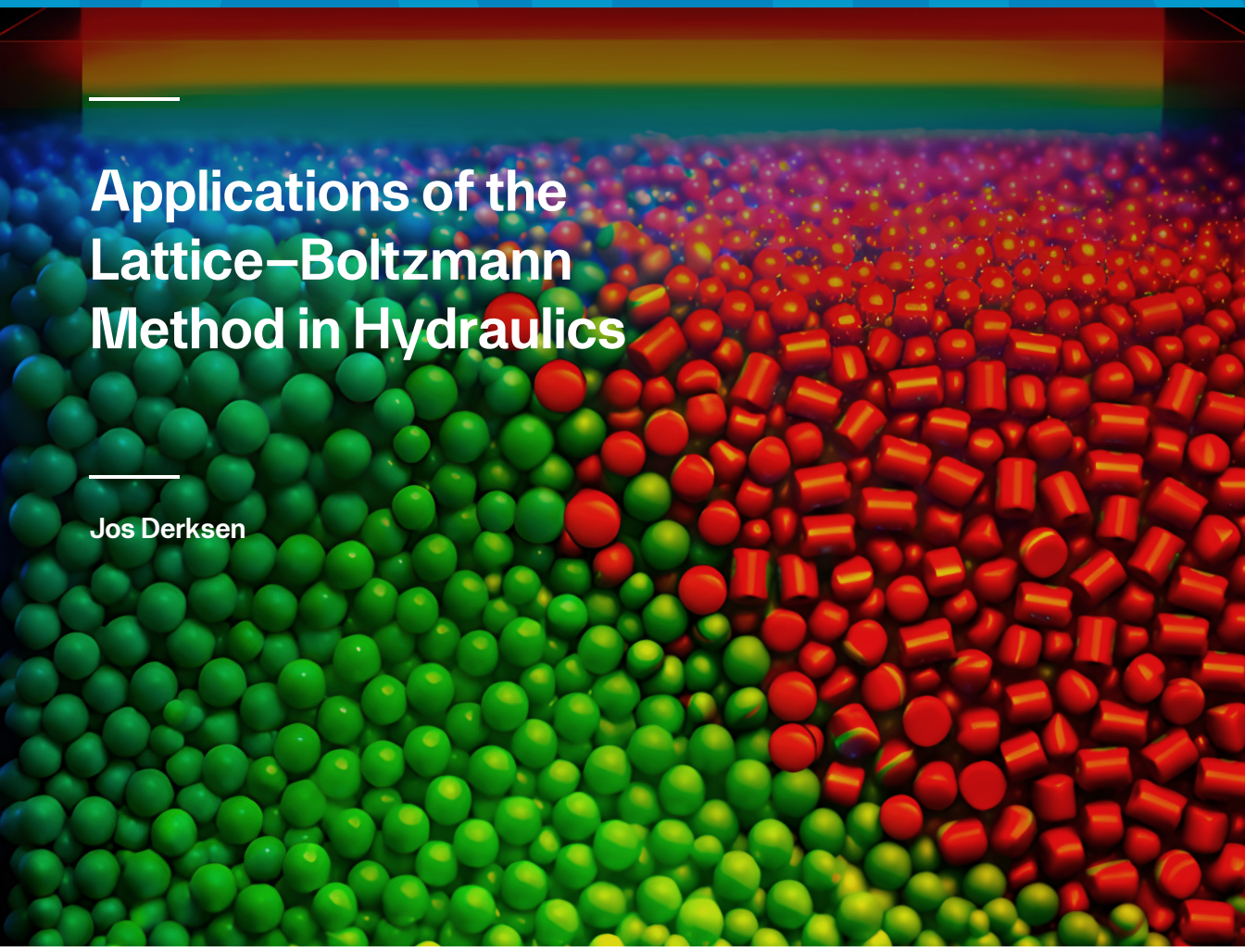
WATER MONOGRAPHS

#WaterMonographs



Applications of the Lattice–Boltzmann Method in Hydraulics

Jos Derksen

Supported by

三峡大学
CHINA THREE GORGES UNIVERSITY

OPEN ACCESS

IAHR AIRH International Association for Hydro-Environment Engineering and Research
Hosted by Spain Water and IWHR, China

IAHR.org

Applications of the Lattice-Boltzmann Method in Hydraulics

Jos J. Derksen

Applications of the Lattice-Boltzmann Method in Hydraulics

Jos J. Derksen
University of Aberdeen, UK

IAHR WATER MONOGRAPH SERIES

Series Editor
Damien Violeau
LNHE, EDF R&D, France
Saint-Venant Hydraulics Laboratory, Ecole des Ponts ParisTech, France
E-mail: damien.violeau@edf.fr

Cover: Erosion of granular beds

ISBN (online version): 978-90-836443-0-1
ISBN (printed version): 978-90-835589-9-8
ISSN 2959-7978

doi: 10.64697/IAHR_Watermonograph_006

Typeset by Clemens Lode, DE in in LaTeX.
Cover designed by Tres Estudio Creativo, Spain

Published by:
IAHR — International Association for Hydro-Environment Engineering and Research

Madrid Office

Paseo Bajo Virgen del Puerto 3, 28005, Madrid, SPAIN
Tel: +34 913357908; Fax: +34 913357935

Beijing Office

A-1 Fuxing Road, Haidian District, 100038, Beijing, CHINA
Tel: +86 1068781128; Fax: +86 1068781890

The International Association for Hydro-Environment Engineering and Research (IAHR), founded in 1935, is a worldwide independent organisation of engineers and water specialists working in fields related to the hydro-environmental sciences and their practical application. Activities range from river and maritime hydraulics to water resources development and eco-hydraulics, through to ice engineering, hydro-informatics and continuing education and training. IAHR stimulates and promotes both research and it's application, and by so doing strives to contribute to sustainable development, the optimisation of the world's water resources management and industrial flow processes. IAHR accomplishes its goals by a wide variety of member activities including: working groups, research agenda, congresses, specialty conferences, workshops and short courses; Journals, Monographs and Proceedings; by involvement in international programmes such as UNESCO, WMO, IDNDR, GWP, ICSU, and by co-operation with other water-related (inter)national organisations.

The monitoring of publishing ethics is a major aspect of editorial and peer-review publications. IAHR is a well-established and long-standing organization whose primary mission is to foster knowledge exchange amongst its community and stakeholder base. Accordingly, IAHR takes its policies on ethics seriously.

The IAHR Water Monograph *Applications of the Lattice-Boltzmann Method in Hydraulics* has been published under the IAHR Publications Ethics and Malpractice Statement. Additionally, IAHR publications follow the Ethical Guidelines and Codes of Conduct provided by the Committee of Publication Ethics (COPE).

About the IAHR Water Monograph Series

The *IAHR Water Monograph Series* is a recent addition to IAHR's long-standing portfolio of publications, which includes peer-reviewed journals, magazines, conference proceedings, white papers, and books. Since its founding in 1935, IAHR has been dedicated to advancing and sharing knowledge that supports progress in hydro-environment engineering and research.

IAHR Water Monographs are medium-length publications-typically between 50 and 150 pages–that bridge the gap between journal articles and full-length books. They are designed to consolidate and communicate authoritative knowledge on specific hydro-environment topics. Each monograph may summarise existing research, address knowledge gaps, or present recent advances in theory, methods, and practice. Topics commonly include physical processes, measurement techniques, theoretical developments, numerical modelling, engineering applications, and relevant historical or cultural contexts. Written in a concise, accessible, and well-illustrated format, Water Monographs serve as valuable resources for both specialists and those newly entering the field.

To ensure academic rigour and credibility, each monograph undergoes a structured peer-review process overseen by the IAHR Task Force on Water Monographs. This Task Force is responsible for issuing calls, evaluating proposals, and appointing editors. Its current members are:

- **Damien Violeau**, EDF, France (Chair)
- **Claudia Adduce**, Roma Tre University, Italy
- **Donatella Termini**, University of Palermo, Italy
- **Ioan Nistor**, University of Ottawa, Canada
- **Massimo Guerrero**, Università di Bologna-DICAM, Italy
- **Pengzhi Lin**, Sichuan University, China
- **Robert Ettema**, Colorado State University, USA
- **Vladimir Nikora**, University of Aberdeen, UK

Once a proposal is approved by the Task Force, an Editor is appointed to work closely with the author(s) throughout the preparation of the manuscript. When the draft is ready for review, three independent experts, and recognised in the relevant field, are invited to assess the work. Their reviews are shared with the authors for revision. Upon satisfactory revision, the Editor makes the

final decision on acceptance. Reviewers are acknowledged in the published monograph, and their names are disclosed to the authors at the end of the process.

All IAHR Water Monographs are published as **Open Access**, ensuring free and unrestricted access to readers worldwide. This is made possible thanks to the support of IAHR and institutional sponsors committed to advancing hydro-environment knowledge. Sponsors are recognised through the inclusion of their logo on the monograph cover; however, they have no involvement in the editorial or peer-review process.

Through this series, IAHR reaffirms its mission to support the global hydro-environment community by making high-quality, targeted knowledge openly and equitably available.

Damien Violeau
Chair of IAHR Water Monograph Series

Table of Contents

Acknowledgement

The author gratefully acknowledges the thoughtful comments and suggestions of the reviewers. They have significantly improved the quality and clarity of this monograph.

Reviewers

This monograph was peer-reviewed by:

Professor Ayurzana Badarch,
School of Civil Engineering and Architecture,
Mongolian University of Science and Technology

Professor Michele La Rocca,
Department of Engineering,
Roma Tre University

One anonymous reviewer.

Nomenclature

Roman

a
radius

a_2, a_3
bending load in the 2 and 3 directions

$\mathbf{c_i}$
discrete velocity set

c_s
speed of sound

c_S
Smagorinsky constant

D
half channel width

d
diameter

E
specific energy

EI
bending stiffness

e
specific kinetic energy in random motion

$\mathbf{F}$
force

f
distribution function

f_i
discrete distribution function

f^{eq}
equilibrium distribution function

$f_i^{(n)}$
n-th order expansion of discrete distribution function

f_i^*
post-collision discrete distribution function

$\mathbf{f}$
body force

$\mathbf{f_s}$
fluid solid interaction force

f
frequency

G
interpolation function

g
gravitational acceleration

h
bed height

$\mathbf{I}$
moment of inertia tensor

k
(soft sphere) spring constant

k^n
lubrication force parameter

k
wavenumber

L
size of lid-driven cavity

$L \times W \times H$
domain size

ℓ
cylinder length

ℓ
mixing length

Ma
Mach number

m
mass

M_2, M_3
bending moments in the 2 and 3 directions

N
Richardson & Zaki exponent

$\mathbf{n}$
normal direction

$nx \cdot ny \cdot nz$
computational domain size

p
pressure

q
quaternion

q_0, $\mathbf{q}$
scalar and vector part of quaternion

R
gas constant

Re
Reynolds number

$\mathbf{r}$
location vector

S_i
forcing source term

$\mathbf{S}$
rotation tensor

S
dimensionless period time

$\mathbf{s}$
tangential spring elongation vector

$\bar{\mathbf{s}}$
resolved deformation rate tensor

s
spacing

T
temperature

$\mathbf{T}$
torque

t
time

t_p
time period

$\mathbf{u}$
bulk fluid velocity

U
lid speed of cavity

u_0
wall-parallel velocity

u_τ
wall shear velocity

u_∞
single particle settling speed

u'
rms value of fluctuating velocity

V
volume

$\mathbf{v}$
random velocity

$\widetilde{v}$
rescaled velocity

w_i
weighing factor

w_2, w_3
deflection in the 2 and 3 directions

$\mathbf{w}$
solid surface velocity

$\mathbf{x}$
location

x, y, z
Cartesian coordinates

Z
impedance

$\widetilde{Z}$
rescaled impedance

Z_0, Z_∞
impedance at zero and ∞ shear

Greek

α
immersed boundary force relaxation parameter

β
$N - 2$

γ
(solid over fluid) density ratio

$\dot{\gamma}_0$
shear rate

Δ
lattice spacing

Δt
time step

$\mathbf{\Delta u}$
relative velocity of marker points

$\Delta \mathbf{x_p}$
particle displacement

δ, δ_λ
normal and lateral distance between marker points

$\delta_{\alpha\beta}$
Kronecker delta

δ_0
(soft sphere) collision parameter

δ
flow penetration distance

ε
perturbation parameter

η_{ijk}
mapping coefficients

η_K
Kolmogorov length scale

θ, θ_c
Shields parameter, critical Shields parameter

λ, δ_d
lubrication force parameters

λ
width of mapping function

μ, μ_b
viscosity and bulk viscosity

μ_{AB}
friction coefficient between particles A and B

μ
mapping function

ν
kinematic viscosity

ν_e
eddy viscosity

ξ
velocity

$\mathbf{\Pi}$
momentum flux tensor

π
fluid stress tensor

σ
stress tensor

σ'
viscous stress tensor

ρ
density

τ
relaxation time

τ
wall shear stress

φ
dimensionless solids flux

ϕ
solids volume fraction

ϕ
area packing fraction

χ
flexibility parameter

Ω
collision operator

ω
angular velocity

Indices – subscript

c
contact

D
drag

ext
external

h
hydrodynamic

IB
immersed boundary

i
discrete velocity index

int
internal (fluid)

j
marker point index

k
lattice node index

LB
lattice Boltzmann

lub
lubrication

p
particle

s
solids

α, β, γ
Cartesian coordinate direction

Indices – superscript

$+$
wall units

c
continuous

Glossary

Term	Explanation
bounce-back rule	rule for bouncing of distribution functions on a boundary achieving no-slip
bulk viscosity	dilation / expansion viscosity
Chapman-Enskog analysis	multiscale expansion of transport equations
collision operator	term in the Boltzmann equation related to (molecular) collisions
deviatoric stress	traceless part of the viscous stress tensor
direct numerical simulation (DNS)	simulation that aims at directly solving the time-dependent Navier-Stokes equation in three dimensions
equilibrium distribution	velocity distribution at equilibrium (no-flow) conditions
immersed boundary (IB) method	numerical method to impose off-grid boundary conditions
ghost cell	lattice cell outside the flow domain for accommodating boundary conditions
large eddy simulation (LES)	simulation that solves the low-pass filtered Navier-Stokes equation and uses a model for the sub-grid scales
lattice-Boltzmann (LB) equation	discrete version of the Boltzmann equation
lattice units	units system in terms of the lattice: unit of space is the lattice spacing, unit of time is the time step
lubrication force	viscous force between closely spaced particles due to their relative velocity
mapping function	relation between grid-based (Eulerian) and particle-based (Lagrangian) properties
marker points	collections of points representing a solid surface in the immersed boundary method
mesoscopic scale	in-between micro and macroscopic scales
monami	submerged canopy waves
particle-resolved simulation (PRS)	simulation that explicitly imposes no-slip at solid particle surfaces
quaternion	a scalar value and a three-dimensional vector used for keeping track of particle orientation
Smagorinsky model	subgrid-scale model in LES based on an eddy viscosity concept

Term	Explanation
streaming	motion of distribution function between neighbouring lattice sites
subgrid scale	scale finer than the computational grid
volume-averaged Navier-Stokes equation	volume averaged flow equations allowing for the presence of dispersed-phase particles

CHAPTER 1

Introduction

1.1 | Aim & scope

The lattice-Boltzmann (LB) method is a way of performing simulations of fluid flow and related transport processes, and in this monograph, we argue that it is a useful addition to more mainstream computational fluid dynamics (CFD) methods based on finite volumes, finite differences or finite elements. There are many reasons for using the LB method, and also many reasons for not using it. The usefulness of the LB method largely depends on the specific application, including the level of detail one is interested in resolving through simulation. It also depends on the availability of computational resources.

