ are assumed known, D is a diffusion coefficient for λ, and Q is a source term. By integrating this equation over the entire volume V and applying the divergence theorem, we get

$$\int_S (\rho \lambda \boldsymbol{u}) \cdot \mathrm{d}\boldsymbol{A} = \int_S (D \nabla \lambda) \cdot \mathrm{d}\boldsymbol{A} + \int_V Q \, \mathrm{d}V, \tag{2.8}$$

where S is the surface of the volume V and $\mathrm{d}\boldsymbol{A}$ is an infinitesimal surface normal element. The concept of the divergence theorem is as central to the FV method as it is for conservation equations in general: Sources and sinks of a quantity within a volume are balanced by that quantity's flux across the volume's boundaries.

[6] That is not to say that the FV method is limited to conservation equations; it can also be used to solve more general hyperbolic problems [6].

This integral over the entire volume can be split up as a sum of integrals over the finite volumes V_i and their surfaces S_i,[7] and each such integral can be written as

$$\sum_{s_j \in S_i} \left[\int_{s_j} (\rho \lambda \boldsymbol{u}) \cdot \mathrm{d}\boldsymbol{A} \right] = \sum_{s_j \in S_i} \left[\int_{s_j} (D \nabla \lambda) \cdot \mathrm{d}\boldsymbol{A} \right] + V_i \bar{Q}_i. \tag{2.9}$$

Here, we have additionally split up the integrals over the surface S_i as a sum over its component surface segments s_j,[8] and the volume integral is replaced with the integrand's average value $\bar{Q}_i$ times the volume V_i.

Equation (2.9) is still exact; no approximations have been made as long as the finite volumes $\{V_i\}$ together perfectly fill out the total volume V. However, as λ and $\nabla\lambda$ are not known on the surfaces S_j, the surface integrals must be related to the volume averages $\bar{\lambda}_i$. Using linear interpolation, this can be done in a simple way that leads to second-order accuracy [5]. Starting with the values of $\bar{\lambda}_i$ of the two volumes adjacent to the surface s_j, λ can be linearly interpolated between the two volumes' nodes so that each node point $\boldsymbol{x}_i$ has its corresponding volume's value of $\lambda(\boldsymbol{x}_i) = \bar{\lambda}_i$. At the point where the straight line between the two nodes crosses the surface s_j, we can find the linearly interpolated values of λ and $(\nabla\lambda) \cdot \mathrm{d}\boldsymbol{A}$. These values can then be applied to the entire surface in the surface integral.

Higher-order accuracy can be achieved by estimating the values of λ and $\nabla\lambda$ at more points on the surface, such as the surface edges which can be determined by interpolation from all the adjacent volumes [5]. Additionally, the interpolation of values on the surface may use node values from further-away volumes [5, 7].

2.1.2.2 Finite Volume Methods for CFD

While the basic formulations of finite volume and finite difference methods are different, CFD using FV methods bear many similarities to finite difference CFD, which is discussed in Sect. 2.1.1.2. For instance, for higher-order interpolation schemes, it is still generally a good idea to use more points in the upwind direction than in the downwind direction [5, 7]. Additionally, the iterative finite difference SIMPLE and SIMPLER schemes for CFD [4] and their descendants may also be adapted for finite volume simulations [7].

One difference is that the staggered grids generally used in FD CFD become too cumbersome to use for the irregular volumes typically used in FV CFD. While the issue of checkerboard instabilities is also present in the FV method for non-staggered grids, this is dealt with by the use of schemes that use more than two node values to approximate the first derivative at a point [7].

[7] For the internal surfaces between adjacent finite volumes, the surface integrals from the two volumes will cancel each other.

[8] In Fig. 2.5, S_i is the triangular surface around each volume, and s_j represents the straight-line faces of these triangles.

2.1.2.3 Advantages and Disadvantages

While finite volume methods are formulated differently to finite difference methods, the two methods are comparable in their relative simplicity. The FV method has some additional **advantages**, however. The control volume formulation makes it fundamentally **conservative**; e.g. mass and momentum will be conserved throughout the entire domain in a closed system. Additionally, the FV method is very appropriate for use with irregular grids, which means that **complex geometries can be captured well** (the grid is adapted to the geometry), and it is straightforward to "spend" more resolution on critical regions in the simulation by making the grid finer in these regions.

The **downside** of irregular grids is that making **appropriate grids for complex geometries** is itself a **fairly complex** problem; indeed, it is an entire field of study by itself. Additionally, **higher-order FV methods are not straightforward** to deal with, in particular in three dimensions and for irregular grids [5]. While FV is not as general a method as FD in terms of what equations it can solve, this is typically not an issue for the equations encountered in CFD.

2.1.3 Finite Element Methods

In finite element methods (FEM), PDEs are solved using an integral form known as the *weak form*, where the PDE itself is multiplied with a weight function $w(x)$ and integrated over the domain of interest. For example, the *Helmholtz equation* $\nabla^2\lambda + k^2\lambda = 0$ (a steady-state wave equation for wavenumber k, further explained in Sect. 12.1.4) in 1D becomes

$$\int w(x)\frac{\partial^2\lambda(x)}{\partial x^2}\,\mathrm{d}x + k^2\int w(x)\lambda(x)\,\mathrm{d}x = 0. \tag{2.10}$$

Generally, an unstructured grid can be used with FEM, with a discretised solution variable λ_i represented at each grid corner node $\boldsymbol{x}_i$. Between the grid corners, the variable $\lambda(\boldsymbol{x})$ is interpolated using *basis functions* $\phi_i(\boldsymbol{x})$ fulfilling certain conditions, i.e.

$$\lambda(x) \approx \sum_i \lambda_i\phi_i(x), \quad \text{for } \{\phi_i\} \text{ such that } \quad \lambda(x_i) = \lambda_i, \quad \sum_i \phi_i(x) = 1 \tag{2.11}$$

in our 1D example. The simplest 1D basis functions are linear functions such that $\phi_i(x_i) = 1, \phi_i(x_{j\neq i}) = 0$, and are non-zero only in the interval (x_{i-1}, x_{i+1}). However, a large variety of basis functions that are not linear (e.g. quadratic and cubic ones) are also available, and the order of accuracy is typically tied to the order of the basis functions.

