Chapter 5

Numerical modelling of fluid–grain interactions

5.1 Fluid simulation using Lattice Boltzmann method

Grain–fluid systems can be found in many scientific and engineering applications, such as suspensions, fluidised beds, sediment transport, and geo-mechanical problems. In general, the fundamental physical phenomenon in these systems are not well understood mainly due to the intricate complexity of grain–fluid interactions and the lack of powerful analysis tools (Han et al., 2007a). In addition to the interaction among soil grains, the motion of soil grains is mainly driven by gravity and the hydrodynamic force exerted by the fluid. While, the fluid flow pattern can be significantly affected by the presence of soil grains, and this often results in a turbulent flow. Hence, the development of an effective numerical framework for modelling both the fluid flow patterns and the grain–fluid interactions is very challenging.

Development of a numerical framework depends crucially on the size of the soil grains relative to the domain/mesh size (Feng et al., 2007). Traditionally, the Navier-Stokes equation is solved by a grid-based Computational Fluid Dynamics (CFD) method, such as the Finite Volume Method (Capecelatro and Desjardins, 2013) or a mesh-free technique such as the Smooth Particle Hydrodynamics (Sun et al., 2013). The grid size in FVM or the smooth length in SPH for discretisation of the Navier-Stokes equation is at least an order of magnitude larger than the grain diameter (Xiong et al., 2014).

In situations where the average domain concentration phase is far from dilute, the computational effort is mostly devoted to the grain dynamics. The hydrodynamic force on the soil grains are applied based on an empirical relation using the domain-averaged local porosity of the soil grains in the grid. As a result, developing a fast fluid hydrodynamics solver is unim-

portant for dense flows. However, most geo-mechanics problems involve complex interactions between the solid and the fluid phase. This requires accurate modelling of the fluid flow pattern. Additionally, geophysical problems, such as submarine landslides and debris flow have a relatively large simulation domain, which requires parallel computation. Implementing traditional grid-based CFD methods face great challenges on multi-processor systems (Xiong et al., 2014). Although mesh-free approaches are free from the problem of parallel scalability, its modelling accuracy and speed are relatively low when compared to grid-based CFD methods. Therefore, an accurate, fast and a highly scalable scheme is required to model fluid - grain systems in geo-mechanics.

The Navier-Stokes equation describes the motion of a non-turbulent Newtonian fluid. The equation is obtained by applying Newton's second law to the fluid motion, together with an assumption that the fluid stress is the sum of the viscous term, proportional to the gradient of the velocity, and the pressure term. Conventional CFD methods compute pertinent flow fields, such as velocity u and pressure p, by numerically solving the Navier-Stokes equation in space x and time t. Alternatively, the transport equation or the Boltzmann equation, which deals with a single particle distribution function $f(x,\xi,t)$ in phase space (x,ξ) and time t, can be used to solve various problems in fluid dynamics.

The Lattice Boltzmann Method (LBM) (Chen and Doolen, 1998; Han et al., 2007b; He and Luo, 1997a,b; Mei et al., 2000; Zhou et al., 2012) is an alternative approach to the classical Navier-Stokes solvers for fluid flows. LBM works on an equidistant grid of cells, called lattice cells, which interact only with their direct neighbours. In LBM, the discretisation of continuum equations is based on microscopic models and mesoscopic continuum theories. LBM is a special discretising scheme of the Boltzmann equation where the particle distribution functions (mass fractions) collide and propagate on a regular grid. The important aspect, however, is the *discretisation of the velocity*, which means that the particle velocities are restricted to a predefined set of orientations.

The theoretical premises of the LB equation are that (1) hydrodynamics is insensitive to the details of microscopic physics, and (2) hydrodynamics can be preserved so long as the conservation laws and associated symmetries are respected in the microscopic and mesoscopic level. Therefore, the computational advantages of LBM are achieved by drastically reducing the particle velocity space ξ to only a very few discrete points without seriously degrading the hydrodynamics (Mei et al., 2000). This is possible because LBM rigorously preserves the hydrodynamic moments of the distribution function, such as mass density and momentum fluxes, and the necessary symmetries (He and Luo, 1997a,b). LBM has evolved as a comprehensive fluid solver and its theoretical aspects link well with the conventional central finite difference scheme (Cook et al., 2004).

5.1.1 Formulation

LBM is a 'micro-particle' based numerical time-stepping procedure for the solution of incompressible fluid flows. Consider a 2D incompressible fluid flow with density ρ and kinematic viscosity v, in a rectangular domain $\boldsymbol{D}$. The fluid domain is divided into a rectangular grid or lattice, with the same spacing *'h'* in both the x- and the y-directions, as shown in figure 5.1.

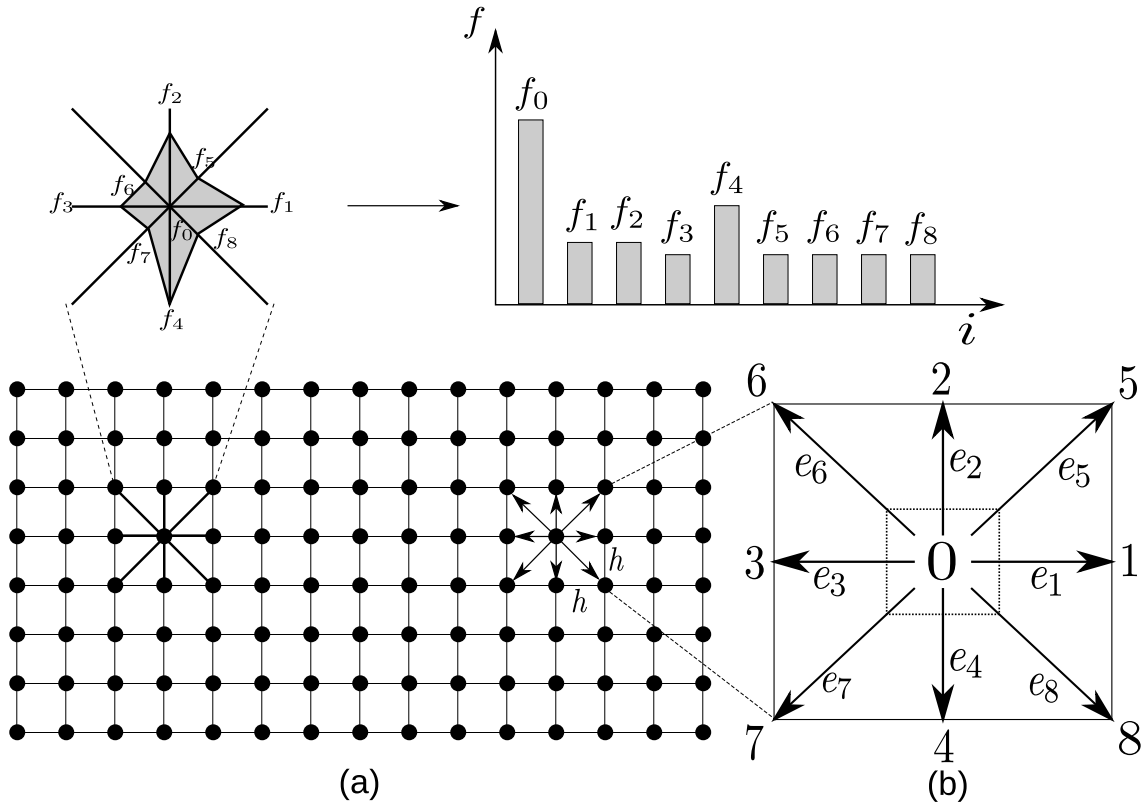


Figure 5.1 The Lattice Boltzmann discretisation and D2Q9 scheme: (a) a standard LB lattice and histogram views of the discrete single particle distribution function/direction-specific densities f_i; (b) D2Q9 model.

These lattices are usually classified in the literature using the $D\alpha Q\beta$-notation, where α denotes the space dimensionality and β is the number of discrete velocities (but also including the possibility of having particle at rest) within the momentum discretisation. The most common lattices are the *D2Q9* and the *D3Q19*-models, see He et al. (1997). The present study focuses on two-dimensional problems, hence the *D2Q9* momentum discretisation is adopted.

LBM discretises the Boltzmann equation in space to a finite number of possible particle spatial positions, microscopic momenta, and time. Particle positions are confined to the nodes of the lattice. The fluid particles at each node are allowed to move to their eight intermediate neighbours with eight different velocities $e_i (i = 1, \ldots, 8)$. A particle can remain at it's own

node, which is equivalent to moving with zero velocity e_o. The particle mass is uniform, hence these microscopic velocities and momentum are always effectively equivalent (Han et al., 2007b). Referring to the numbering system shown in figure 5.1, these nine discrete velocity vectors are defined as

$$\begin{cases} e_0 = (0,0); \\ e_1 = C(1,0); e_2 = C(0,1); e_3 = C(-1,0); e_4 = C(0,-1); \\ e_5 = C(1,1); e_6 = C(-1,1); e_7 = C(-1,-1); e_8 = C(1,-1)\,, \end{cases} \tag{5.1}$$

where C is the lattice speed that is defined as $C = h/\Delta t$, where Δt is the discrete time step. The primary variables in LB formulation are called the *fluid density distribution functions*, f_i, each relating the probable amount of fluid particles moving with the velocity e_i along the i^{th} direction at each node. The macroscopic variables are defined as functions of the particle distribution function (see figure 5.1)

$$\begin{cases} \rho = \sum_{i=0}^{\beta-1} f_i \qquad \text{(macroscopic fluid density)} \\ \text{and} \\ \overrightarrow{u} = \frac{1}{\rho} \sum_{i=0}^{\beta-1} f_i \overrightarrow{e_i} \quad \text{(macroscopic velocity)}\,, \end{cases} \tag{5.2}$$

where $i \in [0, \beta-1]$ is an index spanning the discretised momentum space. There are nine fluid density distribution functions, $f_i (i = 0, \ldots, 8)$, associated with each node in the *D2Q9* model. The evolution of the density distribution function at each time step for every lattice point is governed by

$$f_i(\mathbf{x} + \mathbf{e}_i \Delta t, t + \Delta t) = f_i(\mathbf{x}, t) - \frac{1}{\tau}[f_i(\mathbf{x}, t) - f_i^{eq}(\mathbf{x}, t)] \quad (i = 0, \ldots, 8)\,, \tag{5.3}$$

where for any grid node $\mathbf{x}, \mathbf{x} + \mathbf{e_i}\Delta t$ is its nearest node along the direction i. τ is a non-dimensional relaxation time parameter, which is related to the fluid viscosity; and f_i^{eq} is termed as the equilibrium distribution function that is defined as

$$\begin{cases} f_0^{eq} = w_0 \rho (1 - \frac{3}{2C^2} \mathbf{v}.\mathbf{v}) \\ \text{and} \\ f_i^{eq} = w_i \rho (1 + \frac{3}{C^2} \mathbf{e}_i.\mathbf{v} \frac{9}{2C^2} (\mathbf{e}_i.\mathbf{v})^2 - \frac{3}{2C^2} \mathbf{v}.\mathbf{v}) \quad (i = 0, \ldots, 8)\,, \end{cases} \tag{5.4}$$

in which, w_i represents the fixed weighting values

$$w_0 = \frac{4}{9}, \quad w_{1,2,3,4} = \frac{1}{9}, \quad \text{and} \quad w_{5,6,7,8} = \frac{1}{36}. \tag{5.5}$$

The right-hand side of eq. 5.3 is often denoted as $f_i(\mathbf{x}, t_+)$ and termed the post collision distribution. LBM ensures conservation of total mass and total momentum of the fluid particles at each lattice node (see eq. 5.3). It essentially consists of two phases: *collision* and *streaming*. The collision phase computed in the right-hand side of eq. 5.3 involves only those variables that are associated with each node $\mathbf{x}$, and therefore is a local operation. The streaming phase then explicitly propagates the updated distribution functions at each node to its neighbours $\mathbf{x} + e_i \Delta t$, where no computations are required and only data exchange between neighbouring nodes are necessary. These features, together with the explicit time-stepping nature and the use of a regular grid, make LB computationally efficient, simple to implement and natural to parallelism (Han et al., 2007b).

The streaming step involves the translation of the distribution functions to their neighbouring sites according to the respective discrete velocity directions, as illustrated in figure 5.2 in the *D2Q9* model. The collision step, (see figure 5.3) consists of re-distribution the local discretised Maxwellian equilibrium functions in such a way that local mass and momentum are invariants. In incompressible flows, the energy conservation is equivalent to the momentum conservation (He et al., 1997).