An understanding of the LB method, its implementation in computer code, as well as demonstrations of sample applications in the realm of hydraulics will help the reader in deciding if a specific application could benefit from the method and – if so – will hopefully enable a quick start.

The above summarizes the aim of the monograph. It is a concise guide to the LB method, its theory, its practical implementation, and its application in hydraulics and environmental fluid dynamics. The text is – I think – easy to read, however not comprehensive. It provides guidance to the extensive literature laid down in books and in archival journal papers. As for books, I would like to highlight here the relatively early (2001) work by Succi [59], and the more recent – 2017 – text due to Kruger et al [36].

1.2 | A brief history

Cellular automata date back to John von Neumann and co-workers [70]. Cellular automata are discrete computational models that evolve according to simple update rules. Despite the simplicity of the underlying rules they often exhibit complex behaviour. An archetypical example is the Game of Life as devised by Conway [26].

Lattice Gas Automata (LGA) for fluid dynamics gained significant traction through Frisch, Hasslacher and Pomeau with their paper “Lattice Gas Automata for the Navier-Stokes Equation” [25]. They proposed a discrete system and subsequently argued that – through Chapman-Enskog analysis [6] – it represents the dynamics of an isotropic Newtonian fluid. The system is fully discrete; discrete in

physical space, velocity space and in time and uses Boolean variables in the form of particle occupation states.

Replacing Boolean variables by real-valued distribution functions, as proposed by McNamara and Zanetti [43] showed significant computational advantages. The noise inherent to a Boolean representation requires high resolution in space and time to collect meaningful macroscopic statistics. This noise is largely eliminated by using real-valued distribution functions. This marks the start of the lattice-Boltzmann method.

It is important to realize by researchers and engineers familiar with CFD that the lattice-Boltzmann method can be – and has been extensively – used as a Navier-Stokes solver, although the starting point of CFD and LB are markedly different. CFD starts with continuum partial differential equations that are discretized through finite differences, volumes, or elements. Discretization then results in a set of algebraic equations, the unknowns of which represent the flow field. Such simulations are usually called "Navier-Stokes" simulations. The point of departure of the LB method is the discrete system consisting of particles (more precisely: discrete distribution functions) on a regular lattice. The interactions between the particles are designed such that expanding the system to a continuous system results in a set of partial differential equations – closely resembling the Navier-Stokes equations. Evolving the discrete system then "solves" the partial differential equations.

After its inception by McNamara and Zanetti in 1989, the lattice Boltzmann method has branched out in a variety of directions. One with perhaps less appeal in hydraulics and more in chemical engineering relates to multiphase and multicomponent flows, that is flows with deformable interfaces. This branch is dominated by two approaches: (1) the free-energy method [62]; (2) the pseudopotential method [56]. Developments in this area are in directions such as accurate descriptions of interfacial phenomena (surface tension) [9] as well as dealing with combinations of fluids with large density and/or viscosity contrasts [30].

Incorporating the immersed boundary method [49] in a lattice Boltzmann context has opened up various applications. An important one is suspension flow, i.e. flow of mixtures of fluid and solid particles as in fluvial and aeolian sediment transport. The immersed boundary method is then used to explicitly impose the no-slip condition at the surfaces of the solid particles. In these so-called particle-resolved simulations [66], the lattice spacing is smaller than the particle size by at least one order of magnitude so that the systems that can be studied this way can only contain a fairly limited number of particles [33]. An alternative that overcomes this limitation is the discrete element method (DEM) or unresolved particle approach [42], at the cost of more empiricism. As we will see later in this monograph, the lattice Boltzmann method is able to accommodate DEM as well.

When suspensions get dense, i.e. attain an appreciable solids volume fraction, it is necessary to tightly couple fluid and solids transport and solve the volume-averaged Navier-Stokes (VANS) equation rather than the NS equation to account for the volume that is displaced by the solids. Methods have been developed to extend the lattice Boltzmann method for the VANS equation [61].

The immersed boundary method equipped with immersed boundary capability can also be used for fluid-structure interaction simulations with rigid as well as with deformable solid boundaries. One example is the paper by Huang et al [29] on fish passing a tubular turbine; another is vortex shedding behind a bendable cylinder [17].

There are a number of other applications for which the lattice Boltzmann method offers a numerical procedure for problem solving. To mention a few: sound propagation [69], convection-diffusion-reaction systems [68] and – quite relevant for hydraulics – shallow water flows [75].

1.3 | Organization of the monograph

After this introductory chapter, the reader will find the following in this monograph:

Chapter 2 explains the mathematical foundation of the LB method. Getting this information across is important to appreciate the underlying assumptions. With a view to hydraulics, where the fluid dynamics is usually incompressible; it will be shown what the consequences of the (weak) compressibility of the LB method are for solving incompressible flow.

Chapter 3 deals with practical aspects of the LB method. Topics to be discussed are the use of dimensional analysis for scaling a flow problem and the use of lattice units, boundary conditions, and implementation in computer code. The latter with an emphasis on efficiency.

Chapter 4 discusses solid-liquid suspensions that in hydraulics are particularly relevant for sediment transport. In this chapter we will distinguish between particle-resolved simulations where the lattice spacing is finer than the size of the solid particles, and no-slip is imposed explicitly at the solid surfaces by means of immersed boundaries, and point particle approaches, where the flow around particles is not resolved and hydrodynamic forces and torques on particles require input from (empirical) models. We will also discuss the effect of the shape and deformability of solid particles on the flowability of a solid-liquid suspension. Only particle-resolved simulations can realistically deal with effect of particle shape on fluid-solid interaction.

In **Chapter 5** we discuss turbulent flow. The focus is on single-phase (liquid-only) flow. It is shown how subgrid-scale models can be included in the LB method so that a smooth transition

between direct numerical simulation (DNS) and large eddy simulation (LES) of turbulent flow can be achieved. We show an example of LES of an oscillatory flow.

In **Chapter 6** a brief summary is presented with the most important take-home messages.

CHAPTER 2

Fundamentals of the lattice-Boltzmann method

2.1 | Kinetic theory

2.1.1 | Introduction

Matter consists of molecules. In a gas they move around thereby colliding with one another. In a classical molecular dynamics description of a gas we follow individual molecules by solving their Newtonian equations of motion that include interaction forces between the molecules. Simulating this on a computer is called Molecular Dynamics (MD) [24].

In a continuum approach, the discrete (molecular) nature of matter is abandoned to make place for a description in terms of functions describing properties (pressure, density, velocity, temperature) that continuously depend on (three-dimensional) location in space and on time. Kinetic theory is a description in between molecular and continuous; hence the term *mesoscopic* description. We still think of molecules but rather than considering them individually we deal with the probability distribution of their properties.

2.1.2 | Distribution function

Starting point of kinetic theory is the notion of a *distribution function*. Kinetic theory is a very rich subject that we will touch upon only superficially. For further reading and for more context I recommend the book by Kauzmann [32].

Since distribution functions are also fundamental to the lattice-Boltzmann method it is worthwhile to go over some of the concepts and results from kinetic theory. For simplicity we will consider a dilute, monoatomic gas with the molecules having no internal degrees of freedom (such as vibration or rotation) and the molecules colliding elastically.

The distribution function is denoted as $f(\mathbf{x}, \xi, t)$ with $\mathbf{x}$ the location vector, ξ the velocity vector, and t time. It can be interpreted as a generalization of the density $\rho(\mathbf{x}, t)$, now also taking into account velocity. It is the density of molecules in physical space as well as in *velocity space*. The SI units of the distribution function are $[f] = \mathrm{kg} \cdot \frac{1}{m^3} \cdot \frac{1}{\left(\frac{m}{s}\right)^3}$. Density, momentum and energy follow from integrations over velocity space:

$$\rho(\mathbf{x}, t) = \iiint f(\mathbf{x}, \xi, t)\, d^3\xi \tag{2.1}$$

$$\rho(\mathbf{x}, t)\, \mathbf{u}(\mathbf{x}, t) = \iiint \xi f(\mathbf{x}, \xi, t)\, d^3\xi \tag{2.2}$$

$$\rho(\mathbf{x}, t)\, E(\mathbf{x}, t) = \frac{1}{2} \iiint |\xi|^2 f(\mathbf{x}, \xi, t)\, d^3\xi \tag{2.3}$$

As we will see, it is useful to make a distinction between random thermal motion (with velocity $\mathbf{v}(\mathbf{x}, t)$) and bulk motion of molecules (with velocity $\mathbf{u}(\mathbf{x}, t)$ as identified in Eq. 2.2): $\mathbf{v}(\mathbf{x}, t) \equiv \xi(\mathbf{x}, t) - \mathbf{u}(\mathbf{x}, t)$. The energy associated to $\mathbf{v}(\mathbf{x}, t)$ is

$$\rho(\mathbf{x}, t)\, e(\mathbf{x}, t) = \frac{1}{2} \iiint |\mathbf{v}|^2 f(\mathbf{x}, \xi, t)\, d^3\xi \tag{2.4}$$

is called internal energy. We reiterate that only translational kinetic energy is considered in the above expressions. This is due to us dealing with monoatomic gases that do not have rotational or vibrational degrees of freedom.

A quick way of finding an expression for pressure goes via the ideal gas law and the equipartition theorem. The latter relates internal energy per translational degree of freedom with temperature: $e_{dof} = \frac{1}{2}RT$ for each degree of freedom (dof). With three degrees of freedom this gets $\rho e = \frac{3}{2}\rho RT$. The usual way of writing the ideal gas law in fluid dynamics is $p = \rho RT$ so that

$$p = \rho RT = \frac{2}{3}\rho e = \frac{1}{3} \iiint |\mathbf{v}|^2 f(\mathbf{x}, \xi, t)\, d^3\xi \tag{2.5}$$

It is useful to check units at this stage. When we write $p = \rho RT$ then R is not the (universal) molar based gas constant $R_{molar} = 8.31\ \frac{J}{(\text{mol}\cdot K)}$; instead, $R = \frac{R_{molar}}{m}$ with m the molar mass of the gas. Molar mass has units kg/mol so that R has units $\frac{J}{(\text{kg}\cdot K)} = \frac{m^2}{(s^2\cdot K)}$ which should be given that $e = \frac{3}{2}RT$ and e having units $\frac{m^2}{s^2}$ (the latter from the way e is defined in Eq. 2.4).

2.1.3 | Equilibrium distribution function

If a gas is left alone for a sufficiently long time and molecules have undergone many collisions, thereby exchanging momentum, one can imagine that the distribution of velocities of the molecules reaches an equilibrium. This equilibrium is expected to be isotropic in velocity space. We assume – without lack of generality – the average velocity $\mathbf{u}$ equal to zero so that $\mathbf{v}(\mathbf{x}, t) \equiv \xi(\mathbf{x}, t)$. Isotropy in velocity space then means that only the velocity magnitude (not the direction) matters for the equilibrium distribution function: $f^{eq}(\mathbf{x}, |\mathbf{v}|, t)$. An important result of statistical mechanics is that the equilibrium distribution has the following form:

$$f^{eq}(\mathbf{x}, |\mathbf{v}|, t) = \rho \left(\frac{1}{2\pi RT} \right)^{\frac{3}{2}} e^{-\frac{|\mathbf{v}|^2}{(2RT)}} \tag{2.6}$$

Equation 2.6 goes by the name Maxwell-Boltzmann distribution [44]. It shows, as an example, that increasing the temperature will result in a wider distribution function.

2.1.4 | Evolution of the distribution function

If we take the total derivative of the distribution function with respect to time, we get

$$\frac{df}{dt} = \frac{\partial f}{\partial t}\frac{dt}{dt} + \frac{\partial f}{\partial x_\beta}\frac{dx_\beta}{dt} + \frac{\partial f}{\partial \xi_\beta}\frac{d\xi_\beta}{dt} \tag{2.7}$$

Where we use index notation for the location (x_β) and velocity (ξ_β) vector components and the summation convention: sum over repeated Greek indices (for example $\frac{\partial f}{\partial x_\beta}\frac{dx_\beta}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial z}\frac{dz}{dt}$). In Eq. 2.7, $\frac{dt}{dt} = 1$, $\frac{dx_\beta}{dt} = \xi_\beta$, $\frac{d\xi_\beta}{dt}$ is an acceleration per unit volume and therefore $\frac{d\xi_\beta}{dt} = \frac{F_\beta}{\rho}$ with F_β a force component. Collisions are a reason for a change of the distribution function since they re-distribute momentum. Combining these notions leads to the Boltzmann equation

$$\frac{\partial f}{\partial t} + \xi_\beta \frac{\partial f}{\partial x_\beta} + \frac{F_\beta}{\rho}\frac{\partial f}{\partial \xi_\beta} = \Omega(f) \tag{2.8}$$

where $\Omega(f)$ is the collision operator. It (only) depends on the distribution function, i.e. on the way mass is distributed over physical and velocity space. The Boltzmann equation (Eq. 2.8) can be interpreted as an advection equation with forces and collisions as source terms. In a molecular collision, mass is not lost, so that one property of the collision operator is

$$\iiint \Omega(f) d^3\xi = 0 \tag{2.9}$$

If we also assume that total momentum and total energy is conserved in collisions this implies.

$$\iiint \xi\Omega(f) d^3\xi = \mathbf{0} \tag{2.10}$$

$$\iiint |\xi|^2 \Omega(f) d^3\xi = 0 \tag{2.11}$$

A lot of physics goes into the collision operator as it involves a wide spectrum of collision scenarios and intermolecular forces. To avoid (too much) complexity, simple heuristic approaches have been proposed that prove remarkably useful. Respect for conservation laws and the notion that collisions will eventually drive a system to equilibrium have led to the Bhatnagar, Gross and Krook (BGK) collision operator [3]:

$$\Omega(f) = -\frac{1}{\tau}\left(f - f^{eq}\right) \tag{2.12}$$

The time constant τ determines how fast we relax to equilibrium. For a force-free, homogeneous system $f(\xi, t) = f^{eq}(\xi) + \left[f(\xi, t=0) - f^{eq}(\xi)\right] e^{-\frac{t}{\tau}}$. As expected and as we will see when discussing the lattice-Boltzmann method, τ determines the transport coefficients, most importantly – for fluid mechanics – the viscosity.