Usually, the basis functions themselves are chosen as weighting functions, $w(x) = \phi_i(x)$. This leads to a system of equations, one for each unknown value λ_i. Through the integrals, each value of λ_i in our 1D example is related with λ_{i-1} and λ_{i+1}, assuming linear basis functions.

The main advantage of FEM is that it is mathematically well-equipped for **unstructured grids** and for increasing the order of accuracy through **higher-order** basis functions (though these also require more unknowns λ_i). These grids can be dynamically altered to compensate for **moving geometry**, as in the case of simulating a car crash. One **disadvantage** of FEM is that, like FD methods, it is **not conservative** by default like FV methods are. Another disadvantage is its **complexity** compared to FD and FV methods. For instance, the integrals become tricky to solve on general unstructured grids. And as with FD and FV methods, solving the complex **Navier-Stokes** system of equations is **not straightforward**; see e.g. [8] for more on CFD with the FEM. The **checkerboard** instabilities described in Sect. 2.1.1.2 may appear here also unless special care is taken to deal with these [9].

2.2 Particle-Based Solvers

Particle-based solvers are not based on directly discretising the equations of fluid mechanics, and they thus take an approach distinctly different to that of the conventional solvers of the previous section. Instead, these methods represent the fluid using *particles*. Depending on the method, a particle may represent e.g. an atom, a molecule, a collection of molecules, or a portion of the macroscopic fluid. Thus, while conventional Navier-Stokes solvers take an entirely macroscopic view of a fluid, particle-based methods usually take a microscopic or mesoscopic view.

In this section we briefly present six different particle-based methods, ordered roughly from microscopic, via mesoscopic, to macroscopic. Methods that are related to or viable alternatives to the lattice Boltzmann method are described in more detail.

2.2.1 Molecular Dynamics

Molecular dynamics (MD) is at its heart a fundamentally simple microscopic method which tracks the position of particles that typically represent atoms or molecules. These particles interact through intermolecular forces $\boldsymbol{f}_{ij}(t)$ which are

chosen to reproduce the actual physical forces as closely as possible.[9] Knowing the total force $\boldsymbol{f}_i(t)$ on the ith particle from all other particles, we know its acceleration as per Newton's second law:

$$\frac{\mathrm{d}^2\boldsymbol{x}_i}{\mathrm{d}t^2} = \frac{\boldsymbol{f}_i}{m_i} = \frac{1}{m_i}\sum_{j\neq i}\boldsymbol{f}_{ij}. \tag{2.12}$$

The particle position $\boldsymbol{x}_i$ can then be updated numerically by integrating Newton's equation of motion. While there are many such *integrator* algorithms, a particularly simple and effective one is the Verlet algorithm [10],

$$\boldsymbol{x}_i(t+\Delta t) = 2\boldsymbol{x}_i(t) - \boldsymbol{x}_i(t-\Delta t) + \frac{\boldsymbol{f}_i(t)}{m_i}\Delta t^2. \tag{2.13}$$

This scheme uses the current and previous position of a particle to find its next position. The Verlet scheme can also be equivalently expressed to use the particle's velocity instead of its previous position [10].

Exercise 2.3 Using Taylor expansion as in (2.3), show that the truncation error of the Verlet algorithm in (2.13) is $O(\Delta t^4)$.

However, while MD is a great method for simulating microscale phenomena like chemical reactions, protein folding and phase changes, a numerical method that tracks individual molecules is far too detailed to be used for macroscopic phenomena—consider that a single gram of water contains over 10^{22} molecules. Therefore, MD is highly impractical as a Navier-Stokes solver, and more appropriate methods should be chosen for this application. More on MD and its applications can be found elsewhere [10–12].

2.2.2 *Lattice Gas Models*

Lattice gas models were first introduced in 1973 by Hardy, Pomeau, and de Pazzis as an extremely simple model of 2D gas dynamics [13]. Their particular model was subsequently named the HPP model after its inventors. In this model, fictitious particles exist on a square lattice where they stream forwards and collide in a manner that respects conservation of mass and momentum, much in the same way as molecules in a real gas. As the HPP lattice was square, each node had four neighbours and each particle had one of the four possible velocities $\boldsymbol{c}_i$ that would bring a particle to a neighbouring node in one time step.

[9] This straightforward force approach scales with the number N of particles as $O(N^2)$, though more efficient approaches that scale as $O(N)$ also exist [10].

However, it was not until 1986 that Frisch, Hasslacher, and Pomeau published a lattice gas model that could actually be used to simulate fluids [14]. Their model was also named after its inventors as the FHP model. The difference to the original HPP model is small, but significant: Instead of the square lattice and four velocities of the HPP model, the FHP model had a triangular lattice and six velocities $\boldsymbol{c}_i$. This change turned out to give the model sufficient lattice isotropy to perform fluid simulations [15–17].

Lattice gas models are especially interesting in the context of the lattice Boltzmann method, as **the LBM grew out of lattice gas models**. Indeed, early LB articles are difficult to read without knowledge of lattice gases, as these articles use much of the same formalism and methods.

2.2.2.1 Algorithm

Only up to one particle of a certain velocity can be present in a node at any time. Whether or not a particle of velocity $\boldsymbol{c}_i$ exists at the lattice node at $\boldsymbol{x}$ at time t is expressed by the *occupation number* $n_i(\boldsymbol{x}, t)$, where the index i refers to the velocity $\boldsymbol{c}_i$. This occupation number n_i is a Boolean variable with possible values of 0 and 1, representing the absence and presence of a particle, respectively.

This occupation number can be directly used to determine macroscopic observables: The mass density and momentum density in a node can be expressed as [16]

$$\rho(\boldsymbol{x}, t) = \frac{m}{v_0} \sum_i n_i(\boldsymbol{x}, t), \qquad \rho \boldsymbol{u}(\boldsymbol{x}, t) = \frac{m}{v_0} \sum_i \boldsymbol{c}_i n_i(\boldsymbol{x}, t), \tag{2.14}$$

respectively, where m is the mass of a particle and v_0 is the volume covered by the node.