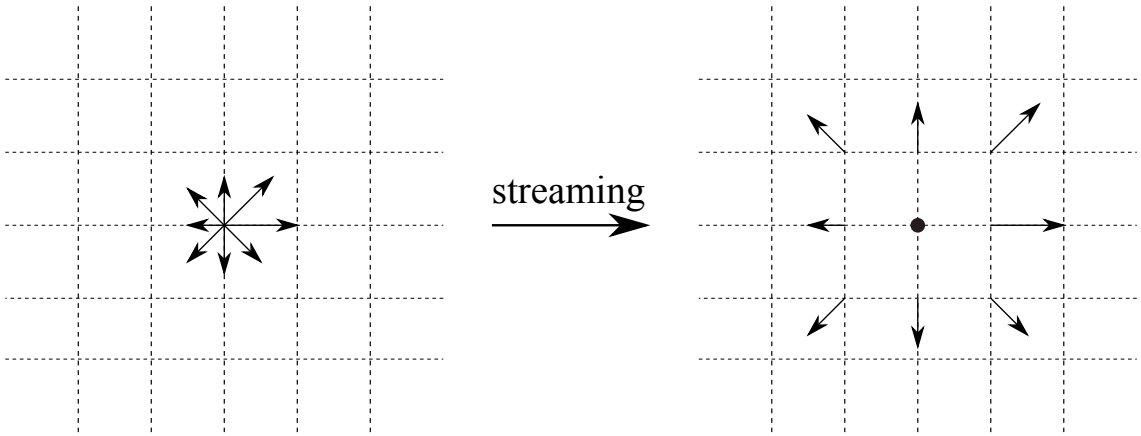


Figure 5.2 Illustration of the streaming process on a *D2Q9* lattice. The magnitude of the distribution functions remains unchanged, but they move to a neighbouring node according to their direction.

The standard macroscopic fluid variables, such as density ρ and velocity v, can be recov-

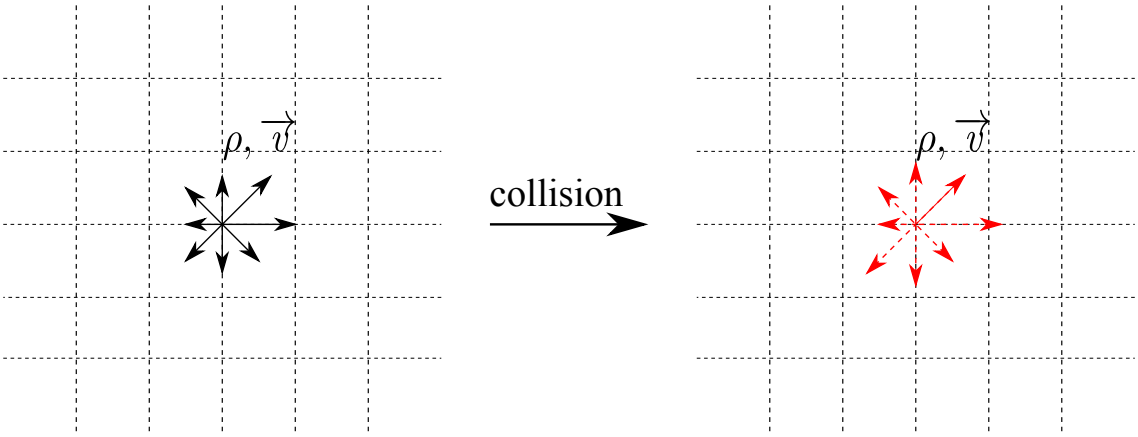


Figure 5.3 Illustration of the collision process on a *D2Q9* lattice. The local density ρ and velocity $\mathbf{v}$ are conserved, but the distribution functions change according to the relaxation to local Maxwellian rule.

ered from the distribution functions as

$$\rho = \sum_{i=0}^{8} f_i, \quad \text{and} \quad \rho \mathbf{v} = \sum_{i=0}^{8} f_i e_i. \tag{5.6}$$

The fluid pressure field 'p' is determined by the equation of state

$$p = {C_s}^2 \rho, \tag{5.7}$$

where C_s is termed the fluid speed of sound and is related to the lattice speed C by

$$C_s = C/\sqrt{3}. \tag{5.8}$$

The kinematic viscosity of the fluid $\boldsymbol{\nu}$ is implicitly determined by the model parameters h, Δt and τ as

$$\nu = \frac{1}{3}(\tau - \frac{1}{2})\frac{h^2}{\Delta t} = \frac{1}{3}(\tau - \frac{1}{2})Ch, \tag{5.9}$$

which indicates that these three parameters are related to each other and have to be appropriately selected to represent the correct fluid viscosity. An additional constraint to the parameter selection is the lattice speed C, which must be sufficiently large in comparison to the maximum fluid velocity v_{max} in the simulation, to ensure accuracy of the solution. This is measured by the 'computational' Mach number, M_a, defined by

$$M_a = \frac{v_{max}}{C}. \tag{5.10}$$

Theoretically, the Mach number is required to be $<< 1$. In practice, M_a should be, at least

smaller than 0.1 (He et al., 1997). From a computational point of view, it is more convenient to choose h and τ as two independent parameters and Δt as the derived parameter:

$$\Delta t = (\tau - \frac{1}{2})\frac{h^2}{3v}. \tag{5.11}$$

It can be observed that τ has to be greater than $1/2$ (He et al., 1997). Since there is no a priori estimation available to determine appropriate values of h and τ for a fluid flow problem with given fluid viscosity v, a *trial and error* approach is employed to obtain results satisfying the requirement of a smaller *Mach Number*. This is similar to the choice of an appropriate Finite Element mesh size, without automatic adaptive mesh techniques.

5.1.2 Lattice Boltzmann - Multi-Relaxation Time (LBM-MRT)

The Lattice Boltzmann Bhatnagar-Gross-Krook (LGBK) method is capable of simulating various hydrodynamics, such as multiphase flows and suspensions in fluid (Succi, 2001; Succi et al., 1989). LBM also offers intrinsic parallelism. However, LBM suffers from numerical instability when the dimensionless relaxation time τ is close to 0.5. The Multi-Relaxation Time – Lattice Boltzmann Method (LBM-MRT) overcomes the deficiencies of linearlised single relaxation LBM-BGK, such as fixed Prandtl number (Pr=ν/κ), where the thermal conductivity 'κ' is unity (Liu et al., 2003). LB-MRT model offers better numerical stability and has more degrees of freedom. In LB-MRT model the advection is mapped onto the momentum space by a linear transformation and the flux is still finished in the velocity space (Du et al., 2006).

The Lattice Boltzmann equation with multiple relaxation time approximation is written as

$$f_\alpha(\mathbf{x}+\mathbf{e}_i\Delta_t, t+\Delta_t) - f_\alpha(\mathbf{x},t) = -\mathbf{S}_{\alpha i}(f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t)\,, \tag{5.12}$$

where $\mathbf{S}$ is collision matrix. The nine eigen values of $\mathbf{S}$ are all between 0 and 2 so as to maintain linear stability and the separation of scales. This ensures that the relaxation times of non-conserved quantities are much faster than the hydrodynamic time scales. The LGBK model is a special case in which the nine relaxation times are all equal and the collision matrix $\mathbf{S} = \frac{1}{\tau}\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. The evolutionary progress involves two steps, advection and flux

$$f_\alpha^+(\mathbf{x},t) - f_\alpha(\mathbf{x},t) = -\mathbf{S}_{\alpha i}(f_i(\mathbf{x},t) - f_i^{eq}(\mathbf{x},t) \tag{5.13}$$

$$f_\alpha(\mathbf{x}+e_\alpha\Delta_t, t+\Delta_t) = f_\alpha^+(\mathbf{x},t)\,. \tag{5.14}$$

The advection (Equation 5.13) can be mapped to the momentum space by multiplying with

a transformation matrix $\mathbf{M}$. The evolutionary equation of the multi-relaxation time Lattice Boltzmann equation is written as

$$\mathbf{f}(\mathbf{x}+\mathbf{e}_i\Delta_t, t+\Delta_t) - \mathbf{f}(\mathbf{x},t) = -M^{-1}\hat{\mathbf{S}}(\hat{\mathbf{f}}(\mathbf{x},t) - \hat{\mathbf{f}}^{eq}(\mathbf{x},t))\,, \tag{5.15}$$

where $\mathbf{M}$ is the tranformation matrix mapping a vector $\mathbf{f}$ in the discrete velocity space $\mathbb{V} = \mathbb{R}^b$ to a vector $\hat{\mathbf{f}}$ in the moment space $\mathbb{V} = \mathbb{R}^b$.

$$\hat{\mathbf{f}} = \mathbf{M}\mathbf{f}, \tag{5.16}$$

$$\mathbf{f}(\mathbf{x},t) = [f_0(\mathbf{x},t), f_1(\mathbf{x},t), \ldots f_8(\mathbf{x},t)]^T\,. \tag{5.17}$$

The collision matrix $\hat{\mathbf{S}} = MSM^{-1}$ in moment space is a diagonal matrix

$$\hat{\mathbf{S}} = \mathrm{diag}\,[s_1, s_2, s_3, \ldots s_9]\,.$$

The transformation matrix $\mathbf{M}$ can be constructed via Gram-Schmidt orthgonalisation procedure. The general form of the transformation matrix $\mathbf{M}$ can be written as

$$\mathbf{M} = \left[|p\rangle, |e\rangle, |e^2\rangle, |u_x\rangle, |q_x\rangle, |u_y\rangle, |q_y\rangle, |p_{xx}\rangle, |p_{xy}\rangle\right]^T\,, \tag{5.18}$$

whose elements are,

$$|p\rangle = |e_\alpha|^0 \tag{5.19a}$$

$$|e\rangle_\alpha = Qe_\alpha^2 - b_2 \tag{5.19b}$$

$$|e^2\rangle_\alpha = a_1(Qe_\alpha^4 - b_6) + a_2(Qe_\alpha^4 - b_6) \tag{5.19c}$$

$$|u_x\rangle_\alpha = e_{\alpha,x} \tag{5.19d}$$

$$|q_x\rangle_\alpha = (b_1 e_\alpha^2 - b_3)e_{\alpha,x} \tag{5.19e}$$

$$|u_y\rangle_\alpha = e_{\alpha,y} \tag{5.19f}$$

$$|q_y\rangle_\alpha = (b_1 e_\alpha^2 - b_3)e_{\alpha,y} \tag{5.19g}$$

$$|p_{xx}\rangle_\alpha = de_{\alpha,x}^2 - e_\alpha^2 \tag{5.19h}$$

$$|p_{xx}\rangle_\alpha = e_{\alpha,x}e_{\alpha,y}\,, \tag{5.19i}$$

where $d = 2$ and $Q = 9$, $b_1 = \sum_{\alpha=1}^{Q} e_{\alpha,x}^2$, $b_2 = \sum_{\alpha=1}^{Q} e_\alpha^2$, $b_3 = \sum_{\alpha=1}^{Q} e_\alpha^2 e_{\alpha,x}^4$, $a_1 = ||e^2||^2$, and $a_2 = \sum_{\alpha=0}^{Q-1}(Qc_\alpha^2 - b_2) \times (Qc_\alpha^4 - b_6)$.

Explicitly, the transformation matrix can be written as

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{5.20}$$

The corresponding equilibrium distribution functions in moment space $\widehat{\mathbf{f}^{eq}}$ is given as

$$\widehat{\mathbf{f}^{eq}} = \left[\rho_0, e^{eq}, e^{2eq}, u_x, q_x^{eq}, q_y^{eq}, p_{xx}^{eq}, p_{xy}^{eq}\right]^T, \tag{5.21}$$

where

$$e^{eq} = \frac{1}{4}\alpha_2 p + \frac{1}{6}\gamma_2(u_x^2 + y_y^2) \tag{5.22a}$$

$$e^{2eq} = \frac{1}{4}\alpha_3 p + \frac{1}{6}\gamma_4(u_x^2 + y_y^2) \tag{5.22b}$$

$$q_x^{eq} = \frac{1}{2}c_1 u_x \tag{5.22c}$$

$$q_y^{eq} = \frac{1}{2}c_2 u_y \tag{5.22d}$$

$$p_{xx}^{eq} = \frac{3}{2}\gamma_1(u_x^2 - u_y^2) \tag{5.22e}$$

$$p_{xy}^{eq} = \frac{3}{2}\gamma_3(u_x u_y). \tag{5.22f}$$

To get the correct hydrodynamic equation, the values of the co-efficients are chosen as $\alpha_2 = 24$, $\alpha_3 = -36$, $c_1 = c_2 = -2$, $\gamma_1 = \gamma_3 = 2/3$, $\gamma_2 = 18$ and $\gamma_4 = -18$. $s_8 = s_9 = \tau$ and $s_1 = s_4 = s_6 = 1.0$ and the others vary between 1.0 and 2.0 for linear stability. Through the Chapman-Enskog expansion (Du et al., 2006), the incompressible Navier-Stokes equation can be recovered and the viscosity is given as

$$\nu = c_s^2 \Delta t(\tau - 0.5). \tag{5.23}$$

5.1.3 Boundary conditions

Boundary conditions (BC) form an important part of any numerical solutions. In many cases, the boundary conditions can strongly influence the accuracy of the algorithm. The velocity and pressure are not primary variables in LBM, hence the standard pressure, velocity, and mixed boundary conditions cannot be imposed directly. Alternative conditions in terms of the distribution functions are adopted to describe the boundary conditions.

Periodic boundary condition

The simplest type of boundary condition is the periodic boundary. In this case, the domain is folded along the direction of the periodic boundary pair. For boundary nodes, the neighbouring nodes are on the opposite boundary, using the normal referencing of neighbours (see figure 5.1a). From the perspective of submarine landslide modelling, the periodic boundary conditions are useful for preliminary analysis, as they imply higher degree of symmetry of the fluid domain. Further information on the periodic boundary condition can be found in Aidun et al. (1998).