2.1.5 | Transport equations

It is time to relate the somewhat esoteric concept of the distribution function with concepts more familiar to fluid dynamics. This we do by deriving transport equations from the Boltzmann equation (Eq. 2.8).

In the first place, we integrate Eq. 2.8 over velocity space: $\frac{\partial}{\partial t}\iiint f d^3\xi + \frac{\partial}{\partial x_\beta}\iiint \xi_\beta f d^3\xi + \frac{F_\beta}{\rho}\iiint \frac{\partial f}{\partial \xi_\beta} d^3\xi = \iiint \Omega(f) d^3\xi$. We realize that (see Eq. 2.1) $\iiint f d^3\xi = \rho$; also (Eq. 2.2) $\iiint \xi_\beta f d^3\xi = \rho u_\beta$; and (Eq. 2.9) $\iiint \Omega(f) d^3\xi = 0$. The term with the force ($\frac{F_\beta}{\rho}\iiint \frac{\partial f}{\partial \xi_\beta} d^3\xi$) is a little less straightforward. We argue that this term is equal to zero based on integration by

parts. As a reminder, consider two functions h and g that have derivatives h' and g'. Then $\int hg' = hg|_{-\infty}^{\infty} - \int h'g$. Applying this $\iiint 1 \cdot \frac{\partial f}{\partial \xi_\beta} d^3\xi = 1 \cdot f|_{-\infty}^{\infty} - 0 = 0$. As a result, integration of the Boltzmann equation over velocity space gets us the continuity equation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_\beta)}{\partial x_\beta} = 0 \tag{2.13}$$

(once more: we are using the summation convention so that Eq. 2.13 actually reads $\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_x)}{\partial x} + \frac{\partial (\rho u_y)}{\partial y} + \frac{\partial (\rho u_z)}{\partial z} = 0$)

A momentum balance equation is obtained when we multiply the Boltzmann equation by ξ_α and then integrate over velocity space: $\frac{\partial}{\partial t} \iiint \xi_\alpha f d^3\xi + \frac{\partial}{\partial x_\beta} \iiint \xi_\alpha \xi_\beta f d^3\xi + \frac{F_\beta}{\rho} \iiint \xi_\alpha \frac{\partial f}{\partial \xi_\beta} d^3\xi = \iiint \xi_\alpha \Omega(f) d^3\xi$. First term on the left-hand side: $\frac{\partial \rho u_\alpha}{\partial t}$; the integral in the second term, we will call – for reasons that will hopefully become clear later – the momentum flux tensor: $\Pi_{\alpha\beta} \equiv \iiint \xi_\alpha \xi_\beta f d^3\xi$ so that the second term gets $\frac{\partial \Pi_{\alpha\beta}}{\partial x_\beta}$; the integral in the third term is $\iiint \xi_\alpha \frac{\partial f}{\partial \xi_\beta} d^3\xi = \xi_\alpha f|_{-\infty}^{\infty} - \iiint \frac{\partial \xi_\alpha}{\partial \xi_\beta} f d^3\xi = -\rho\delta_{\alpha\beta}$ with $\delta_{\alpha\beta}$ the Kronecker delta ($\delta_{\alpha\beta} = 1$ if $\alpha = \beta$; $\delta_{\alpha\beta} = 0$ if $\alpha \neq \beta$). The right-hand side is zero given Eq. 2.10 (momentum conservation in collisions). Eventually the momentum balance has the form

$$\frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial (\Pi_{\alpha\beta})}{\partial x_\beta} = F_\alpha \tag{2.14}$$

(as before: Eq. 2.14 uses the summation convention; writing out – as an example – the x-component of Eq. 2.14: $\frac{\partial \rho u_x}{\partial t} + \frac{\partial (\Pi_{xx})}{\partial x} + \frac{\partial (\Pi_{xy})}{\partial y} + \frac{\partial (\Pi_{xz})}{\partial z} = F_x$)

In Section 2.2 we hinted at splitting the velocity of molecules in average and thermal velocities: $\xi(\mathbf{x}, t) = \mathbf{u}(\mathbf{x}, t) + \mathbf{v}(\mathbf{x}, t)$. Component wise: $\xi_\alpha = u_\alpha + v_\alpha$. Then $\Pi_{\alpha\beta} = \iiint (u_\alpha + v_\alpha)(u_\beta + v_\beta) f d^3\xi = u_\alpha u_\beta \iiint f d^3\xi + u_\alpha \iiint v_\beta f d^3\xi + u_\beta \iiint v_\alpha f d^3\xi + \iiint v_\alpha v_\beta f d^3\xi$; $u_\alpha u_\beta \iiint f d^3\xi = \rho u_\alpha u_\beta$; $u_\alpha \iiint v_\beta f d^3\xi = 0$; $u_\beta \iiint v_\alpha f d^3\xi = 0$. The last term we define as the stress tensor: $\iiint v_\alpha v_\beta f d^3\xi = -\sigma_{\alpha\beta}$. Putting this all together results in what is known as the *Cauchy momentum equation*:

$$\frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial (\rho u_\alpha u_\beta)}{\partial x_\beta} = \frac{\partial \sigma_{\alpha\beta}}{\partial x_\beta} + F_\alpha \tag{2.15}$$

The stress tensor is usually written as the sum of viscous stress and pressure [2]: $\sigma_{\alpha\beta} = \sigma'_{\alpha\beta} - p\delta_{\alpha\beta}$. As an aside, if we would assume the fluid to be an ideal gas in equilibrium ($f = f^{eq}$ the Maxwell-Boltzmann distribution Eq. 2.6), evaluating the integral $\iiint v_\alpha v_\beta f^{eq} d^3\xi$ results in $\rho RT\delta_{\alpha\beta}$ which is

equal to the pressure. Therefore, for an ideal gas in equilibrium, $\sigma_{\alpha\beta} = -p\delta_{\alpha\beta}$. This implies that viscous stresses are the result of non-equilibrium effects.

Viscous stresses are *modelled* in terms of fluid deformation: $\sigma'_{\alpha\beta} = \mu\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right) + \zeta\delta_{\alpha\beta}\frac{\partial u_\gamma}{\partial x_\gamma}$ where μ and ζ are material properties. It is more common [2] to write this as

$$\sigma'_{\alpha\beta} = \mu\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha} - \frac{2}{3}\delta_{\alpha\beta}\frac{\partial u_\gamma}{\partial x_\gamma}\right) + \mu_b\delta_{\alpha\beta}\frac{\partial u_\gamma}{\partial x_\gamma} \tag{2.16}$$

The term with the coefficient μ (which is viscosity) is a traceless stress tensor (*deviatoric* stress); the term with μ_b (which is called bulk viscosity) contains normal viscous stresses due to volume changes (expansion and contraction). It should be clear that the bulk viscosity μ_b is related to the parameters μ and ζ: $\mu_b = \frac{2}{3}\mu + \zeta$.

With the notions regarding the stress tensor (its division into pressure and viscous stress, the latter divided into a deviatoric part and a normal part), the Cauchy momentum equation (Eq. 2.15) can be written as the Navier-Stokes equation:

$$\frac{\partial \rho u_\alpha}{\partial t} + \frac{\partial\left(\rho u_\alpha u_\beta\right)}{\partial x_\beta} = -\frac{\partial p}{\partial x_\alpha} + \frac{\partial}{\partial x_\beta}\left[\mu\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right) + \left(\mu_b - \frac{2}{3}\mu\right)\delta_{\alpha\beta}\frac{\partial u_\gamma}{\partial x_\gamma}\right] + F_\alpha \tag{2.17}$$

In hydraulics applications fluid flow is mostly incompressible, i.e. the fluids involved have constant density. This implies that the continuity (Eq. 2.13) reduces to $\frac{\partial u_\beta}{\partial x_\beta} = 0$. As a consequence, the incompressible Navier-Stokes equation is simpler than its general form:

$$\rho\frac{\partial u_\alpha}{\partial t} + \rho u_\beta\frac{\partial u_\alpha}{\partial x_\beta} = -\frac{\partial p}{\partial x_\alpha} + \frac{\partial}{\partial x_\beta}\left[\mu\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right)\right] + F_\alpha \tag{2.18}$$

2.2 | Lattice-Boltzmann method

2.2.1 | Introduction

We will now build a discrete system by using the concepts from kinetic theory as discussed above. The core concept of kinetic theory is the distribution function $f(\mathbf{x}, \xi, t)$ with $\mathbf{x}$ location (in general in three-dimensional space), ξ velocity (in general also in three dimensions) and t time. Discretization involves discretization in time, space *and velocity*. The discretization is – to some extent and at

some level of abstraction – straightforward. Relating the discrete version of the Boltzmann equation to the Navier-Stokes equation and deriving the discrete version of the equilibrium distribution function are, however, quite elaborate. What we will do in this Section 2.2 is first present the discrete Boltzmann equation and the discrete version of the equilibrium distribution function, and explain the lattice-Boltzmann algorithm. After having explained the algorithm we then relate the lattice-Boltzmann equation to the Navier-Stokes equation (Section 2.2.4 Chapman-Enskog analysis) and discuss the relation between the continuous and discrete distribution function (Section 2.2.5 Discrete equilibrium distribution function). This is mostly based on [7].

2.2.2 | Discretization

We define a discrete and finite set of velocities $\mathbf{c_i} = (c_{ix}, c_{iy}, c_{iz})$ and indicate the distribution function as $f_i(\mathbf{x}, t)$ which is now understood as that the density of molecules at $\mathbf{x}$ and t traveling with velocity $\mathbf{c_i}$. Integrations over velocity space (as in Eqs. 2.1 and 2.2) turn into summations over the finite set of velocities

$$\rho(\mathbf{x}, t) = \sum_i f_i(\mathbf{x}, t) \tag{2.19}$$

$$\rho(\mathbf{x}, t)\,\mathbf{u}(\mathbf{x}, t) = \sum_i \mathbf{c_i} f_i(\mathbf{x}, t) \tag{2.20}$$

In the situations we will be considering, discretization in space implies that locations where $f_i(\mathbf{x}, t)$ is defined are positioned on a simple cubic lattice with the cubes having a side length Δ. Discretization in time implies that $f_i(\mathbf{x}, t)$is defined at moments separated by a time interval Δt.

In this monograph we use the practice of *lattice units*. This means that we choose the units of space and time such that $\Delta = 1$ and $\Delta t = 1$ respectively. In the Chapter 3 we will see how to perform (unit) conversion between physical systems and numerical lattice-Boltzmann (LB) systems.

There are many options when it comes to velocity sets. Velocity sets for solving the Navier-Stokes equation are categorized as DdQq with d the dimensionality of space (1, 2 or 3), and q the number of velocities. The more common velocity sets are D2Q9 and D3Q19; there also are D1Q3, D3Q15, D3Q27 (and many more). Velocity sets are defined by the velocity vectors $\mathbf{c_i}$ and by a weighing factor per velocity w_i. Finally, as will emerge from analysis described in Section 2.2.4, each velocity set comes with a constant c_s that relates pressure with density: $p = c_s^2 \rho$. This is an (isothermal) ideal gas law

with $\frac{\partial p}{\partial \rho} = c_s^2$ and c_s is therefore interpreted as the speed of sound. In the most common velocity sets, $c_s = \sqrt{\frac{1}{3}}\frac{\Delta}{\Delta t}$ (i.e. $c_s = \sqrt{\frac{1}{3}}$ in lattice units).

The discretizations in time, space and velocity lead to the *lattice-Boltzmann equation* [59]:

$$f_i\left(\mathbf{x} + \mathbf{c_i}\Delta t, t + \Delta t\right) = f_i\left(\mathbf{x}, t\right) + \Omega_i\left(\mathbf{x}, t\right) \tag{2.21}$$

Applying lattice units ($\Delta = 1$ and $\Delta t = 1$):

$$f_i\left(\mathbf{x} + \mathbf{c_i}, t + 1\right) = f_i\left(\mathbf{x}, t\right) + \Omega_i\left(\mathbf{x}, t\right) \tag{2.22}$$

In the remainder of this chapter, we will – for convenience but without much lack of generality – work in two dimensions and adopt the D2Q9 velocity set with velocity vectors

$$\begin{aligned} \mathbf{c_0} &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \mathbf{c_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \mathbf{c_2} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \quad \mathbf{c_3} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \mathbf{c_4} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} \\ \mathbf{c_5} &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \mathbf{c_6} = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad \mathbf{c_7} = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \quad \mathbf{c_8} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \end{aligned} \tag{2.23}$$

and

$$w_0 = \frac{4}{9} \quad w_{1-4} = \frac{1}{9} \quad w_{5-8} = \frac{1}{36} \tag{2.24}$$

and $c_s = \sqrt{\frac{1}{3}}$ [7].

With the definition of the velocity set, Figure 2.1 shows that at the end of a time step (which is applying Eq. 2.22 to all nodes on the lattice) a distribution function that started from a lattice node $\mathbf{x}$ ends up at $\mathbf{x} + \mathbf{c_i}$ which is a neighbouring lattice node (except for the distribution with $i = 0$ which stays where it was).

The simplest, yet effective, collision operator is the Bhatnagar, Gross and Krook (BGK) collision operator [3]; see Eq. 2.12. In discrete form it reads

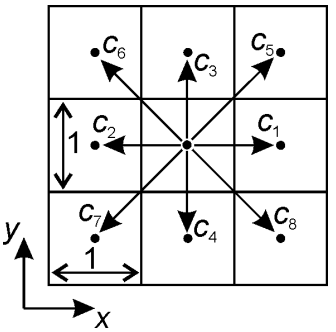


Figure 2.1 | The D2Q9 velocity set with *xy* coordinate system and velocity numbering according to Eq. 2.23.

$$\Omega_i(\mathbf{x}, t) = \Omega_i(f) = -\frac{1}{\tau}\left(f_i - f_i^{eq}\right) \tag{2.25}$$

The f (without index) in $\Omega_i(f)$ means that the collision operator depends on all distribution functions $f_{0\cdots8}$ at the location-time combination $(\mathbf{x}, t)$. As we will argue later in these notes, the discrete version of the equilibrium distribution function reads

$$f_i^{eq} = w_i\rho\left[1 + \frac{u_\alpha c_{i\alpha}}{c_s^2} + \frac{\left(u_\alpha c_{i\alpha}\right)^2}{2c_s^4} - \frac{u_\alpha u_\alpha}{2c_s^2}\right] \tag{2.26}$$

(Note the summation convention: summation over repeated Greek indices). It contains the macroscopic flow variables density ρ and velocity components u_α.