There are two rules that determine the time evolution of a lattice gas. The first rule is *collision*, where particles that meet in a node may be redistributed in a way that conserves the mass and momentum in the node. Generally, collisions can be mathematically expressed as

$$n_i^\star(\boldsymbol{x}, t) = n_i(\boldsymbol{x}, t) + \Omega_i(\boldsymbol{x}, t), \tag{2.15}$$

where $n_i^\star$ is the post-collision occupation number and $\Omega_i \in \{-1, 0, 1\}$ is a collision operator that may redistribute particles in a node, based on all occupation numbers $\{n_i\}$ in that node [15]. Collisions must conserve mass ($\sum_i \Omega_i(\boldsymbol{x}, t) = 0$) and momentum ($\sum_i \boldsymbol{c}_i \Omega_i(\boldsymbol{x}, t) = \boldsymbol{0}$).

Which collisions may occur (i.e. the dependence of Ω_i on $\{n_i\}$) varies between different formulations of lattice gases, but in any case this is cumbersome to express mathematically [16, 17]. Rather, we represent graphically the two types of collisions

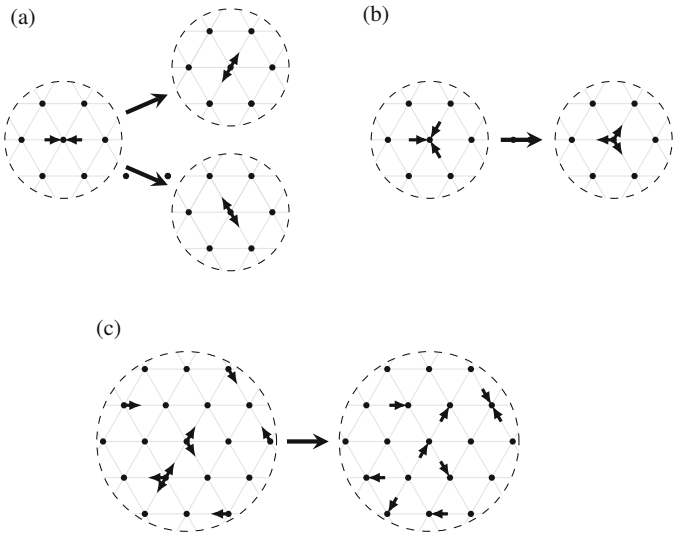

Fig. 2.6 Rules of the original FHP lattice gas model: collision and streaming. (**a**) Two-particle collision; the resolution is chosen randomly from the two options. (**b**) Three-particle collision. (**c**) Streaming

in the original FHP model [14] in Fig. 2.6. Figure 2.6a shows the two possible resolutions between head-on collisions of two particles, which are chosen randomly with equal probability. Figure 2.6b shows the resolution of a three-particle collision: When three particles meet with equal angles between each other, they are turned back to where they came from.

Exercise 2.4

(a) Show that the macroscopic quantities of (2.14) are preserved by these collisions.
(b) Show that there is another possible resolution for both the two-particle collision and the three-particle collision.

The second rule of a lattice gas is *streaming*: after collisions, particles move from their current node to a neighbouring node in their direction of velocity, as shown in Fig. 2.6c. The particle velocities $\boldsymbol{c}_i$ are such that particles move exactly from one node to another from one time step to the next. For the FHP model, which has six velocities of equal magnitude, we have $|\boldsymbol{c}_i| = \Delta x/\Delta t$, Δx being the distance between nodes and Δt being the time step. Thus, the streaming can be expressed mathematically as

$$n_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta t) = n_i^\star(\boldsymbol{x}, t). \tag{2.16}$$

Both rules can be combined into a single equation:

$$n_i(\boldsymbol{x} + \boldsymbol{c}_i \Delta t, t + \Delta t) = n_i(\boldsymbol{x}, t) + \Omega_i(\boldsymbol{x}, t). \tag{2.17}$$

In addition to the HPP and FHP models, there is a number of more complex lattice gas models. Their various features include rest particles with zero velocity, additional collisions, and additional higher-speed particles [16, 17]. However, all of these can be expressed mathematically through (2.17); the difference between them lies in the velocities $\boldsymbol{c}_i$ and the rules of the collision operator Ω_i. All of these models which fulfil certain requirements on lattice isotropy (e.g., FHP fulfils them while HPP does not) can be used for fluid simulations [17].

2.2.2.2 Advantages and Disadvantages

One of the touted advantages of lattice gas models was that the occupation numbers n_i are Boolean variables (particles are either *there* or *not there*), so that collisions are in a sense perfect: The roundoff error inherent in the floating-point operations used in other CFD methods do not affect lattice gas models [15]. Additionally, lattice gases can be massively parallelised [15].

However, a downside of these collisions is that they get out of hand for larger number of velocities. For example, for the three-dimensional lattice gas with 24 velocities [15, 17], there are $2^{24} \approx 16.8 \times 10^6$ possible states in a node. The resolution of any collision in this model was typically determined by lookup in a huge table made by a dedicated program [15].

The FHP model additionally has problems with isotropy of the Navier-Stokes equations, which can only disappear in the limit of low Mach numbers, i.e. for a quasi-incompressible flow [15]. Additionally, lattice gases struggled to reach as high Reynolds numbers as comparable CFD methods [15].

The major issue with lattice gases, however, was **statistical noise**. Like real gases, lattice gases are teeming with activity at the microscopic level. Even for a gas at equilibrium, when we make a control volume smaller and smaller, the density (mass per volume) inside it will fluctuate more and more strongly with time: Molecules continually move in and out, and the law of large numbers applies less for smaller volumes. This is also the case with lattice gases, where the macroscopic values from (2.14) will fluctuate even for a lattice gas at equilibrium.

In one sense, it may be an advantage that lattice gases can qualitatively capture the thermal fluctuations of a real gas [16]. But if the goal is to simulate a macroscopic fluid, these fluctuations are a nuisance. For that reason, lattice gas simulations would typically report density and fluid velocity found through averaging in space and/or time (i.e. over several neighbouring nodes and/or several adjacent time steps), and even averaging over multiple ensembles (i.e. macroscopically similar but microscopically different realisations of the system) [16], though this could only reduce the problem and some noise would always remain.

The problem of statistical noise was more completely dealt with by the invention of the lattice Boltzmann method in the late 1980s [18–20]. This method was first introduced by tracking the occupation number's expectation value $f_i = \langle n_i \rangle$ rather than the occupation number itself, thus eliminating the statistical noise. This was the original method of deriving the LBM, and it was not fully understood how to derive it from the kinetic theory of gases presented in Sect. 1.3 until the mid-90s [21]. This more modern approach of derivation is the one that we will follow in Chap. 3.