No-slip boundary condition

The most commonly adopted BC in the Lattice Boltzmann approach is the no-slip BC, especially the simple bounce-back rule, which is quite elegant and surprisingly accurate. The basic idea is that the incoming distribution functions at a wall node are reflected back to the original fluid nodes, but with the direction rotated by π radians. The bounce-back boundary condition is one of benefits of LBM, as it is trivial to implement and it allows one to effortlessly introduce obstacles into the fluid domain. However, the boundary conditions have been proven to be only first-order accurate in time and space (Pan et al., 2006). A straightforward improvement is to consider the wall-fluid interface to be situated halfway between the wall and the fluid lattice nodes (Ziegler, 1993). It involves, defining the *solid* nodes as those lying within the stationary wall regions, and the *fluid* nodes otherwise. Then if i is the direction between a fluid node n_1 and a solid node n_2, the bounce-back rule requires that the incoming fluid particle from n_1 to n_2 be reflected back along the direction it came from, i.e.,

$$f_{-i}(\mathbf{x}, t+\Delta t) = f_i(\mathbf{x}, t_+)\,, \tag{5.24}$$

where $-i$ denotes the opposite direction of i. The bounce back rule is illustrated in figure 5.4. This simple rule ensures that no tangential velocity exists along the fluid-wall interface, thereby a non-slip condition is imposed, and can be extended to any shapes or objects

in a fluid flow (Han et al., 2007a; Zou and He, 1997). The slip boundary conditions have similar treatment to the non-slip condition, except that the distribution functions are reflected in the boundary instead of bounce-back (Succi, 2001).

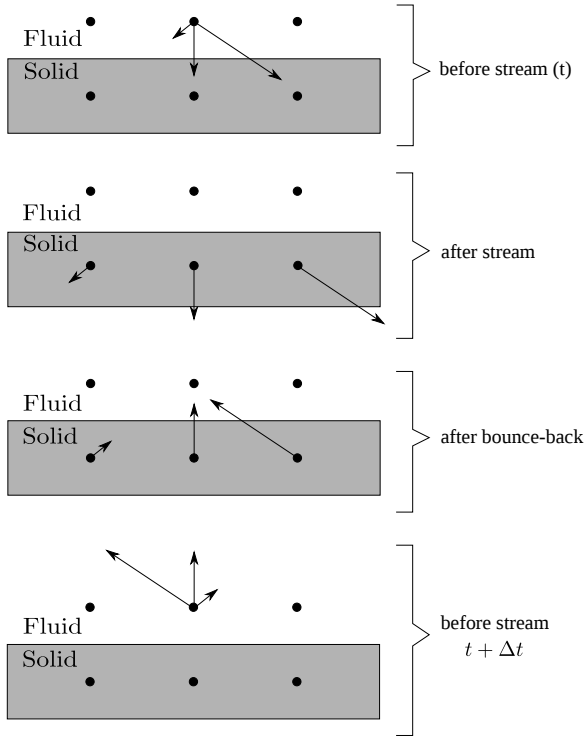


Figure 5.4 Half-way bounce back algorithm for the *D2Q9* model adopted after Sukop and Thorne (2006).

Pressure and velocity boundary condition

The pressure (Drichlet) boundary condition can be imposed in Lattice Boltzmann by specifying a fluid density at the pressure boundary (Zou and He, 1997). To impose a pressure boundary along the y-direction (inlet, left side boundary in figure 5.5), a density $\rho = \rho_{in}$ is specified from which the velocity is computed. The vertical component of the velocity on the boundary is set as zero, $u_y = 0$. After streaming, f_2, f_3, f_4, f_6, f_7 are known, u_x and f_1, f_5, f_8 are to be determined from eq. 5.2 as following

$$f_1 + f_5 + f_8 = \rho_{in} - (f_0 + f_2 + f_3 + f_4 + f_6 + f_7) \tag{5.25}$$

$$f_1 + f_5 + f_8 = \rho_{in} u_x + (f_3 + f_6 + f_7) \tag{5.26}$$

$$f_5 - f_8 = f_2 - f_4 + f_6 - f_7\,, \tag{5.27}$$

Consistency of equations (5.25) and (5.26) gives

$$u_x = 1 - \frac{[f_0 + f_2 + f_4 + 2*(f_3 + f_6 + f_7)]}{\rho_{in}}. \tag{5.28}$$

Using bounce-back rule for the non-equilibrium part of the particle distribution normal to the inlet to find $f_1 - f_1^{eq} = f_3 - f_3^{eq}$. The values of f_5 and f_8 can be obtained from f_1

$$\begin{aligned} f_1 &= f_3 + \frac{2}{3}\rho_{in}u_x \\ f_5 &= f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_{in}u_x \\ f_8 &= f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_{in}u_x\,. \end{aligned} \tag{5.29}$$

The corner node at inlet needs some special treatment. Considering the bottom node at inlet as example, after streaming, f_3, f_4, f_7 are known; ρ is defined, and $u_x = u_y = 0$. The particle distribution functions f_1, f_2, f_5, f_6, f_8 are to be determined. The bounce-back rule for the non-equilibrium part of the particle distribution normal to the inlet and the boundary is used to find

$$f_1 = f_3 + (f_1^{eq} - f_3^{eq}) = f_3 \tag{5.30}$$

$$f_2 = f_4 + (f_1^{eq} - f_3^{eq}) = f_4\,. \tag{5.31}$$

Using these we can compute

$$f_5 = f_7 \tag{5.32}$$

$$f_6 = f_8 = \frac{1}{2}[\rho_{in} - (f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8)]\,. \tag{5.33}$$

Similar procedure can be applied to top inlet node and outlet nodes. Von Neumann boundary conditions constrain the flux at the boundaries. A velocity vector $u = [u_0\ v_0]^T$ is specified from which density/pressure is computed based on the domain. The velocity boundary condition can be specified in a similar way (Zou and He, 1997). The pressure and velocity boundary conditions contribute additional equation(s) to determine the unknown distribution functions. In the velocity boundary, the equation is sufficient to determine the unknown distribution functions in *D2Q9* model, however the pressure boundary conditions require additional constitutive laws to determine the unknown distribution functions.

Table 5.1 LBM parameters used in simulating laminar flow through a circular pipe.

Parameter	**Value**
Density ρ	1000 kg m^{-3}
Relaxation parameter τ	0.51
Kinematic viscosity	1×10^{-6} m^2 s^{-1}
Grid resolution 'h'	1^{-2} m
Number of steps	50,000
Error in predicting horizontal velocity	0.009 %

5.2 Validation of Lattice Boltzmann method

To verify the incompressible LBM model implemented in the above section, numerical simulation of the transient development of the steady state Poiseuille flow in a straight channel is performed. At $t = 0$, the LBM water particles ($\rho = 1000 kg/m^3$) are simulated to flow through a channel of width 'H' (= 0.4 m) and simulation length, 'L' (2.5H) under constant body force. Periodic boundary conditions are applied at either end of the pipe and the pressure gradient is set to zero, which simulates the condition of a continuous flow of fluid in a closed circular pipe. The length 'L' has no effect on the simulation as no streamwise variation is detected in the solution. The parameters adopted in LBM simulation are presented in table 5.1. Sufficient time is allowed for the flow to travel beyond the required development length so that the flow is laminar (Durst et al., 2005). The development length required for a flow to be fully laminar is

$$X_D/H = [(0.619)^{1.6} + (0.0567 R_e)^{1.6}]^{1/1.6}, \tag{5.34}$$

where X_D is the development length and R_e is the Reynold's number. The velocity profile at the end of the simulation is presented in figure 5.5. A maximum horizontal velocity of 0.037863 m s^{-1} is observed along the center-line of the channel. The maximum horizontal velocity is compared with the closed-form based on the Haygen-Poiseuille's flow equation for no-slip boundary condition (Willis et al., 2008)

$$U_x = \frac{\Delta P}{2\mu L}[\frac{H^2}{4} - y^2], \tag{5.35}$$

where v_x is the horizontal velocity (m/s); ΔP is the pressure gradient, μ dynamic viscosity of the fluid. LBM predicts the maximum horizontal velocity within an error of 0.009 %.

In order to further validate the accuracy of the Lattice Boltzmann code, the transient development of the Poiseuille's flow is compared with the CFD simulation performed using

ANSYS Fluent. Finite Volume Method is a common CFD technique, which involves solving the governing partial differential equation (Navier-Stokes) over the discretised control volume. This guarantees the conservation of fluxes over a particular control volume. The finite volume equations yield governing equations of the form

$$\frac{\partial}{\partial t}\int\int\int Q d\mathbf{V} + \int\int F d\mathbf{A} = 0\,, \tag{5.36}$$

where Q is the vector of conserved variables, F is the vector of fluxes in the Navier-Stokes equation, V is the volume of control volume element, and A is the surface area of the control volume element.

A 2D rectangular plane of length 1 m and height 0.04 m is discretised into 400 cells of size 1^{-2} m (see figure 5.6). A constant velocity is applied at the inlet. Water ($\rho = 998.2\ kg/m^3$, $viscosity`\eta' = 1 \times 10^{-3}\ Ns/m^2$) is allowed to flow through the channel and it develops into a fully laminar flow. The Least-Squares approach was adopted to solve the gradient, and a maximum of 100 iteration steps were carried out until the solution converged.

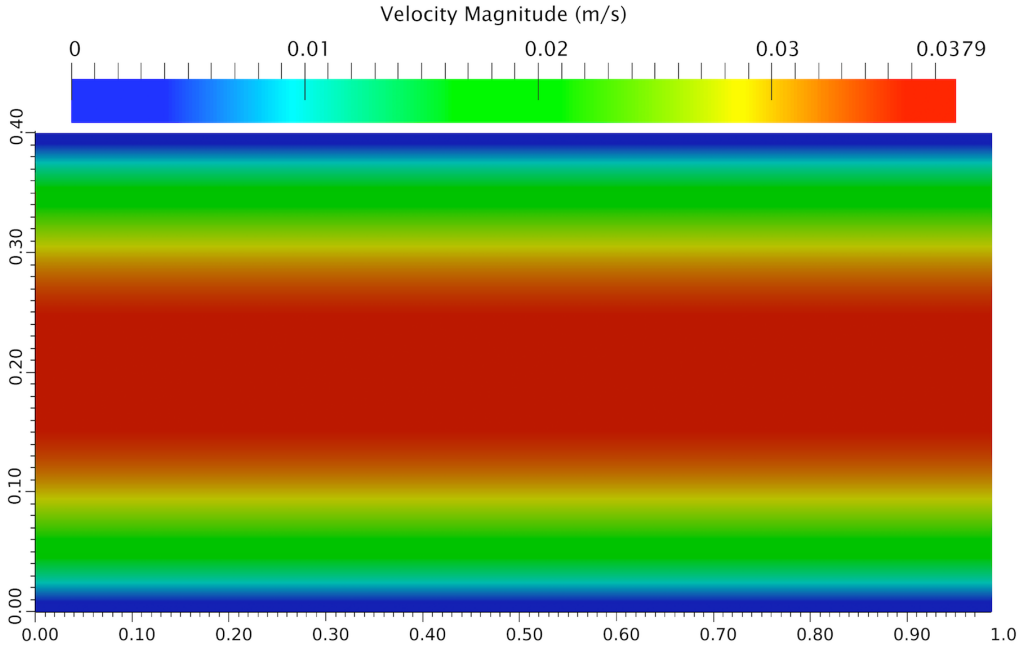


Figure 5.5 Velocity profile: LBM Simulation of a laminar flow through a channel.

The velocity profile obtained from the CFD simulation at cross-section 'L/4' is shown in figure 5.7. Figure 5.8 compares the development of computed velocity profiles at four times with the analytical solution. At normalised time $t = 1$, the flow approaches the steady state. It can be observed that LBM has excellent agreement with CFD and the analytical solution at various stages of flow evolution.

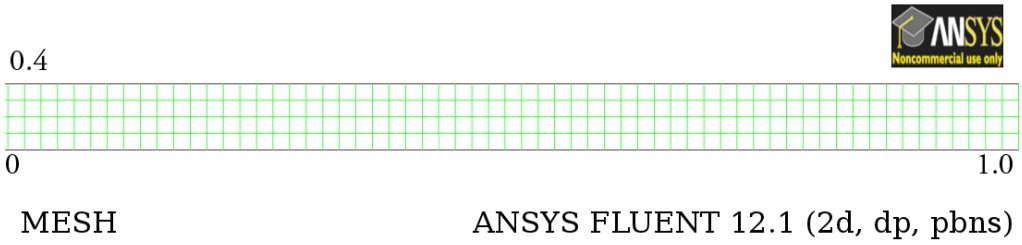


Figure 5.6 Finite Volume mesh used in the CFD analysis of laminar flow through a channel.

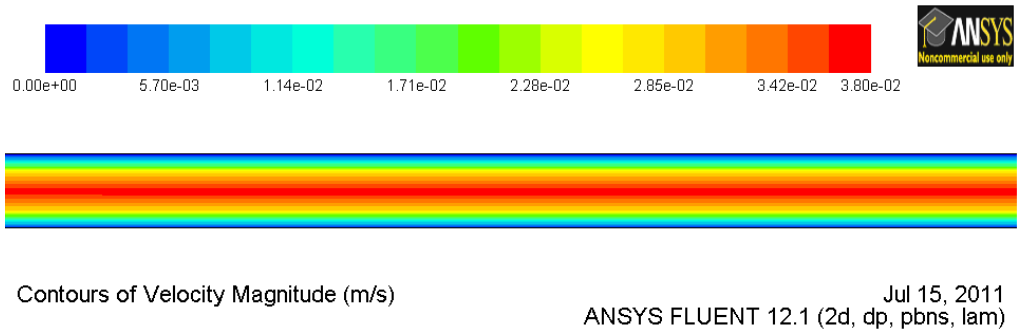


Figure 5.7 Velocity profile: CFD analysis of laminar flow through a channel.