It is worthwhile performing a few reality checks. We will perform these for the D2Q9 velocity set as defined in Eqs. 2.23 and 2.24 with $c_s^2 = \frac{1}{3}$.

(1) Mass conservation requires $\sum_i \Omega_i = 0$. Since $\sum_i f_i = \rho$, it then should be that also $\sum_i f_i^{eq} = \rho$. Writing out $\sum_i f_i^{eq}$ for D2Q9 and applying the *summation convention* gets $\sum_i w_i\rho\left[1 + 3\left(u_x c_{ix} + u_y c_{iy}\right) + \frac{9}{2}\left(u_x c_{ix} + u_y c_{iy}\right)^2 - \frac{3}{2}\left(u_x^2 + u_y^2\right)\right] = \rho\left(1 - \frac{3}{2}u_x^2 - \frac{3}{2}u_y^2\right)\sum_i w_i + 3\rho u_x\sum_i w_i c_{ix} + 3\rho u_y\sum_i w_i c_{iy} + \frac{9}{2}\rho u_x^2\sum_i w_i c_{ix}^2 + \frac{9}{2}\rho u_y^2\sum_i w_i c_{iy}^2 + 9\rho u_x u_y\sum_i w_i c_{ix}c_{iy}$ Realize that (see Eqs. 2.23 and 2.24) $\sum_i w_i = 1$; $\sum_i w_i c_{i\alpha} = 0$; $\sum_i w_i c_{i\alpha}c_{i\beta} = \frac{1}{3}\delta_{\alpha\beta}$ then indeed $\sum_i f_i^{eq} = \rho$.

(2) Momentum conservation over the collision operation requires $\sum_i \Omega_i c_{i\alpha} = 0$. Since $\sum_i f_i c_{i\alpha} = \rho u_\alpha$, it should be that also $\sum_i f_i^{eq} c_{i\alpha} = \rho u_\alpha$. Again, writing out $\sum_i f_i^{eq} c_{i\alpha}$ and using the symmetry properties of the lattice results in $\sum_i f_i^{eq} c_{i\alpha} = \rho u_\alpha$.

The collision operator $\Omega_i(f)$ only depends on local properties, i.e. the set of distribution functions f_i at the current location at the current instant in time. This is because the expression for $\Omega_i(f)$ contains f_i and f_i^{eq} (Eq. 2.26), f_i^{eq} depends on ρ and on u_α, and in turn ρ and u_α can be written in terms of f_i (Eqs. 2.19 and 2.20). This is relevant from a computational perspective: updating the lattice-Boltzmann equation only requires local data.

The Chapman-Enskog analysis [36] relates the lattice-Boltzmann equation with the Navier-Stokes equation. We will go through this analysis in Section 2.2.4. Important results of the analysis are in the first place that the kinematic viscosity of the lattice-Boltzmann fluid relates to the relaxation time according to

$$\nu = c_s^2 \left(\tau - \frac{\Delta t}{2} \right) \tag{2.27}$$

In lattice units and with $c_s^2 = \frac{1}{3}$ this becomes $\nu = \frac{2\tau - 1}{6}$. In the second place the Chapman-Enskog expansion establishes the relation between density and pressure alluded to before: $p = c_s^2 \rho$.

2.2.3 | The lattice-Boltzmann algorithm – time stepping

A lattice-Boltzmann algorithm applies $f_i(\mathbf{x} + \mathbf{c_i}, t+1) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t)$. In practice, each time step in a lattice-Boltzmann algorithm is broken up in two parts: *collision* and *streaming*.

The collision part executes

$$f_i^*(\mathbf{x}, t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t) \tag{2.28}$$

with $f_i(\mathbf{x}, t)$ the pre-collision distribution function and $f_i^*(\mathbf{x}, t)$ the post-collision distribution function. As discussed above: executing Eq. 2.28 is a *local operation*. For each lattice node $\mathbf{x}$ it only involves information at $\mathbf{x}$: since $\Omega_i(\mathbf{x}, t) = -\frac{1}{\tau}(f_i - f_i^{eq})$ the information required for a collision is $f_i(\mathbf{x}, t)$ and $f_i^{eq}(\mathbf{x}, t)$; the latter depends on $\rho(\mathbf{x}, t)$ and $\mathbf{u}(\mathbf{x}, t)$ (see Eq. 2.26) which in turn can be determined using $f_i(\mathbf{x}, t)$ (see Eq. 2.19 and 2.12). Local operations are easily run on a parallel compute platforms through domain decomposition: divide the overall simulation domain in subdomains and assign each subdomain to a compute core. The collision operation (Eq. 2.28) then does not require any communication between compute cores.

In the streaming part of the LB algorithm we execute

$$f_i\left(\mathbf{x}+\mathbf{c_i}, t+1\right)=f_i^*\left(\mathbf{x}, t\right) \tag{2.29}$$

i.e. we shift ("stream") the post-collision distribution f_i^* from its current location $\mathbf{x}$ to its neighbouring location $\mathbf{x}+\mathbf{c_i}$ thereby advancing time by one time unit. This is a simple, however non-local operation. In parallel computing the streaming step requires communication over the edges of the subdomains each of which is assigned to a compute core.

2.2.4 | Chapman-Enskog analysis

The aim of the analysis explained in this section is to "derive" the Navier-Stokes equation (Eq. 2.17) from the lattice-Boltzmann equation (Eq. 2.21). We will be using the LB equation in lattice units and the BGK collision operator. At some point in the derivation we will be needing the specific velocity set for which we will use the D2Q9 set (Eqs. 2.23 and 2.24), as well as the specific expression for the equilibrium distribution for which we take Eq. 2.26.

LB equation:

$$f_i\left(\mathbf{x}+\mathbf{c_i}, t+1\right)-f_i\left(\mathbf{x}, t\right)=-\frac{1}{\tau}\left[f_i\left(\mathbf{x}, t\right)-f_i^{eq}\left(\mathbf{x}, t\right)\right] \tag{2.30}$$

We expand the distribution function around its equilibrium and introduce the small parameter ε:

$$f_i=f_i^{eq}+\varepsilon f_i^{(1)}+\varepsilon^2 f_i^{(2)}+\ldots \tag{2.31}$$

Given that (see above) $\sum_i\left(f_i-f_i^{eq}\right)=0$ and $\sum_i c_{i\alpha}\left(f_i-f_i^{eq}\right)=0$, it has to be that $\sum_i f_i^{(n)}=0$ and $\sum_i c_{i\alpha} f_i^{(n)}=0$ for $n \geq 1$.

We expand the derivatives in the small parameter ε:

$$\frac{\partial f_i}{\partial t}=\varepsilon\frac{\partial^{(1)}}{\partial t} f_i+\varepsilon^2\frac{\partial^{(2)}}{\partial t} f_i+\ldots; \quad c_{i\alpha}\frac{\partial f_i}{\partial x_\alpha}=\varepsilon c_{i\alpha}\frac{\partial^{(1)}}{\partial x_\alpha} f_i \tag{2.32}$$

The $\partial^{(1)}$ and $\partial^{(2)}$ are not derivatives by themselves, they represent contributions to derivatives with certain orders of magnitude in ε.

Now perform a 2nd order Taylor expansion of the LB equation (Eq. 2.30).

$$\left(\frac{\partial}{\partial t}+c_{i\alpha}\frac{\partial}{\partial x_\alpha}\right)f_i+\frac{1}{2}\left(\frac{\partial}{\partial t}+c_{i\alpha}\frac{\partial}{\partial x_\alpha}\right)^2 f_i=-\frac{1}{\tau}\left(f_i-f_i^{eq}\right) \tag{2.33}$$

Then apply the operator $\frac{1}{2}\left(\frac{\partial}{\partial t}+c_{i\alpha}\frac{\partial}{\partial x_\alpha}\right)$ to Eq. 2.33. This gets us

$$\frac{1}{2}\left(\frac{\partial}{\partial t}+c_{i\alpha}\frac{\partial}{\partial x_\alpha}\right)^2 f_i+hot=-\frac{1}{2\tau}\left(\frac{\partial}{\partial t}+c_{i\alpha}\frac{\partial}{\partial x_\alpha}\right)\left(f_i-f_i^{eq}\right) \tag{2.34}$$

where *hot* stands for *higher-order terms* (of order 3 in this case) that we subsequently neglect. Subtracting Eq. 2.34 from 2.33 (and discarding the *hot*) results in

$$\left(\frac{\partial}{\partial t}+c_{i\alpha}\frac{\partial}{\partial x_\alpha}\right)f_i=-\frac{1}{\tau}\left(f_i-f_i^{eq}\right)+\frac{1}{2\tau}\left(\frac{\partial}{\partial t}+c_{i\alpha}\frac{\partial}{\partial x_\alpha}\right)\left(f_i-f_i^{eq}\right) \tag{2.35}$$

We substitute the expansions Eqs. 2.31 and 2.32 and group the terms by orders of ε. Each group establishes one equation since ε can take any (small) value.

Terms of order ε

$$\left(\frac{\partial^{(1)}}{\partial t}+c_{i\alpha}\frac{\partial^{(1)}}{\partial x_\alpha}\right)f_i^{eq}=-\frac{1}{\tau}f_i^{(1)} \tag{2.36}$$

Terms of order ε^2

$$\left(\frac{\partial^{(1)}}{\partial t}+c_{i\alpha}\frac{\partial^{(1)}}{\partial x_\alpha}\right)f_i^{(1)}+\frac{\partial^{(2)}}{\partial t}f_i^{eq}=-\frac{1}{\tau}f_i^{(2)}+\frac{1}{2\tau}\left(\frac{\partial^{(1)}}{\partial t}+c_{i\alpha}\frac{\partial^{(1)}}{\partial x_\alpha}\right)f_i^{(1)} \tag{2.37}$$

Rewrite Eq. 2.37:

$$\frac{\partial^{(2)}}{\partial t}f_i^{eq}+\left(1-\frac{1}{2\tau}\right)\left(\frac{\partial^{(1)}}{\partial t}+c_{i\alpha}\frac{\partial^{(1)}}{\partial x_\alpha}\right)f_i^{(1)}=-\frac{1}{\tau}f_i^{(2)} \tag{2.38}$$

Add up Eqs. 2.36$\times\varepsilon$ and 2.38$\times\varepsilon^2$:

$$\varepsilon\frac{\partial^{(1)}}{\partial t}f_i^{eq}+\varepsilon^2\frac{\partial^{(2)}}{\partial t}f_i^{eq}+\varepsilon c_{i\alpha}\frac{\partial^{(1)}}{\partial x_\alpha}f_i^{eq}+\varepsilon^2\left(1-\frac{1}{2\tau}\right)\left(\frac{\partial^{(1)}}{\partial t}+c_{i\alpha}\frac{\partial^{(1)}}{\partial x_\alpha}\right)f_i^{(1)}=-\varepsilon\frac{1}{\tau}f_i^{(1)}-\varepsilon^2\frac{1}{\tau}f_i^{(2)} \tag{2.39}$$

Revert some expanded derivatives to their original form:

$$\frac{\partial}{\partial t}f_i^{eq}+c_{i\alpha}\frac{\partial}{\partial x_\alpha}f_i^{eq}+\varepsilon^2\left(1-\frac{1}{2\tau}\right)\left(\frac{\partial^{(1)}}{\partial t}+c_{i\alpha}\frac{\partial^{(1)}}{\partial x_\alpha}\right)f_i^{(1)}=-\varepsilon\frac{1}{\tau}f_i^{(1)}-\varepsilon^2\frac{1}{\tau}f_i^{(2)} \tag{2.40}$$

If we sum Eq. 2.40 over i, only the first two terms survive (since $\sum_i f_i^{(n)}=0$ and $\sum_i c_{i\alpha}f_i^{(n)}=0$ for $n\geq 1$) and we end up with the continuity equation

$$\frac{\partial}{\partial t}\rho+\frac{\partial}{\partial x_\alpha}\rho u_\alpha=0 \tag{2.41}$$

Now multiply Eq. 2.40 by $c_{i\beta}$ and sum over i:

$$\frac{\partial}{\partial t}\left(\rho u_\beta\right)+\frac{\partial}{\partial x_\alpha}\sum_i c_{i\alpha}c_{i\beta}f_i^{eq}+\left(1-\frac{1}{2\tau}\right)\frac{\partial}{\partial x_\alpha}\sum_i \varepsilon c_{i\alpha}c_{i\beta}f_i^{(1)}=0 \tag{2.42}$$

Given the first term, this is a momentum balance. For interpretation of the other two terms we need to involve the velocity set and the equilibrium distribution (Eqs. 2.23, 2.24 & 2.26). This allows us to derive (with a view to the 2nd term in Eq. 2.42)

$$\sum_i c_{i\alpha}c_{i\beta}f_i^{eq}=\rho u_\alpha u_\beta+\rho c_s^2\delta_{\alpha\beta} \tag{2.43}$$

At this stage consider the term in Eq. 2.42 that contains $\sum_i c_{i\alpha}c_{i\beta}f_i^{(1)}$.