2.2.3 Dissipative Particle Dynamics

Dissipative particle dynamics (DPD) is, like the LBM, a relatively new mesoscopic method for fluid flows. Originally proposed by Hoogerbrugge and Koelman [22] in 1992, it was later put on a proper statistical mechanical basis [23]. DPD can be considered a coarse-grained MD method that allows for the simulation of larger length and time scales than molecular dynamics (cf. Sect. 2.2.1) and avoids the lattice-related artefacts of lattice gases. Being a fully Lagrangian scheme without an underlying lattice, DPD is intrinsically Galilean invariant and isotropic.

In the following, we will summarise the essential ideas of DPD, as described in a recent review article by Liu et al. [24]. We will not cover smoothed dissipative particle dynamics (SDPD) [25] that is a special case of smoothed-particle hydrodynamics rather than an extension of DPD.

The basis of DPD are particles of mass m that represent clusters of molecules. These particles interact *via* three different forces: conservative (C), dissipative (D) and random (R). Unlike forces in MD, the conservative forces in DPD are soft, which allows larger time steps. The dissipative forces mimic viscous friction in the fluid, while random forces act as thermostat. All these forces describe additive pair interactions between particles (obeying Newton's third law), hence DPD conserves momentum. In fact, DPD is often referred to as a momentum-conserving thermostat for MD. The total force on particle i can be written as sum of all forces due to the presence of other particles j and external forces $\boldsymbol{f}^{\mathrm{ext}}$:

$$\boldsymbol{f}_i = \boldsymbol{f}^{\mathrm{ext}} + \sum_{j \neq i} \boldsymbol{f}_{ij} = \boldsymbol{f}^{\mathrm{ext}} + \sum_{j \neq i} \left(\boldsymbol{f}_{ij}^{\mathrm{C}} + \boldsymbol{f}_{ij}^{\mathrm{D}} + \boldsymbol{f}_{ij}^{\mathrm{R}} \right). \tag{2.18}$$

All interactions have a finite radial range with a cutoff radius r_{c}. Details of the radial dependence of the forces are discussed in [24]. Like in MD, a crucial aspect of the DPD algorithm is the time integration of the particle positions and velocities. Typical employed methods are a modified velocity-Verlet [26] or a leap-frog algorithm [27].

DPD obeys a fluctuation-dissipation theorem (if the radial weight functions are properly chosen) [23] and is particularly suited for hydrodynamics of complex fluids at the mesoscale with finite Knudsen number. Typical applications are suspended

polymers or biological cells, but also multiphase flows in complex geometries. Solid boundaries are typically modelled as frozen DPD particles, while immersed soft structures (e.g. polymers) are often described by particles connected with elastic springs. Similarly to other mesoscopic methods, it is relatively easy to include additional physics, for instance the equation of state of multiphase fluids.

A disadvantage of DPD is that it contains a large number of parameters that have to be selected carefully. The choice of the radial weight functions is delicate and affects the emergent hydrodynamic behaviour. For example, to reach a realistically large viscosity, it is necessary to increase the cutoff distance r_c which in turn leads to more expensive simulations.

2.2.4 Multi-particle Collision Dynamics

In 1999, Malevanets and Kapral [28, 29] introduced the multi-particle collision (MPC) dynamics, which has since become a popular method in the soft matter community. The paradigm of MPC is to coarse-grain the physical system as much as possible while still resolving the essential features of the underlying problem.

Although MPC is nothing more than a modification of direct simulation Monte Carlo (DSMC, cf. Sect. 2.2.5) [30], we discuss both methods separately as they are normally used for completely different applications. In particular, MPC is commonly employed for systems with a small mean free path, while DSMC allows the simulation of rarefied gases with a large mean free path.

MPC is a method of choice for complex systems where both hydrodynamic interactions and thermal fluctuations are relevant. Due to its particle-based nature, it is relatively easy to implement coupled systems of solvent and solutes. Therefore, MPC is most suitable and often employed for the modelling of colloids, polymers, vesicles and biological cells in equilibrium and external flow fields. MPC particularly shows its strengths for systems with Reynolds and Péclet numbers between 0.1 and 10 and for applications where consistent thermodynamics is required and where the macroscopic transport coefficients (viscosity, thermal diffusivity, self-diffusion coefficient) have to be accurately known [31].

There exist also MPC extensions for non-ideal [32], multicomponent [33] and viscoelastic fluids [34]. We refer to [31, 35] for thorough reviews and to [36] for a recent overview.

2.2.4.1 Algorithm

The essential features of the MPC algorithm are: (i) alternating streaming and collision steps, (ii) local conservation of mass, momentum and, unlike standard LBM schemes, energy, (iii) isotropic discretisation. The last two properties ensure that MPC can be used as a viable Navier-Stokes solver.

The basic MPC setup comprises a large number of point-like particles with mass m. These particles can either be fluid or immersed (e.g. colloidal) particles. This feature allows a treatment of solvent and solutes on an equal footing. For example, immersed particles can be directly coupled by letting them participate in the collision and streaming steps [37]. This approach has been successfully employed in numerous colloid and polymer simulations (see [31] and references therein).

During propagation, space and velocity are continuous and the particles move along straight lines for one time step Δt:

$$\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \boldsymbol{c}_i(t)\Delta t, \tag{2.19}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{c}_i$ are particle position and velocity. After propagation, particles collide. How the collision step looks like in detail depends on the chosen MPC algorithm. Generally, each particle-based algorithm with local mass and momentum conservation and an $\mathcal{H}$-theorem (analogous to that described in Sect. 1.3.6) is called a multi-particle collision algorithm.

One special case is the so-called stochastic rotation dynamics (SRD) algorithm. During collision, all particles are sorted into cells of a usually regular cubic lattice with lattice constant Δx. On average, there are N_c particles in each cell. The velocity $\boldsymbol{v}_i$ of each particle i in one cell is decomposed into the average cell velocity $\bar{\boldsymbol{v}}$ (as given by the average velocity of all particles in that cell) and the relative velocity $\delta\boldsymbol{v}_i$. The relative velocities are then rotated in space to give the post-collision velocities

$$\boldsymbol{v}_i^\star = \bar{\boldsymbol{v}} + \boldsymbol{R}\delta\boldsymbol{v}_i \tag{2.20}$$

where $\boldsymbol{R}$ is a suitable rotation matrix. In 2D, velocities are rotated by $\pm\alpha$ where α is a fixed angle and the sign is randomly chosen. In 3D, the rotation is defined by a fixed angle α and a random rotation axis. Rotations are the same for all particles in a given cell but statistically independent for different cells. Apart from this rotation, there is no direct interaction between particles. In particular, particles can penetrate each other, which makes a collision detection unnecessary. It can be shown that the resulting equilibrium velocity distribution is Maxwellian.