In order to study the capability of LB technique to simulate fluid–solid interaction, LB simulation of fluid flow around a rectangular obstacle is compared with the CFD technique. A solid wall of height 'H/2' is placed at length 'L/4' in the channel. Bounce-back algorithm is employed to model the fluid-wall interaction in LBM. In the CFD model, the control volume is discretised into 10,000 cells. A constant velocity is applied in the inlet and the horizontal velocity profile is recorded. Both, CFD and LBM simulations were performed to study the influence of a solid wall on the fluid flow behaviour.

The horizontal velocity profile obtained after 50,000 LBM iterations is presented in figure 5.9. LBM is able to capture the velocity shedding around the edges of the wall. The velocity profile obtained from the CFD analysis is presented in figure 5.10.The horizontal velocity profile at 'L/4' at $t = 1$ is shown in figure 5.11. Similar maximum horizontal velocities are observed in both LBM and CFD analyses. The maximum horizontal velocity from the CFD analysis is slightly higher in comparison with the LBM simulation. The discrepancy in the horizontal velocity profile (figure 5.11) can be attributed to the relaxation parameter used in the LBM, which is obtained by a trial and error procedure. The velocity profile obtained from the LBM simulation compares qualitatively with the FE analysis performed by Zhong and Olson (1991). Thus, it can be concluded that the Lattice Boltzmann method is a suitable form of numerical representation of Navier-Stokes equation to model fluid flows.

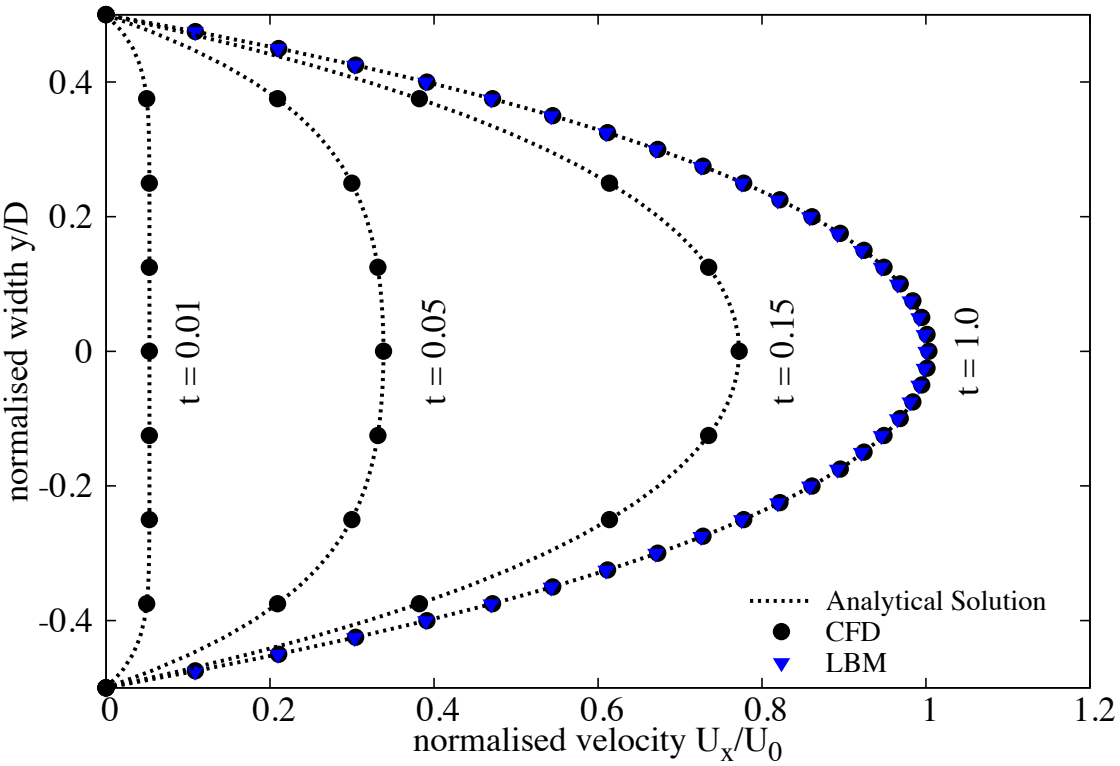


Figure 5.8 Development of the Poiseuille velocity profile in time: comparison between LBM simulation, CFD simulation and the analytical solution. Time is made dimensionless by H/U_0.

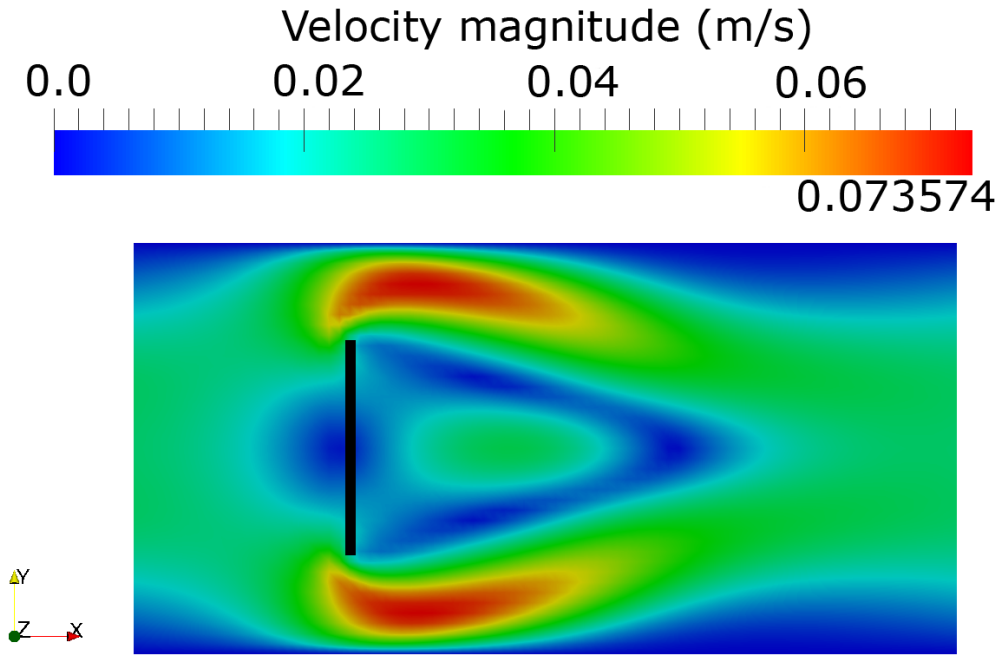


Figure 5.9 LBM simulation of velocity profile for a laminar flow through a pipe with an obstacle at L/4.

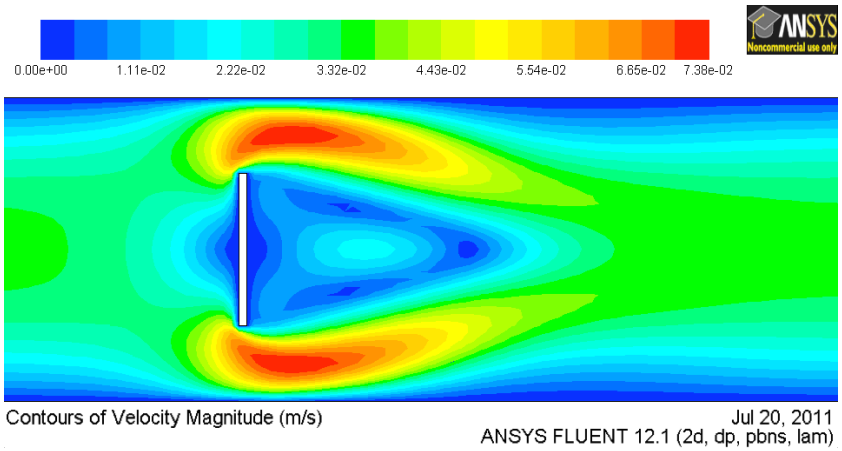


Figure 5.10 CFD simulation of velocity contour for a laminar flow through a pipe with an obstacle at L/4.

5.3 Turbulence in Lattice Boltzmann method

The above formulation of Lattice Boltzmann has been successfully applied to many fluid flow problems, however it is restricted to flows with low Reynold's number. Modelling fluids with low viscosity like water and air remains a challenge, necessitating very small values of h, and/or τ very close to 0.5 (He et al., 1997). The standard Lattice Boltzmann can deal with

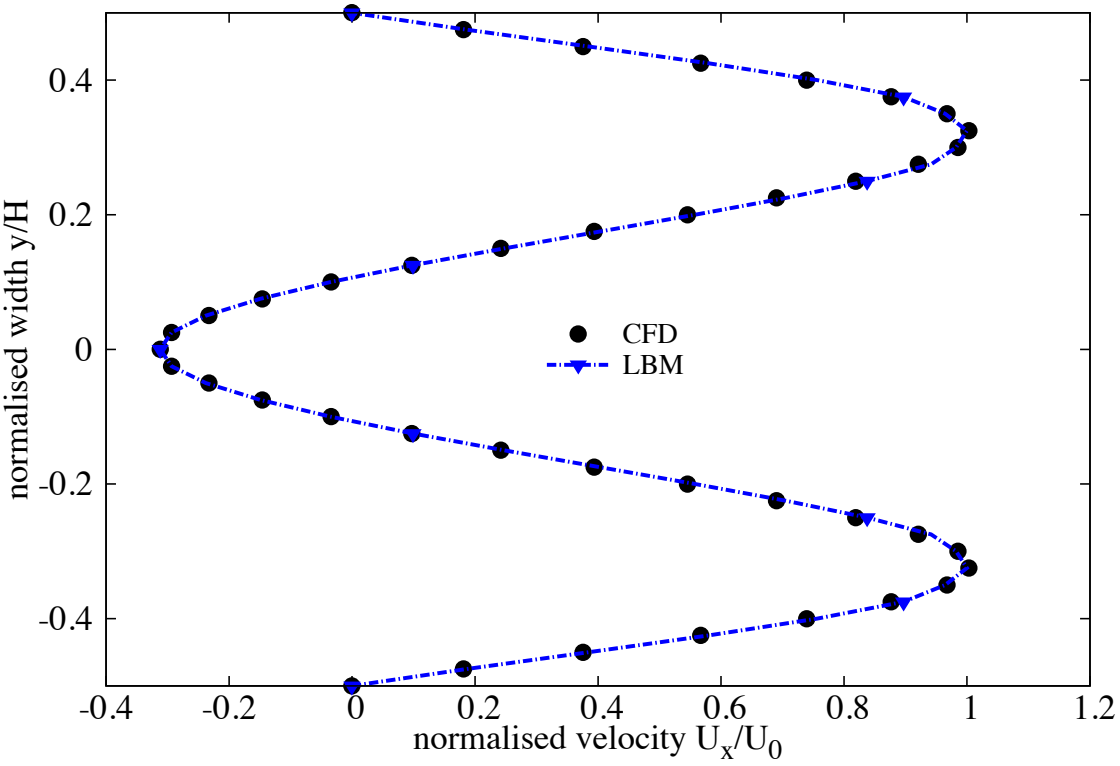


Figure 5.11 LBM and CFD simulation of a flow around an obstacle.

laminar flows, while practical problems with small kinematic viscosity are often associated with flows having large Reynold numbers, i.e. flows which are unsteady or turbulent in nature. The turbulent flows are characterised by the occurrence of eddies with multiple scales in space, time and energy.

The large eddy simulation (LES) is the most widely adopted approach to solve turbulent flow problems. It directly solves the large scale eddies, which carry the predominant portion of the energy, and the smaller eddies are modelled using a sub-grid approach. The separation of scales is achieved by filtering of the Navier-Stokes equations, from which the resolved scales are directly obtained. The unresolved scales are modelled by a one-parameter Smagorinski sub-grid methodology, which assumes that the Reynold's stress tensor is dependent only on the local strain rate (Smagorinsky, 1963). It involves parametrising the turbulent energy dissipation in the flows, where the larger eddies extract energy from the mean flow and ultimately transfer some of it to the smaller eddies which, in turn, pass the energy to even smaller eddies, and so on up to the smallest scales. At the smallest scale, the eddies convert the kinetic energy into the internal energy of the fluid. At this scale, the viscous friction dominates the flow (Frisch and Kolmogorov, 1995).