For this we go back to Eq. 2.36, repeated here: $\left(\frac{\partial^{(1)}}{\partial t}+c_{i\alpha}\frac{\partial^{(1)}}{\partial x_\alpha}\right)f_i^{eq}=-\frac{1}{\tau}f_i^{(1)}$, since it contains $f_i^{(1)}$. Multiply Eq. 2.36 by $c_{i\beta}c_{i\gamma}$ and sum over i:

$$\frac{\partial^{(1)}}{\partial t}\sum_i c_{i\alpha}c_{i\beta}f_i^{eq}+\frac{\partial^{(1)}}{\partial x_\alpha}\sum_i c_{i\alpha}c_{i\beta}c_{i\gamma}f_i^{eq}=-\frac{1}{\tau}\sum_i c_{i\alpha}c_{i\beta}f_i^{(1)} \tag{2.44}$$

This allows expressing $\sum_i c_{i\alpha}c_{i\beta}f_i^{(1)}$ in terms of the equilibrium distribution and the set of velocities. The result is

$$\sum_i c_{i\alpha} c_{i\beta} f_i^{(1)} = -\rho c_s^2 \tau \left[\frac{\partial^{(1)} u_\alpha}{\partial x_\beta} + \frac{\partial^{(1)} u_\beta}{\partial x_\alpha} \right] + \tau \frac{\partial^{(1)}}{\partial x_\gamma} \left(\rho u_\alpha u_\beta u_\gamma \right) \tag{2.45}$$

Substitute these findings (Eqs. 2.43 and 2.45) in the momentum balance Eq. 2.42:

$$\frac{\partial}{\partial t}\left(\rho u_\beta\right) + \frac{\partial}{\partial x_\alpha}\left(\rho u_\alpha u_\beta + \rho c_s^2 \delta_{\alpha\beta}\right) + \left(1 - \frac{1}{2\tau}\right) \frac{\partial}{\partial x_\alpha} \left(-\rho c_s^2 \tau \varepsilon \left[\frac{\partial^{(1)} u_\alpha}{\partial x_\beta} + \frac{\partial^{(1)} u_\beta}{\partial x_\alpha} \right] + \tau \varepsilon \frac{\partial^{(1)}}{\partial x_\gamma} \left(\rho u_\alpha u_\beta u_\gamma \right) \right) = 0$$

This can be simplified: turn $\varepsilon \frac{\partial^{(1)}}{\partial x_\beta}$ to $\frac{\partial}{\partial x_\beta}$; $\frac{\partial}{\partial x_\alpha}\left(\rho c_s^2 \delta_{\alpha\beta}\right) = c_s^2 \frac{\partial \rho}{\partial x_\beta}$. As a result

$$\frac{\partial}{\partial t}\left(\rho u_\beta\right) + \frac{\partial}{\partial x_\alpha}\left(\rho u_\alpha u_\beta\right) = -c_s^2 \frac{\partial \rho}{\partial x_\beta} + \left(\tau - \frac{1}{2}\right) \frac{\partial}{\partial x_\alpha} \left(\rho c_s^2 \left[\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha} \right] - \frac{\partial}{\partial x_\gamma} \left(\rho u_\alpha u_\beta u_\gamma \right) \right) \tag{2.46}$$

We finally argue that the term containing the product of three velocities $\frac{\partial}{\partial x_\gamma}\left(\rho u_\alpha u_\beta u_\gamma\right)$ is negligibly small, at least in the low Mach number limit with $\mathrm{Ma}^2 \equiv \frac{u_\alpha u_\alpha}{c_s^2}$ (the Mach number and a discussion on compressibility will follow in Chapter 3).

We write Eq. 2.46 in a (compressible) Navier-Stokes-like form

$$\frac{\partial}{\partial t}\left(\rho u_\beta\right) + \frac{\partial}{\partial x_\alpha}\left(\rho u_\alpha u_\beta\right) = -\frac{\partial p}{\partial x_\beta} + \nu \frac{\partial}{\partial x_\alpha} \left(\rho \left[\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha} - \frac{2}{3} \delta_{\alpha\beta} \frac{\partial u_\gamma}{\partial x_\gamma} \right] + \frac{2}{3} \rho \delta_{\alpha\beta} \frac{\partial u_\gamma}{\partial x_\gamma} \right) \tag{2.47}$$

with $p = c_s^2 \rho$ the anticipated ideal-gas relation between density and pressure, and $\nu = c_s^2 \left(\tau - \frac{1}{2}\right)$ the again anticipated relation between relaxation time and kinematic viscosity. Another observation is that this LB fluid has a bulk viscosity of two-thirds of the (shear) viscosity.

2.2.5 | Discrete equilibrium distribution function

A cornerstone of kinetic theory is the Maxwell-Boltzmann equilibrium distribution function, Eq. 2.6, repeated here as Eq. 2.48.

$$f^{eq}\left(\mathbf{x}, |\mathbf{v}|, t\right) = \rho \left(\frac{1}{2\pi RT} \right)^{\frac{3}{2}} e^{-\frac{|\mathbf{v}|^2}{(2RT)}} \tag{2.48}$$

In the Chapman-Enskog exercise of the previous sub-section, we made use of a discrete version of the equilibrium distribution function, Eq. 2.26 (repeated here as Eq. 2.49)

$$f_i^{eq} = w_i \rho \left[1 + \frac{u_\alpha c_{i\alpha}}{c_s^2} + \frac{\left(u_\alpha c_{i\alpha}\right)^2}{2c_s^4} - \frac{u_\alpha u_\alpha}{2c_s^2} \right] \tag{2.49}$$

In this section, it will be shown how – globally, not very rigorously – the two expressions (Eqs. 2.48 and 2.49) are related.

As a first step we realize that the velocity $\mathbf{v}$ in Eq. 2.48 is the random (thermal) velocity of the molecules. In Section 2.1.2 the velocity ξ was decomposed in a bulk velocity $\mathbf{u}$ and thermal velocity $\mathbf{v}$: $\xi = \mathbf{v} + \mathbf{u}$. We make Eq. 2.48 dimensionless and consider isothermal conditions. As a result of these two steps the Maxwell-Boltzmann distribution becomes $f^{eq} = \rho \left(\frac{1}{2\pi}\right)^{\frac{3}{2}} e^{-\frac{|(\xi - \mathbf{u})|^2}{2}}$. If we assume that the bulk speed is much lower than the thermal speed we can perform a Taylor expansion of f^{eq} around $\mathbf{u} = \mathbf{0}$ and break it off at second order:

$$f^{eq} \approx \rho\omega\left(\xi\right) \left[1 + \xi_\alpha u_\alpha + \frac{1}{2} u_\alpha u_\beta \left(\xi_\alpha \xi_\beta - \delta_{\alpha\beta}\right) \right] \tag{2.50}$$

where we – for brevity – introduce the function $\omega\left(\xi\right) \equiv \left(\frac{1}{2\pi}\right)^{\frac{3}{2}} e^{-\frac{|\xi|^2}{2}}$. In Eq. 2.50 we still have a continuous velocity ξ, so that the next step is to discretize velocity space. We can do this in the following way

$$f_i^{eq} = \rho w_i \left[1 + \xi_{i\alpha} u_\alpha + \frac{1}{2} u_\alpha u_\beta \left(\xi_{i\alpha} \xi_{i\beta} - \delta_{\alpha\beta}\right) \right] \tag{2.51}$$

where we have replaced continuous $\omega\left(\xi\right)$ by discrete w_i and have replaced continuous ξ_α by discrete $\xi_{i\alpha}$. We now can determine w_i and relate $\xi_{i\alpha}$ to $c_{i\alpha}$ (the latter is the discrete set of lattice velocities) by demanding that the BGK collision operator $\Omega_i(f) = -\frac{1}{\tau}\left(f_i - f_i^{eq}\right)$ conserves mass, momentum and energy. The result then is $f_i^{eq} = w_i \rho \left[1 + \frac{u_\alpha c_{i\alpha}}{c_s^2} + \frac{\left(u_\alpha c_{i\alpha}\right)^2}{2c_s^4} - \frac{u_\alpha u_\alpha}{2c_s^2} \right]$ (with $c_s^2 = \frac{1}{3}$ and w_i as given in Eq. 2.24) which is what we anticipated in Eq. 2.49.

2.2.6 | Closing remarks

While the LB algorithm is quite simple, its mathematical principles are quite complex.

Significant portions of this chapter were based on a D2Q9 lattice and a single relaxation time (BGK) collision operator. We note that collision operators and lattice models are active research areas.

For a detailed overview, [36] is a valuable resource. Multiple Relaxation Time (MRT) collision operators have proven to be superior when it comes to simulation turbulence [73,74] where they show more stable behaviour as compared to BGK operators. Also in the field of interfacial flows MRT has advantages over BGK [50]. Increasing the number of discrete velocities has advantages in terms of improved isotropy [60]. D3Q27 lattices combined with MRT collision operators are currently are the forefront (e.g. [28]).

In Chapter 3 we will be discussing a number of more practical aspects. First and foremost, what to do near the boundaries of a flow domain: how to impose no-slip, free slip, periodic, etc. boundary conditions. In the second place we need to discuss compressibility. Clearly the method is compressible: information travels with finite speed since a distribution function only travels one lattice distance per time step. A reflection of this is the finite speed of sound $c_s = \sqrt{\frac{1}{3}}$. Still, we want to apply the method to incompressible flow. In the third place it needs to be established how to translate a physical flow system (with e.g. SI units) into an LB simulation in lattice units.

CHAPTER 3

Practical aspects of lattice-Boltzmann simulations

3.1 | Introduction

After having gone through a theoretical analysis of the lattice-Boltzmann method in Chapter 2, we now will discuss a number of practical aspects, so as to show what choices need to be made when setting up an actual LB simulation.

Topics to be discussed are the (computational) consequences of compressibility of the numerical method for performing incompressible flow simulations; working with lattice units rather than with physical (such as SI) units; on-lattice boundary conditions; and aspects of computer coding. With "on-lattice" boundary conditions is meant conditions directly in terms of the distribution functions. In Chapter 4 we will be discussing velocity-based immersed boundary conditions in an LB context that has relevance for – among more – simulations of solid-liquid suspensions as for instance encountered in sediment transport.

3.2 | Compressibility and the Mach number

From the analyses presented in Ch 2 it is clear that the speed of sound of a lattice-Boltzmann fluid is finite, which implies that we are dealing with a compressible fluid. The Chapman-Enskog analysis showed that with the LB method, we are solving the Navier-Stokes equation for a compressible fluid with a bulk viscosity that is two-thirds of the (shear) viscosity (Eq. 2.47). This – in general – is not the case for a real fluid. This issue, however, is of not much concern if we are planning to work on (near) incompressible flow problems since the bulk viscosity is of no relevance if $\frac{\partial u_\alpha}{\partial x_\alpha} \approx 0$.

Incompressible flow implies that the Mach number is zero: $\mathrm{Ma} = 0$; near-incompressible flow means $\mathrm{Ma} \ll 1$. The Mach number is defined as the ratio of bulk velocity magnitude and speed of sound:

$$\mathrm{Ma} \equiv \frac{|\mathbf{u}|}{c_s} \tag{3.1}$$

Since in a typical LB simulation $c_s = \sqrt{\frac{1}{3}}$ (in lattice units) it means that we need to make sure that $|\mathbf{u}| \ll 1$ in lattice units. As we will see in the next section – which is about how to deal with lattice

units when solving real flow systems stated in physical units – the constraint on the bulk speed $|\mathbf{u}| \ll 1$ effectively is a constraint on the physical time step Δt.

3.3 | From physical (SI) units to lattice units

For translating a real, physical flow problem into a LB simulation we take the example of incompressible lid-driven cavity flow. The two-dimensional geometry is shown in Figure 3.1: we have a square space filled with liquid water (density $\rho = 10^3$ kg/m^3, dynamic viscosity $\mu = 10^{-3}$ Pa·s). Of the four solid walls, the top wall is moving in the positive x-direction with speed U. On all walls we impose a no-slip boundary condition. As a result of the motion of the top wall, the water close to the top wall will be dragged along with the top wall in the positive x-direction until it hits the right wall where it will be deflected in the downward direction. This creates a clockwise circulating flow in the cavity. The cavity has a side length L=0.02 m (2 cm), and U=0.01 m/s (1 cm/s). We want to simulate the flow in the cavity starting up from zero velocity and evolving to a steady state, i.e. we want to solve the bulk velocity in the cavity as a function of x, y, and t: $\mathbf{u}(x, y, t)$ with the lattice-Boltzmann method.

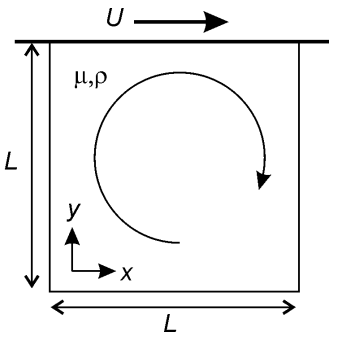


Figure 3.1 | Lid-driven cavity, geometry and coordinate system.

This flow problem is stated in (physical) SI units. We need to translate it into lattice units and need to make decisions regarding time step and lattice cell size. For this, we translate the flow system in dimensionless form by scaling with the input variables U and L where we indicate dimensionless variables with ~:

$$\tilde{x} = \frac{x}{L}, \; \tilde{y} = \frac{y}{L}, \; \tilde{t} = \frac{tU}{L}, \; \tilde{\mathbf{u}} = \frac{\mathbf{u}}{U} \tag{3.2}$$

Dimensional analysis of the flow problem itself shows that the Reynolds number is the all-determining parameter:

$$Re = \frac{\rho U L}{\mu} = \frac{U L}{\nu} \tag{3.3}$$

Based on the paraters given, $Re = 200$. Reynolds similarity now implies that if we are able to perform an LB simulation at Re=200 with lid velocity in lattice units U_{LB} and cavity size in lattice units L_{LB}, and with the outcome of that simulation denoted as $\mathbf{u}_{LB}(x, y, t)$ (in lattice units) then the dimensionless LB result $\widetilde{\mathbf{u}}_{LB} \equiv \frac{\mathbf{u}_{LB}}{U_{LB}}$ is the LB prediction for the actual dimensionless velocity field $\widetilde{\mathbf{u}} = \frac{\mathbf{u}}{U}$ (as defined in Eq. 3.2) so that the LB prediction for the SI velocity field is $\widetilde{\mathbf{u}}_{LB} \times U = \mathbf{u}_{LB} \times \frac{U}{U_{LB}}$ with U the lid velocity in SI units.

From the above we thus are tasked with performing a lid-driven cavity LB simulation at Re=200, and we need to choose our simulation parameters. A first choice is for U_{LB}. Since we want to simulate an incompressible flow, the Mach number Ma needs to be sufficiently small. Since we expect that the maximum fluid speed in the cavity is very close (if not equal) to the speed of the lid, the Mach number will not exceed $\frac{U_{LB}}{c_s}$. We choose U_{LB}=0.1. Then $\frac{U_{LB}}{c_s} \approx 0.17$ which is sufficiently small for near-incompressible conditions. The second choice is for L_{LB}. The spatial resolution of the simulation is governed by L_{LB} since L_{LB} is the number of lattice nodes in x and y direction. We need some fluid mechanics intuition about the flow in a cavity at Re=200. This is expected to be a laminar flow with not much fine-scale flow patterns; we expect a single recirculation loop that fills the largest part of the cavity. We anticipate that this recirculation is going to be well-resolved on a 20×20 mesh, so we choose L_{LB}=20. In order to achieve Re=200, the kinematic viscosity in lattice units thus needs to be $\nu_{LB} = \frac{U_{LB} L_{LB}}{Re} 0.01$. This can be realized by a collision operator with a relaxation time of (see Eq. 2.27) $\tau = 3\nu + \frac{1}{2}$ =0.53.