It should be noted that the originally proposed SRD algorithm [28, 29] violated Galilean invariance. This problem, which was particularly important for small time steps (i.e. a small mean free path), could be corrected by shifting the lattice by a random distance $d \in [-\Delta x/2, +\Delta x/2]$ before each collision step [38]. Furthermore, SRD does not generally conserve angular momentum; a problem that can be avoided as reviewed in [31].

Other collision models than SRD are available. For example, the Anderson thermostat (MPC-AT) [30, 39] is used to produce new particle velocities according to the canonical ensemble rather than merely rotating the existing velocity vectors in space. As noted earlier, DSMC is another MPC-like method that only differs in terms of the particle collisions.

It is also possible to implement a repulsion force between colloids and solvent particles [29] in order to keep the fluid outside the colloids. This, however, requires relatively large repulsion forces and therefore small time steps. Additional coupling approaches are reviewed in [31]. Slip [40] and no-slip boundary conditions [36] are available as well.

2.2.4.2 Advantages and Disadvantages

All MPC algorithms locally conserve mass and momentum[10] and have an $\mathcal{H}$-theorem which makes them unconditionally stable [28]. Due to its locality the MPC algorithm is straightforward to implement and to use, computationally efficient and easy to parallelise. MPC has been successfully ported to GPUs with a performance gain of up to two orders of magnitude [36]. But due to its strong artificial compressibility, MPC is not well suited for the simulation of Stokes flow ($\mathrm{Re} \to 0$) or compressible hydrodynamics [31].

Both hydrodynamics and thermal fluctuations are consistently taken into account. For example, interfacial fluctuations in binary fluids are accurately captured. The hydrodynamic interactions can be switched off [41], which makes it possible to study their relevance. However, it is recommended to use other methods like Langevin or Brownian Dynamics if hydrodynamics is not desired [31]. When hydrodynamics is included, it allows for larger time steps than in MD-like methods. Therefore longer time intervals can be simulated with MPC [31].

Compared to LBM, MPC naturally provides thermal fluctuations, which can be an advantage. Yet, for systems where those fluctuations are not required or even distracting, MPC is not an ideal numerical method. Conventional Navier-Stokes solvers or the LBM are more favorable in those situations [31].

As MPC is a particle-based method, immersed objects such as colloids or polymers can be implemented in a relatively straightforward fashion. This makes MPC particularly suitable for the simulation of soft matter systems. Additionally, the transport coefficients (viscosity, thermal diffusivity, self-diffusion coefficient) can be accurately predicted as function of the simulation parameters [31, 38, 42–44]. On the other hand, it is not a simple task to impose hydrodynamic boundary conditions, especially for the pressure. Furthermore, the discussions in [36] show that no-slip boundary conditions and forcing are not as well under control as for LBM. Multicomponent fluids and also surfactants can be simulated within the MPC framework [33]. However, the LBM seems to be more mature due to the larger number of works published.

[10] Most of them also conserve energy.

2.2.5 *Direct Simulation Monte Carlo*

The direct simulation Monte Carlo (DSMC) method was pioneered by Bird [45] in the 1960s. While its initial popularity was slowed down by limitations in computer technology, DSMC is currently considered a primary method to solve realistic problems in high Knudsen number flows. Typical applications range from spacecraft technology to microsystems. More details on DSMC can be found in dedicated books [46, 47] and review articles [48–50].

DSMC is a particle method based on kinetic theory, where flow solutions are obtained through the collisions of model particles. MPC (cf. Sect. 2.2.4) can be considered DSMC with a modified particle collision procedure [30]. Since DSMC is more than 30 years older than MPC and used for different applications, we provide a short independent overview of DSMC here.

Rather than seeking solutions of the governing mathematical model, e.g. the Boltzmann equation, DSMC incorporates the physics of the problem directly into the simulation procedure. Although this change in paradigm at first raised doubts on whether DSMC solutions were indeed solutions of the Boltzmann equation [51], modern studies have shown that the DSMC method is a sound physical simulation model capable of describing physical effects, even beyond the Boltzmann formulation [49, 50].

Fundamentally, DSMC simulations track a large number of statistically representative particles. While the position and velocity of particles is resolved deterministically during motion, particle collisions are approximated by probabilistic, phenomenological models. These models enforce conservation of mass, momentum and energy to machine accuracy.

The DSMC algorithm has four primary steps [49]:

1. Move particles, complying with the prescribed boundary conditions.
2. Index and cross-reference the particles. The particles are labelled inside the computational domain as a pre-requisite for the next two steps.
3. Simulate collisions. Pairs of particles are randomly selected to collide according to a given collisional model. (It is this probabilistic process of determining collisions that sets DSMC apart from deterministic simulation procedures such as MD.) This separates DSMC from MPC; the latter handling the collision of *all* particles in a cell simultaneously.
4. Sample the flow field. Within predefined cells, compute macroscopic quantities based on the positions and velocities of particles in the cell. This averaged data is typically not necessary for the rest of the simulation and is only used as output information.

DSMC is an explicit and time-marching technique. Depending on the sample size and the averaging procedure, it may produce statistically accurate results. The statistical error in a DSMC solution is inversely proportional to $\sqrt{N}$, N being the number of simulation particles [49]. On the other hand, the computational cost is proportional to N. The main drawback of DSMC is therefore the high computational

demand of problems requiring a large N. This explains why DSMC is mostly used for dilute gases, where its accuracy/efficiency characteristics have no competitor. Nevertheless, the continuous improvement in computational resources is expected to expand the range of physical applications for DSMC simulations in the near future.

2.2.6 Smoothed-Particle Hydrodynamics

SPH was invented in the 1970s to deal with the particular challenges of 3D astrophysics. Since then it has been used in a large number of applications, in recent years also computer graphics where it can simulate convincing fluid flow relatively cheaply. Several books have been written on SPH, such as a mathematically rigorous introduction by Violeau [52] and a more practical introduction by Liu and Liu [53] on which the rest of our description will be based.