In Smargonisky model, the turbulent viscosity ν is related to the strain rate S_{ij} and a filtered length scale 'h' as follows

$$S_{ij} = \frac{1}{2}(\partial_i u_j + \partial_j u_i) \tag{5.37}$$

$$\nu_t = (S_c h)^2 \overline{S} \tag{5.38}$$

$$\overline{S} = \sqrt{\sum_{i,j} \tilde{S}_{i,j} \tilde{S}_{i,j}}\,, \tag{5.39}$$

where S_c is the Smagorinski constant found to be close to 0.03 (Yu et al., 2005). The effect of the unresolved scale motion is taken into account by introducing an effective collision relaxation time scale τ_t, so that the total relaxation time τ_* is written as

$$\tau_* = \tau + \tau_t\,, \tag{5.40}$$

where τ and τ_t are respectively the standard relaxation times corresponding to the true fluid viscosity ν and the turbulence viscosity ν_t, defined by a sub-grid turbulence model. The new

viscosity ν_* corresponding to τ_* is defined as:

$$\begin{aligned}\nu_* &= \nu + \nu_t \\ &= \frac{1}{3}(\tau_* - \frac{1}{2})C^2\Delta t = \frac{1}{3}(\tau + \tau_t - \frac{1}{2})C^2\Delta t \qquad (5.41)\\ \nu_t &= \frac{1}{3}\tau_t C^2\Delta t\,. \qquad (5.42)\end{aligned}$$

The Smagorinski model is easy to implement and the Lattice Boltzmann formulation remains unchanged, except for the use of a new turbulence-related viscosity τ_*. The component s_1 of the collision matrix becomes $s_1 = \frac{1}{\tau+\tau_t}$.

The effectiveness of LBM-LES model in simulating unsteady flows is verified by modelling the Kármán vortex street. In fluid dynamics, a kármán vortex street is a repeating pattern of vortices caused by unsteady separation of fluid flow around circular obstacles. A vortex street will only form typically above a limiting value of Reynolds number of 90. The Reynolds number is computed based on the cylinder diameter 'D' and the mean flow velocity U of the parabolic inflow profile

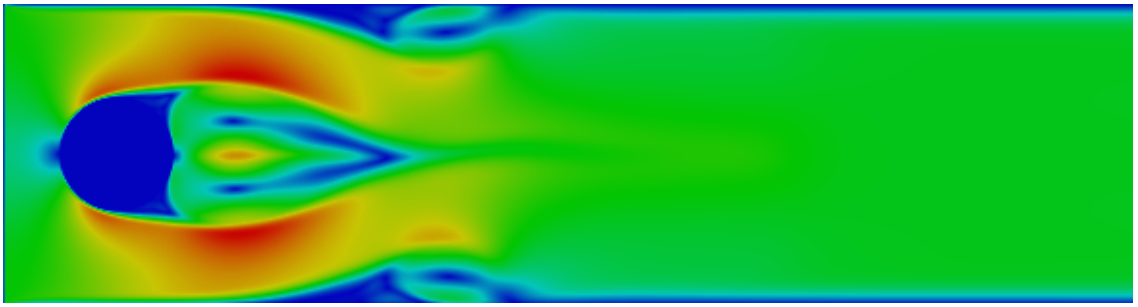
$$Re = \frac{UD}{\nu}\,. \qquad (5.43)$$

LBM particles are simulated to flow through a 2D rectangular channel with an aspect ratio 'L/H' of 2.5. A cylinder of diameter 'd' $= 0.27H$ is placed at H/2. The pressure gradient at the inlet and the outlet is varied to create flows with different mean velocities. Numerical simulations of vortex shedding behind a circular obstacle is carried out for three different fluid flow regimes (Reynolds number of 55, 75, and 112). The fully developed fluid flows for different Reynolds numbers are shown in figure 5.12. The von Kármán vortex street can only be observed at high a Reynolds number of 112 (Re > 90), which shows the ability of the LBM turbulence model in capturing instabilities in fluid flow.

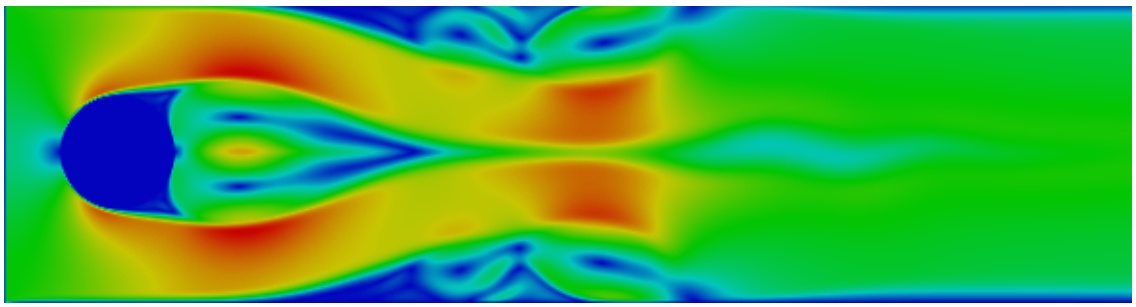
One important quantity taken into account in the present analysis is the Strouhal number St, a dimensionless number describing oscillating unsteady flow dynamics. Strouhal number is computed from the cylinder diameter D, the measured frequency of the vortex shedding f, and the maximum velocity U_{max} at the inflow plane

$$St = \frac{fD}{U_{max}}\,. \qquad (5.44)$$

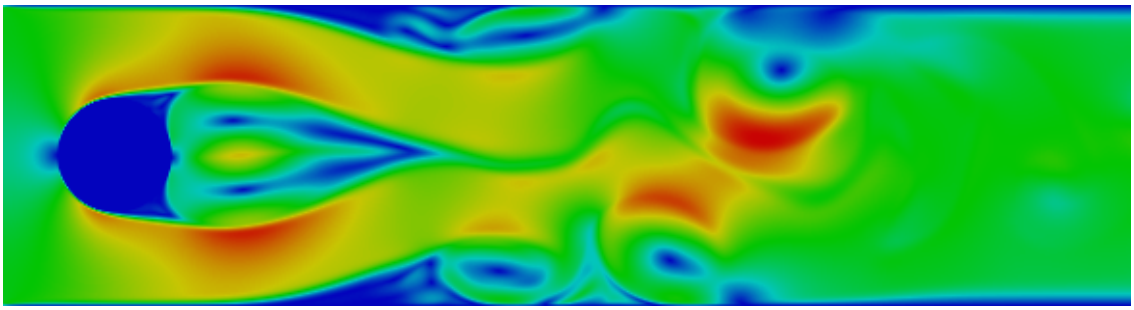
The characteristic frequency f is determined by a spectral analysis (Fast Fourier Transform - FFT) of time series of the fluid pressure. Table 5.2 shows that the Strouhal numbers computed from LBM simulations have a very good agreement with the FVM results obtained by Breuer et al. (2000). This shows the ability of LBM-LES in capturing unsteady flow dynamics.

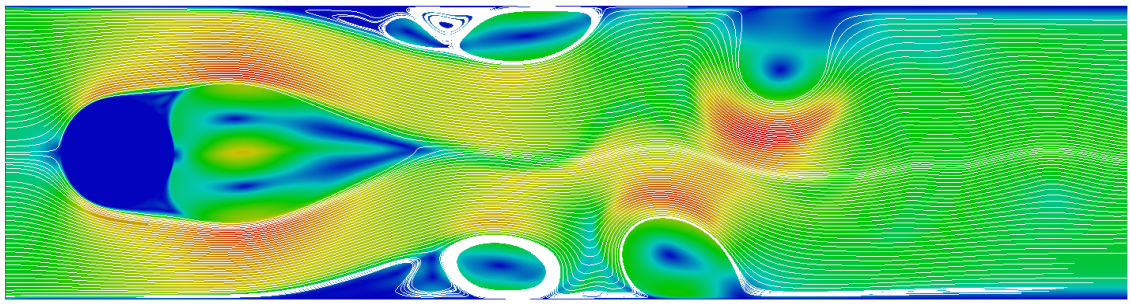


(a) Re = 55



(b) Re = 75





(c) Re = 112

Figure 5.12 Kármán vortex street

Table 5.2 Computed Strouhal number for fluid flows with different Reynolds number

Reynolds number	Strouhal number	
	LBM	FVM
55	0.117	0.117
75	0.128	0.129
112	0.141	0.141

* FVM results are from Breuer et al. (2000)

5.4 Coupled Lattice Boltzmann and DEM for fluid-grain interactions

Modelling fluid–grain interactions in submarine landslides requires the ability to simulate the dynamic fluid – solid boundary interactions. In principle, the conventional FE and FVM based approaches for solving the Navier-Stokes equations with moving boundaries and/or structural interaction (Bathe and Zhang, 2004) can be applied to particle fluid interaction problems. The common feature of these approaches is to model the interaction between the fluid and the solid to a high degree of accuracy. However, the main computational challenge is the need to continuously generate new geometrically adapted meshes to circumvent severe mesh distortion, which is computationally very intensive (Han et al., 2007b).

The Lattice Boltzmann approach has the advantage of accommodating large particle sizes and the interaction between the fluid and the moving particles can be modelled through relatively simple fluid - grain interface treatments. Further, employing DEM to account for the grain/grain interaction naturally leads to a combined LB – DEM solution procedure. The Eulerian nature of the Lattice Boltzmann formulation, together with the common explicit time step scheme of both LBM and DEM makes this coupling strategy an efficient numerical procedure for the simulation of fluid – grain systems.

LBM – DEM technique is a powerful predictive tool for gaining insights into many the fundamental physical phenomena in the fluid-solid interaction domains. Such a coupled methodology was first proposed by (Cook et al., 2004) for simulating fluid-grain systems dominated by fluid-grain and grain-grain interactions. To capture the actual physical behaviour of the fluid-grain system, it is essential to model the boundary condition between the fluid and the grain as a non-slip boundary condition, i.e. the fluid near the grain should have similar velocity as the grain boundary. The soil grains in the fluid domain are represented by lattice nodes. The discrete nature of lattice will result in stepwise representation of the surfaces, which are otherwise circular, this is neither accurate nor smooth, unless sufficiently small lattice spacing

is adopted.

Modified bounce back rule

To accommodate the movement of solid particles in the commonly adopted bounce-back rule (see section 5.1.3), Ladd (1994) modified the 'no-slip' rule for a given boundary link i to be

$$f_i(\mathbf{x}, t+\Delta t) = f_i(\mathbf{x}, t_+) - \alpha_i e_\mathbf{i}.v_b \qquad (\alpha_i = 6w_i\rho/C_s^2)\,, \tag{5.45}$$

where $f_i(\mathbf{x}, t_+)$ is the post collision distribution at the fluid or solid boundary node $\mathbf{x}$, and v_b is the velocity at the nominal boundary point at the middle of the boundary link i

$$\mathbf{v}_b = \mathbf{v}_c + \omega \times (\mathbf{x} + e_\mathbf{i}\Delta t/2 - \mathbf{x}_c)\,, \tag{5.46}$$

in which v_c and ω are the translational and angular velocities at the mass centre of the solid particle, respectively. $\mathbf{x}_c$ and $\mathbf{x} + e_\mathbf{i}\Delta t/2$ are the coordinates of the centre and the nominal boundary point, respectively. The impact force on the soil grain from the link is defined as

$$\mathbf{F_i} = 2[f_i(\mathbf{x}, t_+) - \alpha_i e_\mathbf{i}.\mathbf{v}_b]/\Delta t\,. \tag{5.47}$$

The corresponding torque $\mathbf{T}_i$, produced by the force with respect to the centre of the particle is computed as

$$\mathbf{T}_i = \mathbf{r}_c \times F_i(\mathbf{r}_c = \mathbf{x} + \mathbf{e}_i\Delta t/2 - \mathbf{x}_c)\,. \tag{5.48}$$

Then the total hydrodynamic force and torque exerted on the particle can be calculated by summing up the forces and torques from all the related boundary links

$$\begin{aligned} \mathbf{F} &= \sum_i \mathbf{F}_i \\ \mathbf{T} &= \sum_i \mathbf{T}_i\,. \end{aligned} \tag{5.49}$$

Ladd and Verberg (2001) described a methodology that minimises the oscillations resulting from soil grains crossing lattice at a very high speed. The fluid/grain force interaction method with momentum exchange method is coupled with the treatment of moving curved boundaries scheme (Yu et al., 2003). The simulation of the moving curved grain surfaces results in the intersection of links between two nodes at arbitrary distances (Iglberger et al., 2008). These distance values are called as delta values

$$\delta = \frac{\text{Distance between fluid nod and particle surface}}{\text{Distance between fluid node and particle node}} \in [0, 1]\,. \tag{5.50}$$

For each pair of neighbouring fluid and grain nodes, a delta value has to be calculated. Delta values of zero are not possible as the nodes on the surface are considered as particle nodes. The algorithm for computation of the δ value is presented in Iglberger et al. (2008). Figure 5.13 shows the three possible situations for delta values between 0 and 1. The fluid particles in LBM are always considered moving at the rate of one lattice per time step ($\delta\mathbf{x}/\delta\mathbf{t}$), for delta values smaller than 0.5. For δ values larger than 0.5, the fluid particles would come to rest at an intermediate node $\mathbf{x}_i$. In order to calculate the reflected distribution function in node $\mathbf{x}_f$, an interpolation scheme has to be applied. The linear interpolation scheme of Yu et al. (2003) is used in the present study, which uses a single equation, irrespective of the value of δ being smaller or larger than 0.5, to the reflected distribution function that is computed as

$$f_{\overline{\alpha}}(\mathbf{x}_f,t+\delta t) = \frac{1}{1+\delta}\cdot[(1-\delta)\cdot f_{\alpha}(\mathbf{x}_f,t+\delta t)+\delta\cdot f_{\alpha}(\mathbf{x}_b,t+\delta t) + \delta\cdot f_{\overline{\alpha}}(\mathbf{x}_{f2},t+\delta t)-2w_a\rho_w\frac{3}{c^2}e_a\cdot\mathbf{u}_w], \tag{5.51}$$

where w_α is the weighting factor, ρ_w is the fluid density in node $\mathbf{x}_f$, and $\mathbf{u}_w$ is the velocity at the bounce-back wall. In order to couple the fluid-grain interaction, the LBM approach is extended by adopting a force integration scheme, to calculate the fluid force acting on the grain surface, and the momentum exchanged method described earlier. The physical force acting on grain agglomerates is calculated as the sum over all fluid/particle node pairs, resulting in

$$F=\sum_{\mathbf{x}_b}\sum_{\alpha=1}^{19}\mathbf{e}_\alpha[f_\alpha(\mathbf{x}_b,t)+f_{\overline{\alpha}}(\mathbf{x}_f,t)]\delta\mathbf{x}/\delta t\,. \tag{5.52}$$

After the force calculations, the coupled rigid body physics can be simulated in order to move the grains/grain-agglomerates according to the applied forces. The total hydrodynamic forces and torque exerted on a grain can be computed as (Cook et al., 2004; Noble and Torczynski, 1998)

$$\mathbf{F}_f = Ch[\sum_n(\beta_n\sum_i f_i^m e_\mathbf{i})] \tag{5.53}$$

$$\mathbf{T}_f = Ch[\sum_n(\mathbf{x}_n-\mathbf{x}_c)\times(\beta_n\sum_i f_i^m e_\mathbf{i})]\,. \tag{5.54}$$

The summation is over all lattice nodes covered by the particle, and $\mathbf{x}_n$ represents the coordinate of the lattice node n.