In summary: U_{LB}=0.1 is motivated by limiting compressibility, L_{LB}=20 is motivated by spatial resolution, ν_{LB} =0.01 for achieving the aimed for Reynolds number.

If we run the simulation in lattice units, the time step is – by definition – $\Delta t = 1$. The velocity of the lid and the size of the cavity, however, determine a more physically meaningful interpretation of the time step. With U_{LB}=0.1, the lid moves over 0.1 lattice unit per time step, and it thus takes $\frac{L_{LB}}{U_{LB}}$ =200 time steps for the lid to move once over the cavity. In the physical system it takes 2 s for the lid to move over the cavity so that the time step in the LB simulation is equivalent to 0.01 s in the physical system.

If we would have concerns about compressibility in the simulation, we could reduce the speed of the lid U_{LB} so as to reduce the Mach number. This would require reducing the viscosity to keep Re the same and it would make the physical time step smaller.

The Courant number of this simulation is $C = U_{LB}\frac{\Delta t}{\Delta x}$ =0.1. This is much smaller than the typically used values of $C \approx 0.8$in explicit time stepping finite difference or finite volume solvers [23]. The small Courant number and thus effectively the small time step are the result of the low Mach number requirement associated to the LB method as applied to incompressible flow.

The scaling analysis for lid-driven cavity flow is relatively simple since we were given a velocity scale in the form of the lid speed U. If we were to be asked to simulate the steady flow between two flat, vertical parallel plates as a result of gravity (see Figure 3.2) our input parameters are the distance between the plates $2D$, the properties of the fluid (kinematic viscosity ν and the density ρ), and gravitational acceleration g. Dimensional analysis shows that this flow is governed by the dimensionless group $\frac{D^3 g}{\nu^2}$ (density does not show up since the two balancing effects of gravity and viscous forces are both proportional to the density: ρg and $\rho\nu$) so that in an LB simulation, we must match this dimensionless number (which is sometimes called the Galileo number Ga). Some notion of the hydrodynamics of this system will make us decide on what to choose for D in lattice units (if we expect laminar flow D=10 should provide adequate spatial resolution). We then still have two free parameters ν and g in lattice units to decide upon. We need to do some a priori analysis to estimate the expected flow velocities (which in this basic flow example is very simple) to come up with a ν and g combination (in lattice units) such that we match the dimensionless group and have flow speeds that are sufficiently low in order to meet the compressibility constraints.

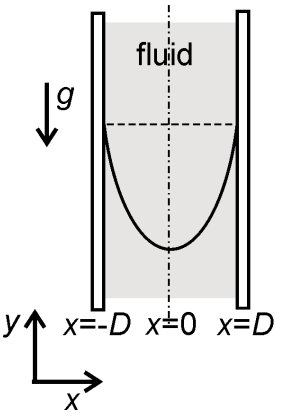


Figure 3.2 | Planar Poiseuille flow driven by gravity.

3.4 | On-lattice boundary conditions

The lid-driven cavity example clearly shows the role of – and the need for – defining boundary conditions. Boundary conditions in LB simulations are a topic of active research and come in a large variety of forms and implementations. Here we discuss a few of the more basic and most used ones, where we limit ourselves to "on-lattice" boundary conditions that deal with boundaries aligned with the lattice. Immersed boundary (IB) methods are a way of dealing with "off-lattice" boundary conditions. IB methods will be discussed in Chapter 4.

First we look at a convenient framework to implement on-lattice boundary conditions. In the streaming step (Eq. 2.29) $f_i\left(\mathbf{x}+\mathbf{c_i}, t+1\right) = f_i^*\left(\mathbf{x}, t\right)$, post-collision distribution functions f_i^* are transferred to neighbouring lattice sites. Lattice sites next to boundaries need to receive f_i^*'s from all directions, also from directions that (seemingly) come from outside the flow domain. This is illustrated in Figure 3.3 (left panel) that focuses on the lower-left corner of the flow domain, e.g. the lower-left corner of the lid-driven cavity. One way to implement boundary conditions is to build a layer of "ghost" cells at the other side of the boundary (the dashed cells in the right panel of Figure 3.3) and populate these ghost cells with f_i^*'s that represent the boundary condition. An LB time step would then consist of (1) collide; (2) populate ghost cells; (3) stream towards all cells *inside* the flow domain.

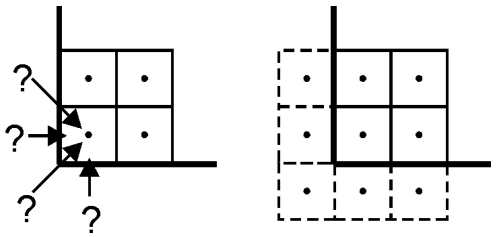


Figure 3.3 | Implementing boundary conditions by populating ghost cells. Bold lines are no-slip walls.

3.4.1 | No-slip boundary condition at fixed wall

Fluid sticks at solid walls. Therefore, if the wall is not moving, the velocity at the wall is zero. A static no-slip wall can be achieved by applying a bounce-back boundary condition: the f_i^*'s that want to leave the flow domain are bounced back when they hit the solid wall. We show this in Figure 3.4 (left panel): the black arrows bounce on the wall and are reverted and become the red arrows. The right panel of Figure 3.4 shows how this is implemented by means of ghost cells. Figure 3.4 illustrates a

so-called half-way bounce-back scenario, where the actual boundary location is half a cell size away from the lattice nodes (i.e. the cell centres) that are populated with f_i^*'s. There are also bounce-back implementations – not discussed here – where nodes are located on the no-slip boundary.

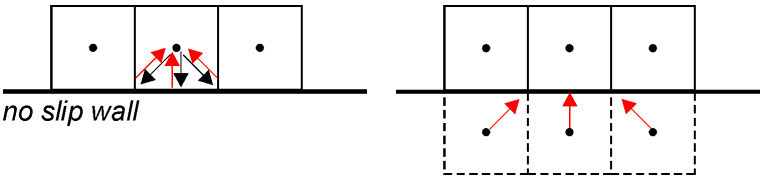


Figure 3.4 | Bounce-back on a no-slip wall (left) and implementation in terms of ghost cells (right).

3.4.2 | No-slip boundary condition at moving wall

A wall moving parallel to itself (as the top wall in the lid-driven cavity) requires an extension of the bounce-back condition. Consider the situation in Figure 3.5. The motion of the wall in the positive x-direction will add momentum to the "particle" (f_i^*) that is bounced off the wall in the positive x-direction, and it will reduce the momentum of the particle that is reflected in the negative x-direction. This is illustrated by the longer red vector that points to up-right, and the shorter red vector that points up-left. This process can be described as

$$f_5(\mathbf{x}, t+1) = f_7^*(\mathbf{x}, t) + 2\frac{\rho(\mathbf{x}, t)\, u_0 w_7}{c_s^2} \quad f_6(\mathbf{x}, t+1) = f_8^*(\mathbf{x}, t) - \frac{2\rho(\mathbf{x}, t)\, u_0 w_8}{c_s^2} \qquad 3.4$$

where we used the numbering of velocities as earlier given in Figure 2.1, repeated for convenience in Figure 3.6.

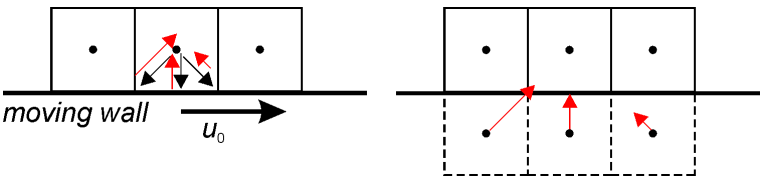


Figure 3.5 | Bounce-back on parallel moving wall.

We note that a change in momentum is achieved by a change in mass. As we can see in Eq. 3.4, the net change in mass is equal to zero since $w_7 = w_8 \left(= \frac{1}{36}\right)$.

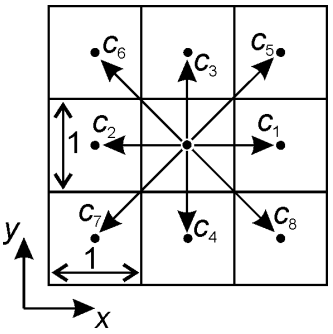


Figure 3.6 | Numbering of velocities in the D2Q9 velocity set.

3.4.3 | Periodic boundary conditions

Periodic conditions imply that what leaves the domain on one side, re-enters on the other side. This is illustrated in Figure 3.7. The masses represented by the red arrows on the left of the domain (in the left panel of Figure 3.7) come from the right side of the domain. The right panel of Figure 3.7 shows how this can be implemented using ghost cells: we copy the far right layer of cells *inside* the flow domain to the ghost cells on the left *outside* the flow domain.

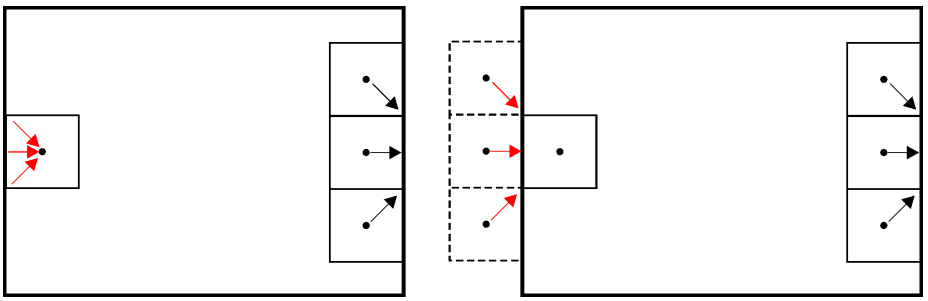


Figure 3.7 | Left-right periodic boundary condition (left panel) and implementation by ghost cells (right panel).

3.4.4 | Free-slip non-penetrating boundary

A free-slip (i.e. zero shear stress) wall can be achieved by specular reflection (i.e. a mirror-like reflection) of f_i^*'s on the boundary, see Figure 3.8. The right panel of this figure is the same as the right panel of Figure 3.4 (no-slip). Note, however, that the red arrows come from a different location: they are the reflected black arrows of the left panel.

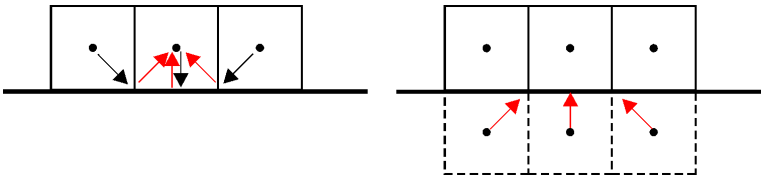


Figure 3.8 | Free slip wall.

3.5 | Elements of computer coding

The ingredients of an LB computer code are:

(1) Definitions: set grid size $nx \times ny$, number of time steps, kinematic viscosity ν, weighing factors w_i.
(2) Initialize $f_i^*(\mathbf{x}, t = 0)$, e.g. by setting it equal to the equilibrium distribution function with uniform ρ and $\mathbf{u} = \mathbf{0}$.
Start the time stepping loop; what comes now is done each time step:
(3) On-lattice boundary conditions: Fill the "ghost cells" with boundary values of f_i^* (see Section 3.4)
(4) Stream, i.e. execute $f_i(\mathbf{x} + \mathbf{c_i}, t + 1) = f_i^*(\mathbf{x}, t)$ (Eq. 2.29)
(5) Collide ($f_i^*(\mathbf{x}, t) = f_i(\mathbf{x}, t) + \Omega_i(\mathbf{x}, t)$) and return to (3) until the pre-set number of time steps has been completed, or until the convergence/steady state condition has been achieved
(6) Leave the time stepping loop and write to file, plot,… etc.

Below are coding suggestions (in pseudo fortran) regarding some of the main ingredients of the LB algorithm.

3.5.1 | Streaming

In streaming we execute Eq. 2.29 (repeated here as Eq. 3.5):

$$f_i(\mathbf{x} + \mathbf{c_i}, t + 1) = f_i^*(\mathbf{x}, t) \tag{3.5}$$

After we have dealt with our boundary conditions by populating the ghost cells, this can be implemented in computer code as a loop over all the active (i.e. non-ghost) lattice nodes. One could

be tempted to work with two arrays f(0…8,1…nx,1…ny)for the left-hand side of Eq. 3.5 and fstar(0…8,0…nx+1,0…ny+1)for the right-hand side and with nx the number of active lattice nodes in *x*-direction and ny the number of active nodes in *y*-direction. Note that fstar needs to include ghost cells, hence the extension to 0…nx+1,0…ny+1 where f has 1…nx,1…ny. Then do something like this piece of *pseudo* code (where I refer to the velocity numbering in Figure 3.6).

```
for j=1:ny
  for i=1:nx
    f(0,i,j)=fstar(0,i,j)
    f(1,i,j)=fstar(1,i-1,j)
    f(2,i,j)=fstar(2,i+1,j)
    f(3,i,j)=fstar(3,i,j-1)
    f(4,i,j)=fstar(4,i,j+1)
    f(5,i,j)=fstar(5,i-1,j-1)
    f(6,i,j)=fstar(6,i+1,j-1)
    f(7,i,j)=fstar(7,i+1,j+1)
    f(8,i,j)=fstar(8,i-1,j+1)
  end
end
```

This is not a (memory) efficient way to do streaming as it requires two arrays f and fstar (specifically in three dimensions and for large domains, these arrays can get very large) that eat up a lot of computer memory. We can do, however, with one array f(0…8,0…nx+1,0…ny+1). At the start of the streaming operation this array contains f_i^*, including its values in the ghost nodes. Then we run the following

```
for j=1:ny
  for i=1:nx
    f(2,i,j)=f(2,i+1,j)
    f(4,i,j)=f(4,i,j+1)
    f(7,i,j)=f(7,i+1,j+1)
    f(8,i,j)=f(8,i-1,j+1)
  end
end

for j=ny:-1:1
```

```
  for i=nx:-1:1
    f(1,i,j)=f(1,i-1,j)
    f(3,i,j)=f(3,i,j-1)
    f(5,i,j)=f(5,i-1,j-1)
    f(6,i,j)=f(6,i+1,j-1)
  end
end
```

Note that in the second double loop, we go in reverse order (the -1 in the for statements). The trick here is to update from f_i^* to f_i in the same direction as you go through the loops so that on the right side of the equal sign, there actually is an f_i^* and not an already updated f_i. In one dimension this is relatively simple. In two dimensions (as is the case here), one should realize that the data is ordered as

$$(i,j): \quad [(1,1)(2,1)\cdots(nx,1)]\,[(1,2)(2,2)\cdots(nx,2)]\cdots\cdots[(1,ny)(2,ny)\cdots(nx,ny)]$$

3.5.2 | Filling in the ghost nodes

As an example of filling the ghost nodes with the relevant f_i^* values, here is how it can be done for the bottom wall of the cavity, which is a no-slip wall. The first row of active nodes above the bottom has j=1. Therefore, the row of ghost nodes underneath the bottom wall has j=0. The velocity directions entering through the bottom are 3, 5, 6 (see Figure 3.6). The opposite velocity directions are 4, 7, 8 respectively. The associated f_i^*'s come from one layer above, hence the j+1 on the right-hand side. As can be seen in Fig. 3.4, the f_i^*'s of the diagonal velocity directions need to be shifted in the x-direction, hence the i+1 and i-1 on the right-hand side for directions 5 and 6.

```
% bounce back at bottom
j=0;
for i=1:nx
  f(3,i,j)=f(4,i,j+1);
  f(5,i,j)=f(7,i+1,j+1);
  f(6,i,j)=f(8,i-1,j+1);
end
```

3.5.3 | Calculating the equilibrium distribution & performing collisions

Here is a piece of code dealing with how to do the collision step (Eq. 2.28, Chapter 2). We loop over all active nodes.