At the base of SPH is an interpolation scheme which uses point particles that influence their vicinity. For instance, any quantity λ at point $\boldsymbol{x}$ can be approximated as a sum over all particles, with each particle j positioned at $\boldsymbol{x}_j$:

$$\lambda(\boldsymbol{x}) = \sum_j \frac{m_j}{\rho_j} \lambda_j W(|\boldsymbol{x} - \boldsymbol{x}_j|, h_j). \tag{2.21}$$

Here, m_j is the particle's mass, ρ_j is the density at $\boldsymbol{x}_j$, λ_j is the particle's value of λ, and $W(|\boldsymbol{x} - \boldsymbol{x}_j|, h_j)$ is a kernel function with characteristic size h_j which defines the region of influence of particle j.[11] Thus, SPH particles can be seen as overlapping blobs, and the sum of these blobs at $\boldsymbol{x}$ determines $\lambda(\boldsymbol{x})$. For instance, the density ρ_i at particle i can be found by setting $\lambda(\boldsymbol{x}_i) = \rho(\boldsymbol{x}_i)$, so that

$$\rho_i = \sum_j \frac{m_j}{\rho_j} \rho_j W(|\boldsymbol{x}_i - \boldsymbol{x}_j|, h_j) = \sum_j m_j W(|\boldsymbol{x}_i - \boldsymbol{x}_j|, h_j). \tag{2.22}$$

The formulation of SPH and its adaptive resolution gives it a great advantage when dealing with large unbounded domains with huge density variations, such as in astrophysics. It can also deal with extreme problems with large deformations, such as explosions and high-velocity impacts, where more traditional methods may struggle. The particle formulation of SPH also allows for perfect conservation of mass and momentum.

On the downside, SPH has problems with accuracy, and it is not quite simple to deal with boundary conditions. Additionally, the formulation of SPH makes it

[11] While the kernel function can be e.g. Gaussian, it is advantageous to choose kernels that are zero for $|\boldsymbol{x} - \boldsymbol{x}_j| > h$, so that only particles in the vicinity of $\boldsymbol{x}$ need be included in the sum. Additionally, the fact that h_j can be particle-specific and varying allows adaptive resolution.

difficult to mathematically prove that the numerical method is consistent with the equations of the hydrodynamics that it is meant to simulate.

2.3 Summary

In this chapter, we have looked at various numerical methods for fluids in two main categories: *Conventional* methods, and *particle-based* methods.

Conventional methods are generic numerical methods, applied to solving the equations of fluid mechanics. These methods represent the fluid variables (such as velocity and pressure) as values at various points (nodes) throughout the domain. The interpretation of these node values varies. In the finite difference (FD) method, the idea of a continuous field is dropped in favour of a field defined only on a square grid of nodes. In the finite volume (FV) method, a node value represent the average of the fluid variables in a small volume around the node. In the finite element (FE) method, the continuous field is approximated by a kind of interpolation of the node values. These methods have in common that the node values are used to approximate the derivatives in the partial derivative equations in question.

While these general numerical methods are reasonably simple in principle (FE being somewhat more complex than FD or FV), they are complicated by the inherent difficulties of the equations of fluid mechanics. These represent a nonlinear, simultaneous system of equations where the solutions can behave in a very intricate way, especially in cases like turbulence or flows in complex geometries. Additionally, the pressure is implicit in the incompressible Navier-Stokes simulations. Such difficulties have caused the development of somewhat complex iterative algorithms such as SIMPLE and SIMPLER [4]. One troublesome issue that emerges is that of checkerboard instabilities, as described in Sect. 2.1.1.2, which can be dealt with by staggered grids or asymmetric schemes, both of which add complexity.

FD, FV, and FE simulations ultimately end up being expressed as matrix equations. Solving these equations efficiently is a pure linear algebra problem which is nevertheless important for these methods, and many different solution methods have been developed. Another complex mathematical problem that emerges when using irregular grids in FV and FE simulations is building this grid automatically for a given geometry, and this has also been a topic of extensive studies [54].

All in all, conventional methods have been thoroughly explored in the past decades, and are currently considered workhorse methods in CFD. Though it is possible to implement methods of higher order, in practice nearly all production flow solvers are second-order accurate [55].

Particle-based methods typically do not attempt to solve the equations of fluid mechanics directly, unlike conventional methods. Instead, they represent the fluid through particles, which themselves may represent atoms, molecules, collections or distributions of molecules, or portions of the macroscopic fluid. These methods are

quite varied, and are often tailored to a particular problem.[12] It is therefore difficult to give a general summary of these methods as a whole.

However, it can be said that it can be difficult to relate the dynamics of some particle-based methods, such as smoothed-particle hydrodynamics, to a macroscopic description of the fluid. This makes it difficult to quantify generally the accuracy of these methods. Additionally, it must be said that the *microscopic* particle-based CFD methods are typically not appropriate for CFD. Even the lattice gas models described in Sect. 2.2.2, which were originally intended for flow simulations, had major problems with noise from fluctuations in the microscopic particle populations. For that reason, lattice gases were largely abandoned in favour of the very similar lattice Boltzmann method, which instead took a *mesoscopic* approach that eliminated this noise.

All in all, different solvers have different advantages and disadvantages, and different types of fluid simulations pose different demands on a solver. For that reason, it is generally agreed (e.g. [5, 15, 56, 57]) that there is **no one method** which **is generally superior** to all others.

2.4 Outlook: Why Lattice Boltzmann?

While we will not describe the lattice Boltzmann method in detail until Chap. 3, we will here compare it in general terms to the other methods of this chapter.

The lattice Boltzmann method (LBM) originally grew out of the lattice gas models described in Sect. 2.2.2. While lattice gases track the behaviour of concrete particles, the LBM instead tracks the *distribution* of such particles. It can be debated whether the LBM should be called a particle-based method when it only tracks particle distributions instead of the particles themselves, but it is clear that it has much in common with many of the methods described in Sect. 2.2.

The LBM has a strong physical basis, namely the Boltzmann equation described in Sect. 1.3.4. Well-established methods exist to link its dynamics to the macroscopic conservation equations of fluids.[13] It can thus be found that the "standard" LBM is a second-order accurate solver for the weakly compressible Navier-Stokes equation; this is detailed in Sect. 4.5.5. The "weak compressibility" refers to errors that become relevant as $\mathrm{Ma} \to 1$ (cf. Sect. 4.1).