When grains are not in direct contact among themselves, but are driven by the fluid flow

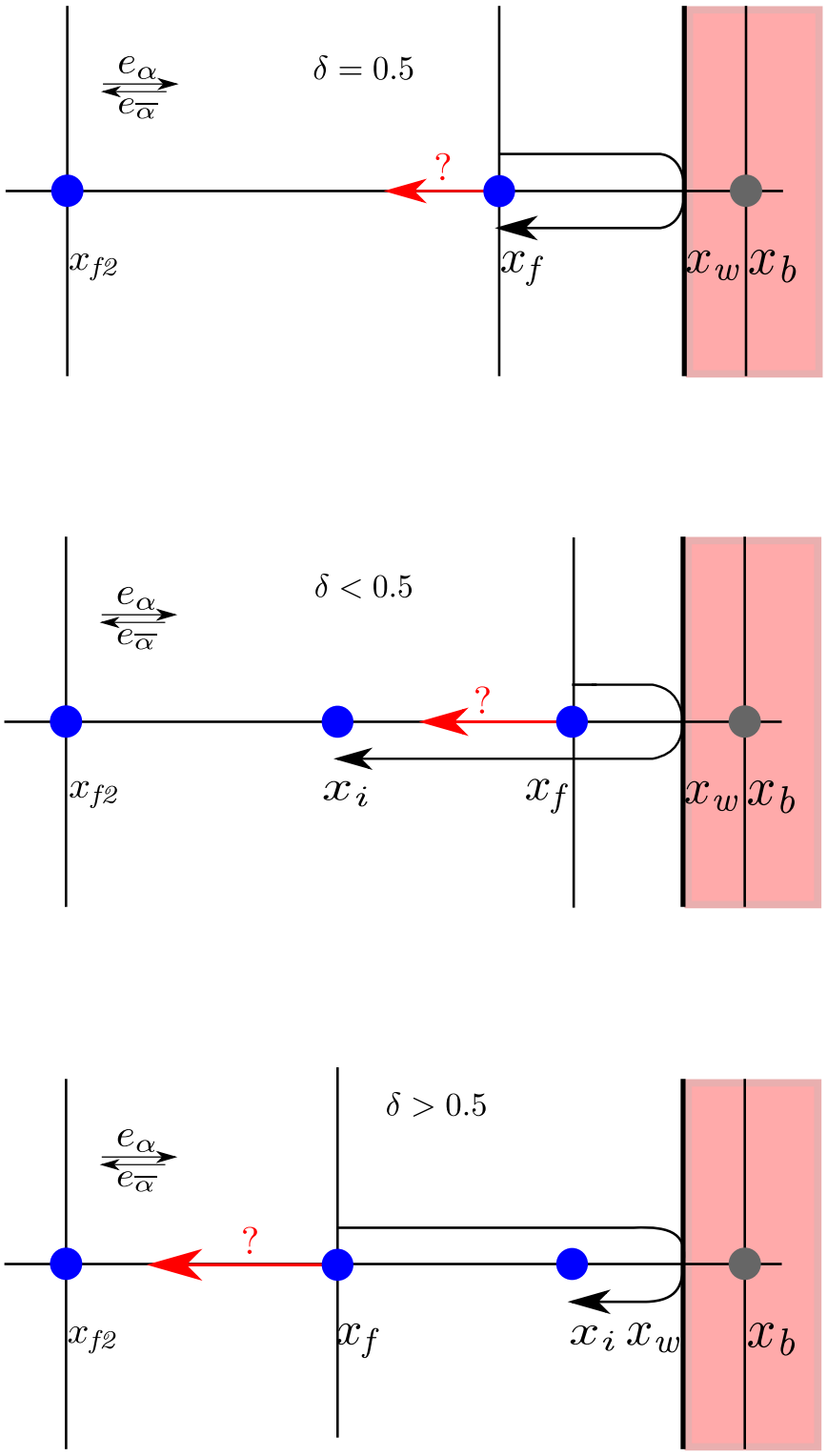


Figure 5.13 Bounce back boundaries for different values of δ

and body force, i.e. gravity, their motion can be determined by Newton's equation of motion

$$m\mathbf{a} = \mathbf{F}_f + m\mathbf{g} \tag{5.55}$$

$$J\ddot{\theta} = \mathbf{T}_f\,, \tag{5.56}$$

where m and J are respectively the mass and the moment of inertia of a grain, $\ddot{\theta}$ is the angular acceleration, $\mathbf{g}$ is the gravitational acceleration, $\mathbf{F_f}$ and $\mathbf{T}_f$ are respectively the hydrodynamic forces and torque. The equation can be solved numerically by an explicit numerical integration, such as central difference scheme.

The interaction between the soil grains and the solid grains with the walls are dealt with DEM technique. To solve the coupled DEM–LBM, the hydrodynamic force exerted on the soil grain and the static buoyancy force are considered by reducing the gravitational acceleration to $(1-\rho/rho_s)\mathbf{g}$, where ρ_s is the density of the grains. When taking into account all forces acting on an element, the dynamic equations of DEM can be expressed as

$$m\mathbf{a} + c\mathbf{v} = \mathbf{F_c} + \mathbf{F_f} + m\mathbf{g}\,, \tag{5.57}$$

where $\mathbf{F}_c$ denotes the total contact forces from other elements and/or the walls, and c is a damping coefficient. The term $c\boldsymbol{v}$ represents a viscous force that accounts for the effect of all possible dissipation forces in the system including energy lost during the collision between particles. Considering a linear contact model

$$\mathbf{F}_c = k_n\delta\,, \tag{5.58}$$

where k_n is the normal stiffness and δ is the overlap, the critical time step associated with the explicit integration is determined as (He et al., 1997)

$$\Delta t_{cr} = 2(\sqrt{1+\xi^2} - \xi)/\omega\,, \tag{5.59}$$

where $\omega = \sqrt{k_n/m}$ is the local contact natural frequency and $\xi = c/2m\omega$ is the critical damping ratio. the actual time step used for the integration of the Discrete Element equations is

$$\Delta t_D = \lambda\Delta t_{cr}\,. \tag{5.60}$$

The time step factor λ is chosen to be around 0.1 to ensure both stability and accuracy (He et al., 1997).

When combining the Discrete Element modelling of the grain interaction with the LB formulation, an issue arises. There are now two time steps: Δt for the fluid flow and Δt_D for

the particles. Since Δt_D is normally smaller than Δt, Δt_D is slightly reduced to a new value Δt_s so that Δt and Δt_s have an integer ration n_s

$$\Delta t_s = \frac{\Delta t}{n_s} \qquad (n_s = [\Delta t / \Delta t_D] + 1)\,. \tag{5.61}$$

This basically results in a sub-cycling time integration for the Discrete Element part. At every step of the fluid computation, n_s sub-steps of integration are performed for the Discrete Element Method (5.57) using the time step Δt_s. The hydrodynamic force $\mathbf{F}_f$ is unchanged during the sub-cycling.

5.4.1 Particle Kissing

In multiphase flows, fundamental mechanisms of fluid – grain and grain – grain interactions are very important for accurately predicting the flow behaviours. The sedimentation of two circular grains serves as the simplest problem to study these two types of interactions, and many experimental and numerical studies have been carried out to investigate the behaviour of sedimentation of circular particles (Komiwes et al., 2005; Wang et al., 2014). Fortes (1987) observed experimentally that in the sedimentation of two particles in a Newtonian fluid inside a vertical channel, the two particles would undergo the draft, kissing and tumbling (DKT) phenomenon.

The draft, kiss and tumbling effect may be observed when two particles fall under the action of gravity in a viscous Newtonian fluid. The *draft*: grain 2 is first placed within the hydrodynamic drag above grain 1. As the hydrodynamic drag of grain 1 is a depression zone, grain 2 is attracted inside. The *kiss*: grain 2 increases its vertical velocity until it touches grain 1. The horizontal velocity of grain 1 increases and its vertical velocity decreases below that of grain 2. *Tumbling*: grain 2 having the same horizontal velocity and higher vertical velocity than grain 1, overtakes grain 1.

LBM-DEM simulation of two grains under gravity in a viscous Newtonian fluid reproduces the Draft, Kiss and Tumbled effect (see figure 5.14). They are in agreement with the experimental description of the Draft, Kiss and Tumbled effect. For better understanding of the DKT effect, the time history of three distances between the particles (normalised to the diameter of the grain D) i.e., the difference in the transverse coordinates δ_x/D and longitudinal coordinates δ_y/D of the two grain centres, and the gap between the two surfaces $\delta = \sqrt{{\delta_x}^2 + {\delta_y}^2} - 1$ (see figure 5.15c).

As shown in figure 5.14, grain 1 trails grain 2. As grain 2 approaches the depression zone, corresponding to negative fluid pressure behind grain 1, the velocity of the trailing grain increases as grains approach closer, this is in agreement with the experimental description of

the Draft. Grain 2 increases its vertical velocity more than grain 1 until it touches grain 1. The kiss happens at normalised time $(t/\sqrt{D/g}) = 25$. At this stage, the gap δ between the particles is zero, the actual gap is about one lattice spacing for the LBM collision model. After this time, the vertical velocity of grain 1 decreases and its horizontal velocity increases as the grains tumble. At this stage, the grains still remain in contact, i.e., the gap remains unchanged $\delta = 0$. Subsequently, the two grains separate and move away from each other. Figure 5.15b shows that the terminal velocities of the two grains are in good agreement with the terminal velocity of a single grain found by an independent simulation and calculated using the empirical Schiller and Nauman formula (Komiwes et al., 2005).

5.5 GP-GPU Implementation

Graphics Processing Unit (GPU) is a massively multi-threaded architecture that is widely used for graphical and now non-graphical computations. Today's GPUs are general purpose processors with support for accessible programming interface. The main advantage of GPUs is their ability to perform significantly more floating point operations (FLOPs) per unit time than a CPU. General Purpose computations on GPUs (GPGPUs) often achieve speedups of orders of magnitude in comparison with optimised CPU implementations.

In the present study Nvidia GE Force GTX 580 GPU with 512 cuda cores is used. A GPU consists of several *Streaming Multiprocessors* (SMs). Each SM contains 32 CUDA processors. Each CUDA processor has a fully pipelined integer arithmetic logic unit (ALU) and floating point unit (FPU). The FPU complies with the IEEE 754-2008 industry standard for floating-point arithmetic, capable of double precision computations. The SM schedules work in groups of 32 threads called warps. Each SM features two warp schedulers and two instruction dispatch units, allowing two warps to be issued and executed concurrently. Each thread has access to both L1 and L2 caches, which improves the performance for programs with random memory access.

Many-core processors are promising platforms for intrinsically parallel algorithms such as the Lattice Boltzmann Method (LBM). Since the global memory for GPU devices shows high latency and LBM is data intensive, the memory access pattern is an important issue for achieving good performances. Whenever possible, global memory loads and stores should be coalescent and aligned, but the propagation phase in LBM can lead to frequent misaligned memory accesses. Also, the data transfer between the host and the device is very expensive. In the present study, the LBM implementation follows carefully chosen data transfer schemes in global memory.