We do a few things to try and speed up the computations (where it should be noted that everything that is calculated inside the loop is calculated for each node and each time step so that computational efficiency inside the loop pays off):

- Write out ("unroll") the expression for `rho`.
- Multiplication is cheaper than division; we do one division to determine `1.0/rho` and then use `rrho` twice.
- Determine `usq=ux*ux+uy*uy` and use it multiple times.
- Similar for `cdotu`.

```
for j=1:ny
  for i=1:nx
    rho=f(0,i,j)+f(1,i,j)+f(2,i,j)+f(3,i,j)+f(4,i,j)+f(5,i,j)+
        f(6,i,j)+f(7,i,j)+f(8,i,j);
    rrho=1.0/rho;
    ux=f(1,i,j)-f(3,i,j)+f(5,i,j)-f(6,i,j)-f(7,i,j)+f(8,i,j);
    ux=ux*rrho;
    uy=f(2,i,j)-f(4,i,j)+f(5,i,j)+f(6,i,j)-f(7,i,j)-f(8,i,j);
    uy=uy*rrho;
    usq=ux*ux+uy*uy;
    feq(0)=m0*rho*(1.0-1.5*usq);
    cdotu=ux;
    feq(1)=m1*rho*(1.0+3.0*cdotu+4.5*cdotu*cdotu-1.5*usq);
    cdotu=uy;
    feq(2)=m1*rho*(1.0+3.0*cdotu+4.5*cdotu*cdotu-1.5*usq);
    cdotu=-ux;
    feq(3)=m1*rho*(1.0+3.0*cdotu+4.5*cdotu*cdotu-1.5*usq);
    cdotu=-uy;
    feq(4)=m1*rho*(1.0+3.0*cdotu+4.5*cdotu*cdotu-1.5*usq);
    cdotu=ux+uy;
    feq(5)=m2*rho*(1.0+3.0*cdotu+4.5*cdotu*cdotu-1.5*usq);
    cdotu=-ux+uy;
```

```
    feq(6)=m2*rho*(1.0+3.0*cdotu+4.5*cdotu*cdotu-1.5*usq);
    cdotu=-ux-uy;
    feq(7)=m2*rho*(1.0+3.0*cdotu+4.5*cdotu*cdotu-1.5*usq);
    cdotu=ux-uy;
    feq(8)=m2*rho*(1.0+3.0*cdotu+4.5*cdotu*cdotu-1.5*usq);
    for l=0:8
      f(l,i,j)=(1.0-rtau)*f(l,i,j)+rtau*feq(l);
    end
  end
end
```

CHAPTER 4

Solid-liquid systems – sediment transport

4.1 | Introduction

Liquid streams laden with solid particles (sediment) is a key phenomenon in hydraulics. Simulations of such systems, including those based on the lattice-Boltzmann method, are relevant for a better understanding of the way solids and liquid interact. The complexity and (often large) scale of sediment transporting systems primarily asks for experimental research with simulations in a supporting role, e.g. in identifying rate-limiting physical phenomena.

Simulations of solid-liquid systems can be categorized as Eulerian-Lagrangian and Eulerian-Eulerian. In the latter category the solids phase is treated as a continuum, as is the liquid phase. In the former category the solids are treated as individual particles or as clusters of particles ("parcels") that are tracked through the continuous liquid phase. In this chapter we will only consider Eulerian-Lagrangian simulations. For more details of Eulerian-Eulerian simulations applied to fluid-solid systems we refer to [10].

Eulerian-Lagrangian approaches can be further divided into two sub-categories: particle-resolved simulations and particle-unresolved simulations (the latter also termed point particle simulations or discrete particle model (DPM) simulations). In particle-resolved simulation (PRS), the flow around individual particles is resolved, i.e. the simulation explicitly imposes no-slip at the solid-liquid interfaces. This requires a high level of spatial resolution; the computational grid on which the fluid flow is solved needs to be finer by one order of magnitude than the size of a particle in each coordinate direction. A common way of imposing no-slip at particle surfaces is by an immersed boundary method.

In particle-unresolved simulations we do not resolve the flow around individual particles; the grid on which the fluid flow is solved is coarser than the particle size. This then requires strong assumptions to recover the dynamics of the particles. In particle-resolved simulations, the hydrodynamic forces and torques on the particles are part of the flow solution, so these are used for updating the equations of motion of the particles. In particle-unresolved simulations we need models for hydrodynamic forces (and torques) and the way they depend on (local) conditions. On the positive side, the much less stringent resolution requirements of particle-unresolved simulations make it

possible to consider many more particles (by three to four orders of magnitude) in a simulation as compared to a particle-resolved simulation.

In this chapter both resolved and unresolved particle approaches will be treated. We start with the former by first discussing the immersed boundary method for imposing no-slip at the solid particle surfaces. Particle-resolved simulations directly account for particle shape and results will be presented for spherical as well as for non-spherical particles.

One application of particle-resolved simulations is to formulate and refine hydrodynamic force models and – more general – models for the ways solids and liquid interact to be used in particle-unresolved simulations. Procedures for performing particle-unresolved simulations will be discussed, and results will be presented only for particles having a spherical shape.

4.2 | Particle-resolved lattice-Boltzmann simulations

The theoretical and practical groundwork for particle-resolved suspension simulations through a lattice-Boltzmann (LB) approach were laid in two papers by A.J.C. Ladd [38,39]. The papers are not only among the first in terms of highly resolved suspension simulations, they also give a comprehensive account of the LB method and the way it relates to Newtonian fluid dynamics and how moving no-slip boundaries can be accounted for.

Figure 4.1 shows the way a circular object is represented [39]. The numerical location of the no-slip surface is halfway the link between two lattice nodes that is cut by the surface. As can be seen in the figure, a curved surface results in a stair-step boundary condition that approaches the actual surface upon refining the lattice. There is fluid inside the particle surfaces which is kept there for convenience. This internal fluid is decoupled from the external fluid. The convenience lies in the fact that this approach avoids the need for creating fluid when a moving particle uncovers a node and destroying fluid when it covers a node. Reference [39] deals with a number of practicalities when applying the approach to multiple spherical particles suspended in fluid.

4.2.1 | Immersed boundary method

The stair-step boundaries and associated particle shape fluctuations when a particle moves over the lattice have motivated the use of immersed boundaries in LB simulations. Initial attempts to apply immersed boundaries in the context of lattice-Boltzmann simulations were based on the feedback approach due to Goldstein et al [27]. The idea is to locally apply forces on the fluid that oppose a difference between the fluid velocity at the immersed boundary and the desired velocity of the

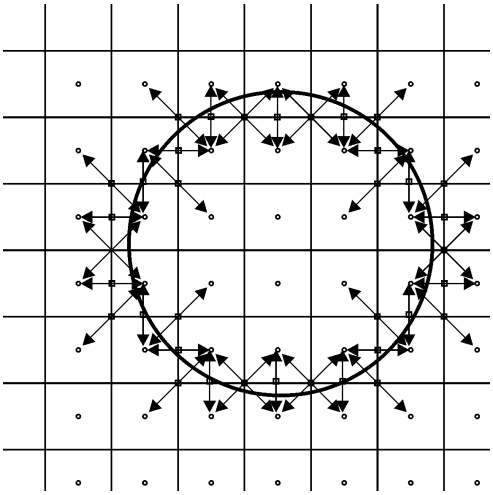


Figure 4.1 | Representation of a circular no-slip boundary on a square lattice. Circular dots are lattice nodes sitting in the centre of square cells. The bounce-back conditions representing no-slip are imposed halfway the links between those two nodes that intersect with the circle (square dots).

immersed boundary. Iteration or time stepping then establishes and maintains the velocity condition. An early – 1999 – application of this method is the representation of the impeller in a mixing tank in single-phase large-eddy simulations [19]. The surface of the impeller is represented by a closely spaced set of off-lattice marker points with a typical nearest neighbour spacing of $0.5-0.7\Delta$. The fluid velocity at the marker points is determined through interpolation (linear or quadratic) from the lattice and is symbolically expressed as

$$\mathbf{u}_j^{(n)} = \sum_k G_k\left(\mathbf{r}_j^{(n)}\right)\mathbf{u}_k^{(n)} \tag{4.1}$$

with j indicating marker points, k lattice points, (n) a time instant, and $G_k\left(\mathbf{r}_j\right)$ interpolation coefficients that depend on the relative position $\mathbf{r}_j$ of lattice and marker points. The sum is over the lattice points surrounding marker point j. We then update the force on the fluid responsible for imposing the velocity boundary condition at marker point j as

$$\mathbf{F}_j^{(n)} = \mathbf{F}_j^{(n-1)} + \alpha\left(\mathbf{w}_j^{(n)} - \mathbf{u}_j^{(n)}\right) \tag{4.2}$$

with $\mathbf{w}_j^{(n)}$ the velocity boundary condition and α a relaxation parameter. The final step is to distribute the forces at the marker points to their surrounding lattice nodes for which we use the interpolation coefficients as introduced in Eq. 4.1:

$$\mathbf{F}_k^{(n)} = \sum_j G_k \left(\mathbf{r}_j^{(n)} \right) \mathbf{F}_j^{(n)} \tag{4.3}$$

with the sum over all marker points *j* that are within the interpolation range of lattice point *k*.

In the section Algorithms – A.2 we have listed sample computer code (fortran77) to execute the velocity interpolation (Eq. 4.1) and immersed boundary force calculations (Eqs. 4.2 and 4.3).

The stability and accuracy of the method hinges on the free parameter α [19]. Having a free parameter is a disadvantage. On the positive side, the method largely retains the locality of the arithmetic operations inherent to the LB method. This feature of the LB method enables simple computational parallelization strategies.

As noted, the initial application of the above method was used for imposing the moving no-slip condition at the surface of a revolving impeller. Extending it to solid particles moving through fluid is straightforward [63]. Different from an impeller, however, the linear and angular motion of particles and therefore their solid surface velocity is not known a priori and needs to be solved. This involves solving the equations of linear and rotational motion of each particle. This is the subject of Section 4.2.3. The equations of motion include the hydrodynamic force and torque on each particle. These are directly available through integration of the force distribution that is responsible for imposing the immersed boundary (Eq. 4.3) over the surface of the particle.

Alternative force-based immersed boundary methods in an LB context have been proposed by e.g. [21]. They stay close to the original ideas of Peskin [49] of an “elastic restoration force” that counteracts deformations of the immersed solid object as a result of fluid flow. The restoring forces act on off-grid marker points. They are distributed over neighbouring lattice nodes through a regularized delta function [21].

The force-based immersed boundary methods discussed here have fluid inside a particle. As a consequence, the immersed boundary forces accelerate external as well as internal fluid. It is the external force that has direct physical significance as the hydrodynamic force acting on the particle. As a result, one needs to distinguish between the external and internal force contributions. One way to do this is to assume that the fluid inside the particle moves as a solid body. This specifically is a good approximation when, in addition to immersed boundary marker points at the fluid-solid interface,

a layer of marker points is added inside the particle [18]. Based on this approximation, the external force $\mathbf{F_{ext}}$equals the overall immersed boundary force $\mathbf{F_{IB}}$ minus the inertial force of the internal fluid [18]:

$$\mathbf{F_{ext}} = \mathbf{F_{IB}} - \rho V_p \frac{d\mathbf{u_p}}{dt} \tag{4.4}$$

with ρ the fluid density and V_p the volume of the particle. For the external torque an equivalent expression can be derived in terms of the overall immersed boundary torque and the rotational acceleration of the internal fluid.

Given that $\mathbf{F_{IB}}$ is a directly available output of the immersed boundary method, its inclusion in the equation of linear motion of the particles leads to inertia terms of the form $(\rho_p - \rho) V_p \frac{d\mathbf{u_p}}{dt}$. For cases with particles and fluid having similar density, the density difference and therefore the inertia term gets small which has problematic consequences for the stability of numerical integration of the equation of motion. Specific time stepping algorithms have been devised to mitigate these issues [22].

Given that the immersed boundary methods discussed above require including forces acting on the fluid we need to extend the description of the LB method as compared to Section 2.2 that did not deal with external forces.