[12] For instance, molecular dynamics is tailored to simulating phenomena on an atomic and molecular level, and smoothed-particle hydrodynamics was invented to deal with the largely empty domains of astrophysical CFD.

[13] The most common such method is covered in Sect. 4.1, with a number of alternative methods referenced in Sect. 4.2.5.

The LBM gains a major advantage from being based on the Boltzmann equation rather than the equations of fluid mechanics: It becomes much simpler to implement than conventional methods. However, there is a corresponding disadvantage: understanding and adapting the LBM typically requires some knowledge about the Boltzmann equation, in addition to knowledge about fluid mechanics.

In conventional methods, much of the complexity lies in determining derivative approximations non-locally from adjacent nodes. In particular, it is difficult to discretise the non-linear advection term $\boldsymbol{u}\cdot\nabla\boldsymbol{u}$. In contrast, the detail in the LBM lies in the particle description *within* the nodes themselves, causing that *"non-linearity is local, non-locality is linear"*[14]: interactions between nodes are entirely linear, while the method's non-linearity enters in a local collision process within each node. This property makes the LBM very amenable to high-performance computing on parallel architectures, including GPUs. Coupled with the method's simplicity, this means that parallelised LB simulations can be tailor-made for a particular case more quickly than simulations using a conventional method [15].

A number of publications have compared the LBM to other methods (e.g. [15, 56–59]). From these comparisons, some takeaway messages about the LBM's advantages (+) and disadvantages (−) can be found for a number of topics:

Simplicity and Efficiency

+ For solving the incompressible Navier-Stokes equation, the LBM is similar to pseudocompressible methods, which gain simplicity and scalability by allowing artificial compressibility [56].
+ Like pseudocompressible methods, the LBM does not involve the Poisson equation [56] which can be difficult to solve due to its non-locality.
+ The heaviest computations in the LBM are local, i.e. restricted to within nodes, further improving its amenability to parallelisation [15].
− LBM is memory-intensive. Propagating populations requires a large number of memory access events. As we will see in Sect. 13.3.2, these are a major bottleneck of LB computations.
− The LBM, being inherently time-dependent, is not particularly efficient for simulating steady flows [57].

Geometry

+ The LBM is well suited to simulating mass-conserving flows in complex geometries such as porous media [15, 56, 59].
+ Moving boundaries that conserve mass can be implemented particularly well in the LBM [56], making it an attractive method for soft matter simulations [59].

Multiphase and Multicomponent Flows

+ There is a wide range of multiphase and multicomponent methods available for the LBM [56].

[14] This concise description is attributed to Sauro Succi.

+ Coupled with the LBM's advantages in complex geometries, this means that it is well suited to simulating multiphase and multicomponent flows in complex geometries [15].
− As in other lattice-based methods, there are spurious currents near fluid-fluid interfaces (cf. Sect. 9.4.1).
− According to [56], no multiphase or multicomponent methods for the LBM have capitalised well on its kinetic origins, meaning that these methods are not very different from those in conventional CFD.
− The range of viscosities and densities are somewhat limited in multiphase and multicomponent simulations [56].

Thermal Effects

+ Thermal fluctuations, which originate on the microscale but are averaged out on the macroscale, can be incorporated into the LBM mesoscopically. We will not discuss fluctuations in this book. Instead, the interested reader should refer to [60–62] for ideal and to [63, 64] for non-ideal systems with thermal fluctuations.
− Energy-conserving (thermal) simulations are not straightforward in the LBM [15, 56]. We come back to this topic in Sect. 8.4.

Sound and Compressibility

+ As the LBM is a (weakly) compressible Navier-Stokes solver, it may be well-suited for simulating situations where sound and flow interact, such as aeroacoustic sound generation [65].
− The LBM is not appropriate for directly simulating long-range propagation of sound at realistic viscosities [56, 65].
− The LBM may not be appropriate for simulating strongly compressible (i.e. transonic and supersonic) flows [15, 56].

Other Points

+ The LBM is appropriate for simulating mesoscopic physics that are hard to describe macroscopically [15].

While the lattice Boltzmann method has many advantages, it is, like all other numerical methods for fluids, not well suited for all possible applications. However, the LBM is a relatively young method, and it is still evolving at a quick pace, meaning that the range of problems to which it can be applied well is still increasing.

References

1. R.J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady State and Time Dependent Problems* (SIAM, Philadelphia, 2007)
2. C. Pozrikidis, *A Practical Guide to Boundary Element Methods with the Software Library BEMLIB* (CRC Press, Boca Raton, 2002)