There are three ways to accelerate GP-GPU applications: (a) Using 'drop-in' libraries,

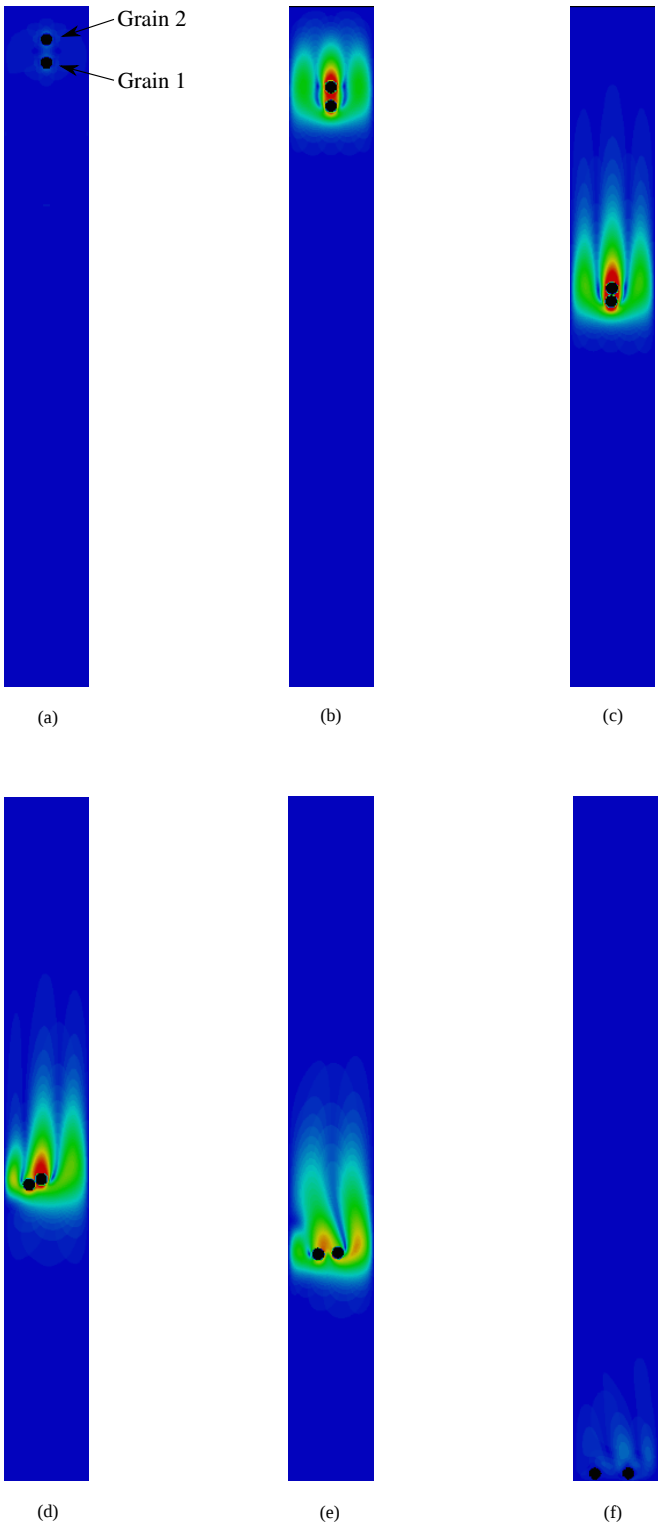
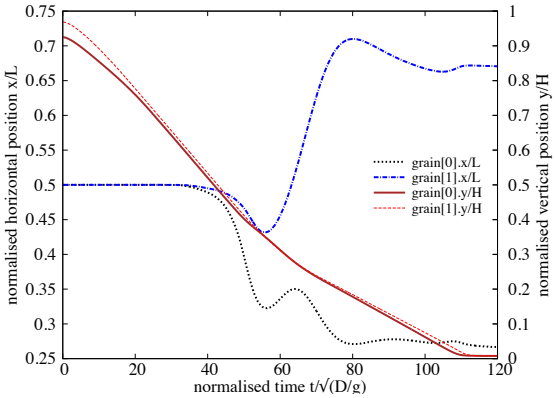
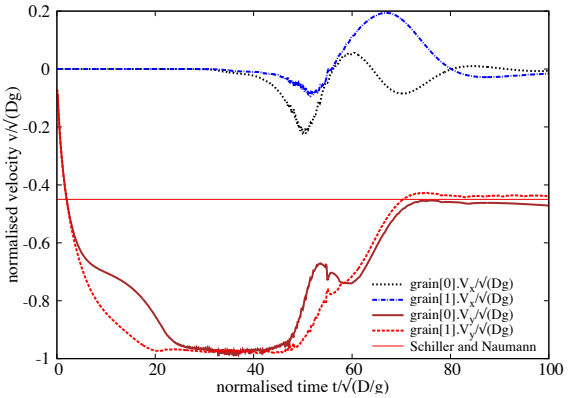


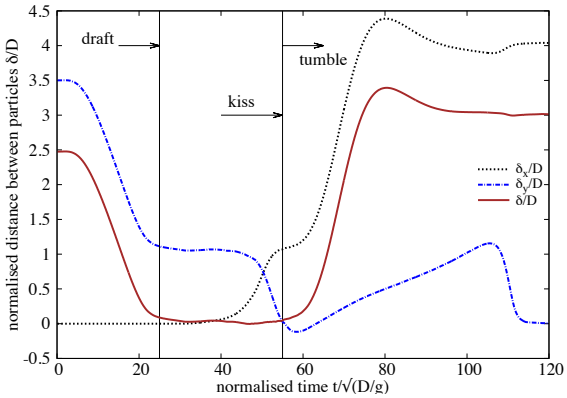
Figure 5.14 Time series of draft, kiss and tumble of two grains during sedimentation in a viscous fluid.



(a) transverse and longitudinal position



(b) normalised velocity



(c) distance between grains

Figure 5.15 Time history of two circular grains during sedimentation.

(b) using directives by exposing parallelism, and (c) using dedicated GP-GPU programming languages. OpenACC (Open Accelerators) is an open GPU directives programming standard for parallel computing on heterogeneous CPU/GPU systems. Unlike conventional GPU programming languages, such as CUDA, OpenACC uses directives to specify parallel regions in the code and performance tuning works on exposing parallelism. OpenACC targets a host-directed execution model where the sequential code runs on a conventional processor and computationally intensive parallel pieces of code (kernels) run on an accelerator such as a GPU (see figure 5.16).

The original GPGPU LBM – DEM code was implemented using OpenACC API v1.0, which was released in November 2011. The current implementation uses OpenACC API v2.0a (OpenACC-Members, 2013) and has two compute constructs, the kernels construct and the parallel construct. LBM – DEM implementation predominantly uses the OpenACC gang and vector parallelism. The LBM – DEM code runs sequential and computationally less intensive functions on the CPU, OpenMP multi-threading is used when possible. Computationally intensive functions are converted to a target accelerator specific GPU parallel code. Schematics of a hybrid CPU/GPU system is shown in Figure 5.16.

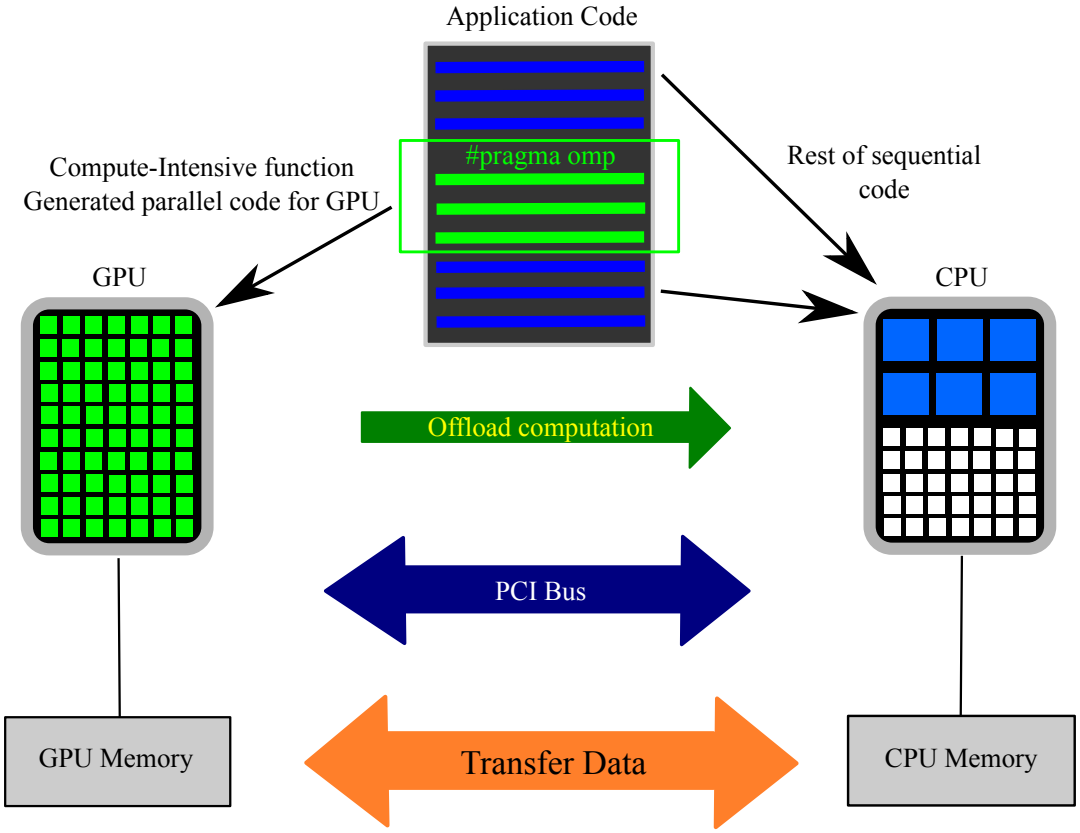


Figure 5.16 Schematics of a hybrid CPU/GPU system.

Generally speaking, the occupancy rate of the SPs, i.e. the ratio between the number of threads run and the maximum number of executable threads, is an important aspect to take into consideration for the optimisation of a CUDA kernel. Even though a block may only be run on a single SM,it is possible to execute several blocks concurrently on the same SM. Hence tuning the execution grid layout allows one to increase the occupancy rate.Nevertheless,reaching the maximum occupancy is usually not possible, as the threads executed in parallel on one SM have to share the available registers (Obrecht and Kuznik, 2011).

The loop nests in a kernels construct are converted by the compiler into parallel kernels that run efficiently on GPU. There are three steps to this process. The first is to identify the loops that can be executed in parallel. The second is to map that abstract loop parallelism onto concrete hardware parallelism. In OpenACC terms, gang parallelism maps to grid-level parallelism, (equivalent to a CUDA blockIdx) and vector parallelism maps to thread-level parallelism (equivalent to a CUDA threadIdx). The compiler normally maps a single loop across multiple levels of parallelism using strip-mining. Finally, in step three the compiler generates and optimizes the actual code to implement the selected parallelism mapping.

An OpenACC parallel construct creates a number of parallel threads that immediately begin executing the body of the parallel construct redundantly. When a thread reaches a work-sharing loop, that thread will execute some subset of the loop iterations, depending on the scheduling policy as specified by the program or at the runtime. The code generation and optimization for a parallel construct is essentially the same as for the kernels construct. A key difference is that unlike a kernels construct, the entire parallel construct becomes a single target parallel operation, aka a single CUDA kernel. Both constructs allow for automatic vectorization within the loops (Wolfe, 2012).

An excerpt from the LBM-DEM code showing the OpenACC GPU implementation of the hydrodynamic force compuration is presented in Listing 5.1. The kernels loop construct tells the compiler to map the body of the following loop into an accelerator kernel. The GPU implementation uses a two-dimensional grid splitting the iterations across both the vector and gang modes. The kernel is mapped to a vector mode mapped (aligned with CUDA threadidx%x) with a vector length (thread block size) of 128. The kernel was also mapped to gang parallelism, aligned to CUDA blockidx%x, to avoid partition camping by mapping the stride-1 loop to the x dimension. The compiler strip-mines the loop into chunks of 256 iterations, mapping the 256 iterations of a chunk in vector mode across the threads of a CUDA thread block, and map the n/256 chunks in gang mode across the thread blocks of the CUDA grid. The consecutive iterations (i and i+1), which refer to contiguous array elements (fhf[i] and fhf[i+1]), are mapped to adjacent CUDA threads in the same thread block, to optimize for coalesced memory accesses.

Listing 5.1 OpenACC GPU implementation of the hydrodynamic force computation.

```
// OpenACC Kernels copy data between the host and the device
#pragma acc kernels
copyout(fhf1[0:nbgrains], fhf2[0:nbgrains], fhf3[0:nbgrains])
copyin(obst[0:][0:], g[0:nbgrains], ey[0:], f[0:][0:][0:], ex[0:])
// Create individual threads for each DEM grain
#pragma acc parallel for
for (i=0; i<nbgrains; i++) {
  // Reset hydrodynamic forces to zero at the start of time step
  fhf1[i]=fhf2[i]=fhf3[i]=0.;
  // Iterate through all lattice nodes
  for (y=0; y<ly; y++) {
    for (x=0; x<lx; x++) {
      if (obst[x][y]==i) {
        // generate code to execute the iterations in parallel with
        // no synchronization
        #pragma acc for independent
        for (iLB=1; iLB<Q; iLB++) {
          next_x=x+ex[iLB];
          next_y=y+ey[iLB];
          if (iLB<=half) halfq=half;
          else halfq= -half;
          if (obst[next_x][next_y]!=i) {
            fnx=(f[x][y][iLB+halfq]+f[next_x][next_y][iLB])*ex[iLB+halfq];
            fny=(f[x][y][iLB+halfq]+f[next_x][next_y][iLB])*ey[iLB+halfq];
            fhf1[i]=fhf1[i]+fnx;
            fhf2[i]=fhf2[i]+fny;
            fhf3[i]=fhf3[i]-fnx*(y-(g[i].x2-wall_bottom_y)/dx)
                    +fny*(x-(g[i].x1-wall_left_x)/dx);
          }
        }
      }
    }
  }
}
```

Memory transaction optimisation is more important than computation optimisation. Registers do not give rise to any specific problem apart from their limited amount. Global memory, being the only one accessible by both the CPU and the GPU, is a critical path as it suffers from high latency. However, this latency is mostly hidden by the scheduler which stalls inactive warps until data are available. For data intensive LBM, this aspect is generally the limiting factor (Obrecht and Kuznik, 2011). To optimise global memory transactions, the memory access is coalesced and aligned, as explained above. The memory transactions between the host and the target through a PCI bus are kept to a minimum.

A two-dimensional fluid – grain system, which consists of 7.2 million LBM nodes and 2500 DEM grains is used to demonstrate the ability of GPU parallel code. The wall time required to compute 100 iterations of the given LBM – DEM problem is compared for executions running on a single CPU thread, multi-threaded CPU (using OpenMP) and the GP-GPU implementations (see table 5.3). The speed-up of parallel implementations are measured against the single CPU thread. OpenMP parallelised multi-threaded CPU execution running on 12 cores achieved a speed-up of 13.5x in comparison to a serial implementation. GP-GPU implementation using OpenACC delivered an impressive 126x speed-up in comparison to a single thread CPU execution and about 10 times quicker than a CPU parallel code. In other words, a simulation that would have ordinarily taken 126 days to compute, could now be finished in a day using a GPU.

Table 5.3 GPU vs CPU parallelisation

Execution	Computational Time (s)	Speedup
CPU 1 OpenMP thread	2016	–
CPU 2 OpenMP threads	1035	1.5 x
CPU 4 OpenMP threads	660	3.0 x
CPU 12 OpenMP threads	150	13.5 x
GPU OpenACC	16	126.0 x

[#] Wall time for 100 iteration for 7.2 Million LBM nodes and 2500 DEM grains.

[*] CPU OpenMP threads - 6 core Intel Xeon @ 3.3GHz

[†] GPU threads - GeForce GTX 580 - 512 CUDA cores

Scalability is an important criterion when developing high-performance computing codes. Scalability in GPUs is measured in terms of SM utilisation. It is important to distribute sufficient work to all SMs such that on every cycle the warp scheduler has at least one warp eligible to issue and instruction. In general, sufficient warps on each SM should be available to hide instruction and memory latency and to provide a variety of instruction types to fill the execution pipeline. Figure 5.17 shows the scalability of GP-GPU implementation as the LBM

domain size is increased from 500,000 to 9 million nodes. With increase in LBM nodes the computation time increases linearly with a slope of about 2, which shows that the LBM–DEM implementation algorithm scales nicely with the domain size.

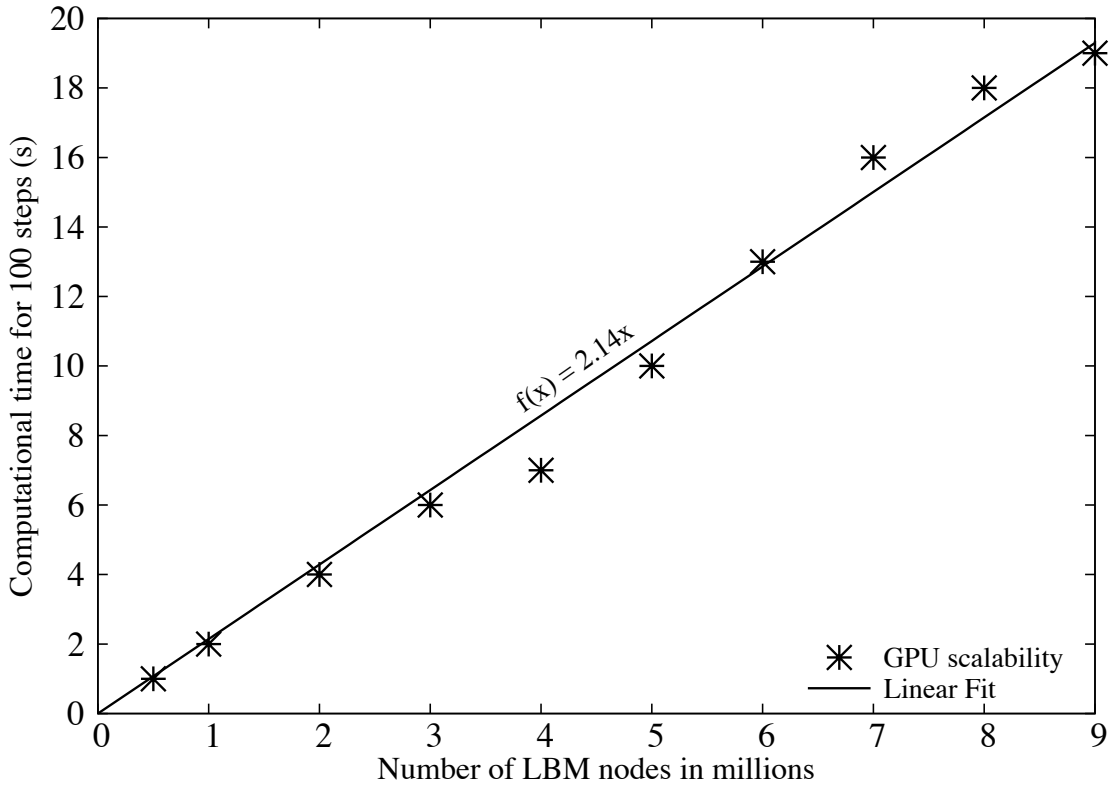


Figure 5.17 GPU scalability with increase in LBM nodes

References

Aidun, C., Lu, Y., and Ding, E. (1998). Direct analysis of particulate suspensions with inertia using the discrete Boltzmann equation. *Journal of Fluid Mechanics*, 373(-1):287–311.

Bathe, K. and Zhang, H. (2004). Finite element developments for general fluid flows with structural interactions. *International Journal for Numerical Methods in Engineering*, 60(1):213–232.

Breuer, M., Bernsdorf, J., Zeiser, T., and Durst, F. (2000). Accurate computations of the laminar flow past a square cylinder based on two different methods: lattice-Boltzmann and finite-volume. *Journal of Heat and Fluid Flow*, 21(2):186–196.

Capecelatro, J. and Desjardins, O. (2013). An Euler–Lagrange strategy for simulating particle-laden flows. *Journal of Computational Physics*, 238(0):1–31.

Chen, S. and Doolen, G. G. D. (1998). Lattice Boltzmann method for fluid flows. *Annual review of fluid mechanics*, 30(1):329–364.

Cook, B., Noble, D., and Williams, J. (2004). A direct simulation method for particle-fluid systems. *Engineering Computations*, 21(2/3/4):151–168.

Du, R., Shi, B., and Chen, X. (2006). Multi-relaxation-time lattice Boltzmann model for incompressible flow. *Physics Letters A*, 359(6):564–572.

Durst, F., Ray, S., Unsal, B., and Bayoumi, O. (2005). The development lengths of laminar pipe and channel flows. *Journal of fluids engineering*, 127:1154.

Feng, Y. T., Han, K., and Owen, D. R. J. (2007). Coupled lattice Boltzmann method and discrete element modelling of particle transport in turbulent fluid flows: Computational issues. *International Journal for Numerical Methods in Engineering*, 72(9):1111–1134.

Fortes, A. (1987). Nonlinear mechanics of fluidization of beds of spherical particles. *... of Fluid Mechanics*.

Frisch, U. and Kolmogorov, A. (1995). *Turbulence: the legacy of AN Kolmogorov*. Cambridge Univ Pr.

Han, K., Feng, Y., and Owen, D. (2007a). Coupled lattice Boltzmann and discrete element modelling of fluid–particle interaction problems. *Computers & Structures*, 85(11-14):1080–1088.

Han, S., Zhu, P., and Lin, Z. (2007b). Two-dimensional interpolation-supplemented and Taylor-series expansion-based lattice Boltzmann method and its application. *... in Nonlinear Science and Numerical Simulation*, 12(7):1162–1171.

He, X. and Luo, L. L.-S. (1997a). Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation. *Physical Review E*, 56(6):6811.

He, X. and Luo, L.-S. (1997b). A priori derivation of the lattice Boltzmann equation. *Physical Review E*, 55(6):R6333–R6336.

He, X., Zou, Q., Luo, L. S., and Dembo, M. (1997). Analytic solutions of simple flows and analysis of nonslip boundary conditions for the lattice Boltzmann BGK model. *Journal of Statistical Physics*, 87(1):115–136.

Iglberger, K., Thurey, N., and Rude, U. (2008). Simulation of moving particles in 3D with the Lattice Boltzmann method. *Computers & Mathematics with Applications*, 55(7):1461–1468.

Komiwes, V., Mege, P., Meimon, Y., and Herrmann, H. (2005). Simulation of granular flow in a fluid applied to sedimentation. *Granular Matter*, 8(1):41–54.

Ladd, A. J. (1994). Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *Journal of Fluid Mechanics*, 271:285–309.

Ladd, A. J. C. and Verberg, R. (2001). Lattice-Boltzmann Simulations of Particle-Fluid Suspensions. *Journal of Statistical Physics*, 104(5):1191–1251.

Liu, S. H., Sun, D. A., and Wang, Y. (2003). Numerical study of soil collapse behavior by discrete element modelling. *Computers and Geotechnics*, 30(Compendex):399–408.

Mei, R., Shyy, W., Yu, D., and Luo, L.-S. (2000). Lattice Boltzmann Method for 3-D Flows with Curved Boundary. *Journal of Computational Physics*, 161(2):680–699.

Noble, D. and Torczynski, J. (1998). A lattice-Boltzmann method for partially saturated computational cells. *International Journal of Modern Physics C-Physics and Computer*, 9(8):1189–1202.

Obrecht, C. and Kuznik, F. (2011). A new approach to the lattice Boltzmann method for graphics processing units. *Computers & Mathematics ...*.

OpenACC-Members (2013). The OpenACC Application Programming Interface Version 2.0 (June, 2013; Corrected: August, 2013). Technical report, OpenMP Standard Group.

Pan, C., Luo, L., and Miller, C. (2006). An evaluation of lattice Boltzmann schemes for porous medium flow simulation. *Computers & fluids*, 35(8-9):898–909.

Smagorinsky, J. (1963). General circulation experiments with the primitive equations. *Monthly weather review*, 91(3):99–164.

Succi, S. (2001). *The lattice Boltzmann equation for fluid dynamics and beyond*. Oxford University Press.

Succi, S., Foti, E., and Higuera, F. (1989). Three-Dimensional Flows in Complex Geometries with the Lattice Boltzmann Method. *Europhysics Letters (EPL)*, 10(5):433–438.

Sukop, M. and Thorne, D. (2006). *Lattice Boltzmann modeling: An introduction for geoscientists and engineers*. Springer Verlag.

Sun, X., Sakai, M., and Yamada, Y. (2013). Three-dimensional simulation of a solid–liquid flow by the DEM–SPH method. *Journal of Computational Physics*, 248(0):147–176.

Wang, L., Guo, Z., and Mi, J. (2014). Drafting, kissing and tumbling process of two particles with different sizes. *Computers & Fluids*.

Willis, A., Peixinho, J., Kerswell, R., and Mullin, T. (2008). Experimental and theoretical progress in pipe flow transition. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1876):2671.

Wolfe, M. (2012). PGInsider August 2012: OpenACC Kernels and Parallel Constructs.

Xiong, Q., Madadi-Kandjani, E., and Lorenzini, G. (2014). A LBM–DEM solver for fast discrete particle simulation of particle–fluid flows. *Continuum Mechanics and Thermodynamics*.

Yu, D., Mei, R., Luo, L., and Shyy, W. (2003). Viscous flow computations with the method of Lattice Boltzmann equation. *Progress in Aerospace Sciences*, 39(5):329–367.

Yu, H., Girimaji, S., and Luo, L. (2005). Lattice Boltzmann simulations of decaying homogeneous isotropic turbulence. *Physical Review E*, 71(1):016708.

Zhong, Q. and Olson, M. (1991). Finite element–algerbraic closure analysis of turbulent separated-reattaching flow around a rectangular body. *Computer methods in applied mechanics and engineering*, 85(2):131–150.

Zhou, H., Mo, G., Wu, F., Zhao, J., Rui, M., and Cen, K. (2012). GPU implementation of lattice Boltzmann method for flows with curved boundaries. *Computer Methods in Applied Mechanics and Engineering*, 225-228(null):65–73.

Ziegler, D. (1993). Boundary conditions for Lattice Boltzmann simulations. *Journal of Statistical Physics*, 71(5):1171–1177.

Zou, Q. and He, X. (1997). On pressure and velocity boundary conditions for the Lattice Boltzmann BGK model. *Physics of Fluids*, 9(6):1591–1598.