3. C. Canuto, M.Y. Hussaini, A.M. Quarteroni, A. Thomas Jr, et al., *Spectral Methods in Fluid Dynamics* (Springer Science & Business Media, New York, 2012)
4. S.V. Patankar, *Numerical Heat Transfer and Fluid Flow* (Taylor & Francis, Washington, DC, 1980)
5. J.H. Ferziger, M. Peric, A. Leonard, *Computational Methods for Fluid Dynamics*, vol. 50, 3rd edn. (Springer, New York, 2002)
6. R.J. LeVeque, *Finite-Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics (Cambridge University Press, Cambridge, 2004)
7. H.K. Versteeg, W. Malalasekera, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*, 2nd edn. (Pearson Education, Upper Saddle River, 2007)
8. O.C. Zienkiewicz, R.L. Taylor, P. Nithiarasu, *The Finite Element Method for Fluid Dynamics*, 7th edn. (Butterworth-Heinemann, Oxford, 2014)
9. C.A.J. Fletcher, *Computational Techniques for Fluid Dynamics*, vol. 2 (Springer, New York, 1988)
10. D. Frenkel, B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications*, 2nd edn. Computational science series (Academic Press, San Diego, 2002)
11. M.P. Allen, D.J. Tildesley, *Computer Simulation of Liquids*. Oxford Science Publications (Clarendon Press, Oxford, 1989)
12. M. Karplus, J.A. McCammon, Nat. Struct. Biol. **9**(9), 646 (2002)
13. J. Hardy, Y. Pomeau, O. de Pazzis, J. Math. Phys. **14**(12), 1746 (1973)
14. U. Frisch, B. Hasslacher, Y. Pomeau, Phys. Rev. Lett. **56**(14), 1505 (1986)
15. S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond* (Oxford University Press, Oxford, 2001)
16. J.P. Rivet, J.P. Boon, *Lattice Gas Hydrodynamics* (Cambridge University Press, Cambridge, 2001)
17. D.A. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models* (Springer, New York, 2005)
18. G.R. McNamara, G. Zanetti, Phys. Rev. Lett. **61**(20), 2332 (1988)
19. F.J. Higuera, S. Succi, R. Benzi, Europhys. Lett. **9**(4), 345 (1989)
20. F.J. Higuera, J. Jimenez, Europhys. Lett. **9**(7), 663 (1989)
21. X. He, L.S. Luo, Phys. Rev. E **56**(6), 6811 (1997)
22. P.J. Hoogerbrugge, J.M.V.A. Koelman, Europhys. Lett. **19**(3), 155 (1992)
23. P. Español, P. Warren, Europhys. Lett. **30**(4), 191 (1995)
24. M.B. Liu, G.R. Liu, L.W. Zhou, J.Z. Chang, Arch. Computat. Methods Eng. **22**(4), 529 (2015)
25. P. Español, M. Revenga, Phys. Rev. E **67**(2), 026705 (2003)
26. R.D. Groot, P.B. Warren, J. Chem. Phys. **107**(11), 4423 (1997)
27. I. Pagonabarraga, M.H.J. Hagen, D. Frenkel, Europhys. Lett. **42**(4), 377 (1998)
28. A. Malevanets, R. Kapral, J. Chem. Phys. **110**(17), 8605 (1999)
29. A. Malevanets, R. Kapral, J. Chem. Phys. **112**(16), 7260 (2000)
30. H. Noguchi, N. Kikuchi, G. Gompper, Europhys. Lett. **78**(1), 10005 (2007)
31. G. Gompper, T. Ihle, D.M. Kroll, R.G. Winkler, in *Advanced Computer Simulation Approaches for Soft Matter Sciences III*, Advances in Polymer Science (Springer, Berlin, Heidelberg, 2008), pp. 1–87
32. T. Ihle, E. Tüzel, D.M. Kroll, Europhys. Lett. **73**(5), 664 (2006)
33. E. Tüzel, G. Pan, T. Ihle, D.M. Kroll, Europhys. Lett. **80**(4), 40010 (2007)
34. Y.G. Tao, I.O. Götze, G. Gompper, J. Chem. Phys. **128**(14), 144902 (2008)
35. R. Kapral, in *Advances in Chemical Physics*, ed. by S.A. Rice (Wiley, New York, 2008), p. 89–146
36. E. Westphal, S.P. Singh, C.C. Huang, G. Gompper, R.G. Winkler, Comput. Phys. Commun. **185**(2), 495 (2014)
37. A. Malevanets, J.M. Yeomans, Europhys. Lett. **52**(2), 231 (2000)
38. T. Ihle, D.M. Kroll, Phys. Rev. E **67**(6), 066706 (2003)
39. E. Allahyarov, G. Gompper, Phys. Rev. E **66**(3), 036702 (2002)
40. J.K. Whitmer, E. Luijten, J. Phys. Condens. Matter **22**(10), 104106 (2010)

41. M. Ripoll, R.G. Winkler, G. Gompper, Eur. Phys. J. E **23**(4), 349 (2007)
42. N. Kikuchi, C.M. Pooley, J.F. Ryder, J.M. Yeomans, J. Chem. Phys. **119**(12), 6388 (2003)
43. T. Ihle, E. Tözel, D.M. Kroll, Phys. Rev. E **72**(4), 046707 (2005)
44. C.M. Pooley, J.M. Yeomans, J. Phys. Chem. B **109**(14), 6505 (2005)
45. G.A. Bird, Phys. Fluids **6**, 1518 (1963)
46. G.A. Bird, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows* (Claredon, Oxford, 1994)
47. C. Shen, *Rarefied gas dynamics: Fundamentals, Simulations and Micro Flows* (Springer, New York, 2005)
48. G.A. Bird, Ann. Rev. Fluid Mech. **10**, 11 (1978)
49. E.S. Oran, C.K. Oh, B.Z. Cybyk, Ann. Rev. Fluid Mech. **30**, 403 (1998)
50. G.A. Bird, Comp. Math. Appl. **35**, 1 (1998)
51. W. Wagner, J. Stat. Phys. **66**, 1011 (1992)
52. D. Violeau, *Fluid Mechanics and the SPH Method: Theory and Applications*, 1st edn. (Oxford University Press, Oxford, 2012)
53. G.R. Liu, M.B. Liu, *Smoothed Particle Hydrodynamics: A Meshfree Particle Method* (World Scientific Publishing, Singapore, 2003)
54. P.J. Frey, P.L. George, *Mesh Generation: Application to Finite Elements* (Wiley, Hoboken, 2008)
55. Z. Wang, Prog. Aerosp. Sci. **43**(1–3), 1 (2007)
56. R.R. Nourgaliev, T.N. Dinh, T.G. Theofanous, D. Joseph, Int. J. Multiphas. Flow **29**(1), 117 (2003)
57. S. Geller, M. Krafczyk, J. Tölke, S. Turek, J. Hron, Comput. Fluids **35**(8-9), 888 (2006)
58. M. Yoshino, T. Inamuro, Int. J. Num. Meth. Fluids **43**(2), 183 (2003)
59. B. Dünweg, A.J.C. Ladd, in *Advances in Polymer Science* (Springer, Berlin, Heidelberg, 2008), pp. 1–78
60. A.J.C. Ladd, J. Fluid Mech. **271**, 285 (1994)
61. R. Adhikari, K. Stratford, M.E. Cates, A.J. Wagner, Europhys. Lett. **71**(3), 473 (2005)
62. B. Dünweg, U.D. Schiller, A.J.C. Ladd, Phys. Rev. E **76**(3), 036704 (2007)
63. M. Gross, M.E. Cates, F. Varnik, R. Adhikari, J. Stat. Mech. **2011**(03), P03030 (2011)
64. D. Belardinelli, M. Sbragaglia, L. Biferale, M. Gross, F. Varnik, Phys. Rev. E **91**(2), 023313 (2015)
65. E.M. Viggen, The lattice Boltzmann method: Fundamentals and acoustics. Ph.D. thesis, Norwegian University of Science and Technology (NTNU), Trondheim (2014)