There are two main approaches for incorporating body forces in LB methods. One is via adapting the collision operator, another is via including force as a source term in the lattice-Boltzmann equation (Eq 2.23). We will follow the latter approach. We define $f_i^* = f_i + \Omega_i$ with f_i the pre-collision distribution function, and f_i^* the post-collision distribution function. This expression we now supplement with a source term S_i

$$f_i^* = f_i + \Omega_i + S_i \tag{4.5}$$

The source term S_i relates to the body force component F_α in the following manner [36]

$$S_i = \left(1 - \frac{1}{2\tau}\right) w_i \left(\frac{c_{i\alpha}}{c_s^2} + \frac{\left(c_{i\alpha}c_{i\beta} - c_s^2\delta_{\alpha\beta}\right) u_\beta}{c_s^4}\right) F_\alpha \tag{4.6}$$

Equation 4.6 is the result of yet another Chapman-Enskog analysis involving the Navier-Stokes equation that includes a force term. Also as a result of the Chapman-Enskog analysis the relation between momentum and distribution functions (Eq. 2.20) requires a force correction:

$$\rho \mathbf{u} = \sum_i f_i \mathbf{c_i} + \frac{1}{2}\mathbf{F} \tag{4.7}$$

4.2.2 | Close-range interactions in dense suspensions

From an engineering perspective, applications with high solids loading and/or very inhomogeneous solids distribution are particularly interesting. An example of the latter is encountered in sediment transport. An archetypical sediment transport situation shows a dense granular bed of solid particles over which there is a liquid flow that partly mobilizes the particles giving rise to an inhomogeneous distribution of solids. High solids loading inside the bed, directly above the bed a region with appreciable amounts of particles and higher above the bed a virtually clear liquid. In principle, the tools explained above, more specifically the immersed boundary method, are a means to simulate systems like this. They need two extensions that will be discussed in this and in the next sub-section. In the first place we need to consider how to represent close-range interactions between particles. In the second place we need methods to numerically deal with the dynamics of the solid particles. The latter is a relatively trivial exercise for the special case of spherical particles, a little less so for non-spherical particles.

As noted above, the immersed boundary method represents solid particle surfaces by closely spaced marker points where we match the fluid velocity to the local solid surface velocity. The solid surfaces move over a fixed cubic lattice of points on which the fluid flow is solved. In regions with high solids loading, the distance between surfaces of different particles are often small or – for two touching particles – even zero. In such cases the flow dynamics in the space between the particles gets under-resolved. At the same time there are strong lubrication and – upon touching – "dry" interaction forces between particles. Such close-range interaction forces need to be taken explicitly into account in a simulation.

The marker points for executing the immersed boundary method are also used to detect close proximity between particle surfaces. Figure 4.2 shows two marker points (1 and 2) on the surface of two different particles (A and B). When the distance δ as defined in Figure 4.2 falls below the preset value δ_d and at the same time $\delta_\lambda < \lambda$ (δ_λ defined in Figure 4.2, λ a preset distance) the radial lubrication force becomes active according to

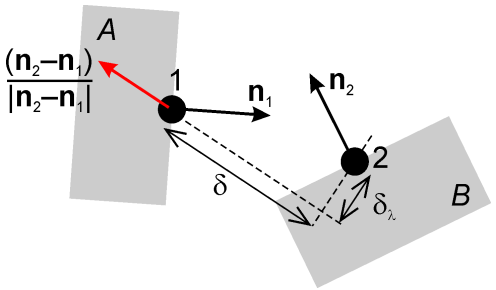


Figure 4.2 | Two marker points 1 and 2 on two different particles (A and B respectively) in close proximity. Definition of normal vectors and radial distance δ and lateral distance δ_λ.

$$\mathbf{F}_{12}^{n,lub} = k^n \left(\frac{1}{\delta} - \frac{1}{\delta_d}\right) \frac{|\delta_\lambda - \lambda|}{\lambda} \mathbf{\Delta u}^n \text{ if } \delta < \delta_d \text{ and } |\delta_\lambda| < \lambda; \quad \mathbf{F}_{12}^{n,lub} = \mathbf{0} \text{ otherwise} \tag{4.8}$$

with $\mathbf{\Delta u}^n$ the relative velocity of marker points 1 and 2 along the average normal $\mathbf{n} = \frac{\mathbf{n_2}-\mathbf{n_1}}{|\mathbf{n_2}-\mathbf{n_1}|}$. Since this force is to account for hydrodynamic interactions unresolved by the lattice, the distances δ_d and λ have been set to values of the same order as the lattice spacing: $\delta_d = \lambda = \Delta$. For the radial lubrication interaction between two spheres with diameter d the expression for the coefficient k^n reads $k^n = \frac{3}{8}\pi\rho\nu d^2$ [35]. For interactions involving non-spherical particles we need to make ad-hoc choices for k^n. As an example, for cylinders we have been substituting the cylinder diameter in the latter expression [16].

When $\delta < \delta_0$ with $\delta_0 = 0.1\Delta$ (and if still $\delta_\lambda < \lambda$) we assume that a dry contact is present that we represent by a radial repulsive linear spring force

$$\mathbf{F}_{12}^{n,c} = k\left(\delta_0 - \delta\right) \frac{|\delta_\lambda - \lambda|}{\lambda} \mathbf{n} \text{ if } \delta < \delta_0 \text{ and } |\delta_\lambda| < \lambda; \quad \mathbf{F}_{12}^{n,c} = \mathbf{0} \text{ otherwise} \tag{4.9}$$

with k the spring constant.

The onset of the radial contact force also triggers the dry friction force by activating a spring in tangential direction. Each time step after its activation, until the contact gets deactivated, the tangential spring is stretched according to $\mathbf{s} = \int (\mathbf{u}_2 - \mathbf{u}_1 - \mathbf{\Delta u}^n)\, dt$ with, in the simulations, the integral being a sum over the finite size time steps Δt. The associated friction force is

$$\mathbf{F}_{12}^{t,c} = k\mathbf{s} \quad \text{if } k\,|\mathbf{s}| \le \mu_{AB}\,|\mathbf{F}_{12}^{n,c}|\,; \quad \mathbf{F}_{12}^{t,c} = \mu_{AB}\,|\mathbf{F}_{12}^{n,c}|\,\frac{\mathbf{s}}{|\mathbf{s}|} \quad \text{if } k\,|\mathbf{s}| > \mu_{AB}\,|\mathbf{F}_{12}^{n,c}| \tag{4.10}$$

where we use the same spring constant k as for the radial contact force and introduce a friction coefficient μ_{AB} between particle A and B. In most practical cases we will only consider one overall friction coefficient μ. The contact and lubrication forces are summed up per particle to get the total force and torque due to close-range interaction acting on each particle.

Identifying close-distance marker points makes use of a link-list algorithm that is listed in Algorithm section A.1.

4.2.3 | Rigid particle dynamics

The dynamical equations for particle linear and angular velocity then are written as

$$V_p\left(\rho_p-\rho\right)\frac{d}{dt}\mathbf{u_p}=-\mathbf{F_{IB}}+\mathbf{F_c}-\left(\rho_p-\rho\right)V_p g\mathbf{e_z} \tag{4.11}$$

$$\left(\mathbf{I}-\mathbf{I_{int}}\right)\frac{d}{dt}\omega_\mathbf{p}+\omega_\mathbf{p}\times\left(\left(\mathbf{I}-\mathbf{I_{int}}\right)\omega_\mathbf{p}\right)=-\mathbf{T_{IB}}+\mathbf{T_c} \tag{4.12}$$

In Eq. 4.12, $\mathbf{I_{int}}$ is the moment of inertia of the internal fluid. For modest to low density ratios the coefficients in front of the $\frac{d}{dt}$ terms of Eqs. 4.11 and 4.12 can get small (as also discussed in the context of Eq. 4.4). This then leads to severe time step limitations if an Euler forward method would be applied to integrate the equations numerically. For this reason, a split-derivative time-stepping procedure [22] has been used for updating Eqs. 4.11 and 12. This allows a time step that is the same as the time step of the LB scheme.

More specifically, the term $V_p\left(\rho_p-\rho\right)\frac{d}{dt}\mathbf{u_p}=V_p\rho_p\left(1-\frac{1}{\gamma}\right)\frac{d}{dt}\mathbf{u_p}$ in Eq. 4.11 has been discretized as $V_p\rho_p\left[\frac{\mathbf{u_p}^{(n+1)}-\mathbf{u_p}^{(n)}}{\Delta t}-\frac{1}{\gamma}\frac{\mathbf{u_p}^{(n)}-\mathbf{u_p}^{(n-1)}}{\Delta t}\right]$ with (n) denoting the time level and $\gamma=\frac{\rho_p}{\rho}$. This then leads to the following update rule for linear velocity

$$\mathbf{u_p}^{(n+1)}=\left(1+\frac{1}{\gamma}\right)\mathbf{u_p}^{(n)}-\frac{1}{\gamma}\mathbf{u_p}^{(n-1)}-\frac{\Delta t\mathbf{F_{IB}}^{(n)}}{V_p\rho_p}+\frac{\Delta t\mathbf{F_c}^{(n)}}{V_p\rho_p}-\Delta t\left(1-\frac{1}{\gamma}\right)g\mathbf{e_z} \tag{4.13}$$

Once the linear velocity is updated, we displace the center location of each particle through an Euler explicit step: $\Delta\mathbf{x_p}=\mathbf{u_p}\Delta t$.

Rotational motion of each particle is solved in a reference frame attached to the particle and along its principal axes so that the moment of inertia tensor is diagonal and constant. An approach analogous to that of linear motion has been followed for numerically integrating rotational motion (Eq. 4.12):

$$\omega_{\mathbf{p}}^{(n+1)} = \left(1+\frac{1}{\gamma}\right)\omega_{\mathbf{p}}^{(n)} - \frac{1}{\gamma}\omega_{\mathbf{p}}^{(n-1)} - \Delta t\mathbf{I}^{-1}\mathbf{T}_{\mathbf{IB}}^{(n)} + \Delta t\mathbf{I}^{-1}\mathbf{T}_{\mathbf{c}}^{(n)} - \Delta t\left(1-\frac{1}{\gamma}\right)\mathbf{I}^{-1}\left[\omega_{\mathbf{p}}^{(n)} \times \left(\mathbf{I}\omega_{\mathbf{p}}^{(n)}\right)\right] \tag{4.14}$$

Keeping track of the orientation of the particles makes use of quaternions [37]. Each particle's orientation is characterized by a unit quaternion $q = (q_0, \mathbf{q})$ with q_0 a scalar value and $\mathbf{q}$ a three-dimensional vector (q_1, q_2, q_3) and $\sqrt{q_0^2+q_1^2+q_2^2+q_3^2} = 1$. An exact solution for the evolution of a quaternion rotating with an angular velocity $\omega_{\mathbf{p}}$ over a time interval Δt starting from $q^{(n)}$ at time level (n) is available [37]:

$$q^{(n+1)} = q^{(n)} \circ \left(\cos\left(\frac{1}{2}\Delta t\right), \omega_{\mathbf{p}} \sin\left(\frac{1}{2}\Delta t\right)\right) \tag{4.15}$$

with the symbol $\circ$ denoting a quaternion multiplication. We use Eq. 4.15 for updating the quaternion of each particle from one time step to the next. In the Algorithm section A.3 we have listed computer code that executes Eq. 4.15.

Quaternions effectively facilitate transferring information between the (x_1, x_2, x_3) coordinate system attached to a (non-spherical) particle and the inertial (x, y, z) system. The rotation of a vector $\mathbf{x}$ in the (x_1, x_2, x_3) system to a vector $\mathbf{y}$ in the (x, y, z) can be expressed as

$$\mathbf{y} = \mathbf{S}\mathbf{x} \tag{4.16}$$

with [37]

$$\mathbf{S} = \begin{bmatrix} 1-2\left(q_2^2+q_3^2\right) & 2\left(q_1q_2-q_0q_3\right) & 2\left(q_1q_3+q_0q_2\right) \\ 2\left(q_2q_1+q_0q_3\right) & 1-2\left(q_1^2+q_3^2\right) & 2\left(q_2q_3-q_0q_1\right) \\ 2\left(q_3q_1-q_0q_2\right) & 2\left(q_3q_2+q_0q_1\right) & 1-2\left(q_1^2+q_2^2\right) \end{bmatrix} \tag{4.17}$$

The coordinates of the marker points for the IBM are stored for one reference particle in the (x_1, x_2, x_3) coordinate system. Equation 4.17 is used for each particle at each time step to transfer its marker points to the (x, y, z) system in order to apply the IBM. One result of the IBM is the

torque $\mathbf{T_{IB}}$ associated to each particle in the (x, y, z) system. Since we solve the equation of rotational motion (Eq. 4.12) in the (x_1, x_2, x_3) system, $\mathbf{T_{IB}}$ needs to be rotated to the latter system. This requires the inverse of $\mathbf{S}$ which is its transpose: $\mathbf{S}^{-1} = \mathbf{S}^T$.

4.2.4 | Examples of particle-resolved sediment transport simulations

Incipient particle motion and critical Shields number under laminar flow

Spherical solid particles all having the same radius a are placed along with a Newtonian liquid with kinematic viscosity ν and density ρ in a rectangular three-dimensional domain, see Figure 4.3. The solids have a density $\rho_p > \rho$. Gravity points in the negative z-direction (see Figure 4.3 for the coordinate system used). Periodic conditions apply in the x and y-direction. There are two no-slip walls at the top and bottom of the domain. A shear flow is created by moving the top wall in the x-direction with a speed u_0 creating an average shear rate in the liquid phase $\dot{\gamma}_0 = \frac{d\langle u_x \rangle}{dz}$.

The flow system as described above can be characterized by a set of dimensionless numbers. Most prominently there is the Shields parameter $\theta = \frac{\rho\nu\dot{\gamma}_0}{g(\rho_p-\rho)d}$ with g gravitational acceleration ($d = 2a$). In addition we define a Reynolds number as $Re = \frac{\dot{\gamma}_0 d^2}{(4\nu)}$, and the density ratio as $\frac{\rho_p}{\rho}$. In this text we focus on the effect of θ on solids and liquid velocity profiles and on granular bed mobility. For this we have fixed the density ratio to $\frac{\rho_p}{\rho} = 4.0$ and the friction coefficient (see Eq. 4.10) to μ =0.1 and consider Re in the range 0.04 to 0.37. Regarding friction, it has been noted [12] that friction and friction induced rolling of particles is important for mobilizing granular beds.

Figure 4.4 show solids and liquid average stream wise velocity profiles for a range of Shields parameters. Solids velocities are systematically lagging fluid velocities and their mobility strongly depends on θ, with practically no solids motion for $\theta \leq 0.15$.

With the data generated over a range of Shields numbers we were able to narrow down an interval for the critical Shields number θ_c that demarcates immobile and mobile beds. This critical Shields number interval is $0.10 < \theta_c < 0.15$ [12] for the laminar flow conditions we are considering here. This interval is independent of the Reynolds number (at least up the highest Re considered here which was $Re = 0.37$). Figure 4.5 illustrates this. We plot there the dimensionless solids flux (φ) as a function of θ, for three different Reynolds numbers as indicated. In between θ =0.10 and 0.15 the solids flux switches from zero to non-zero.

In closing this sub-section on granular bed erosion under laminar flow it is useful to (qualitatively) show the capabilities of particle-resolved simulations (based on the LB method and immersed boundaries) for non-spherical particles. In Figure 4.6 we show a recent (and as of yet unpublished) comparison between erosion of granular beds consisting of spheres, short cylinders, and a mixture

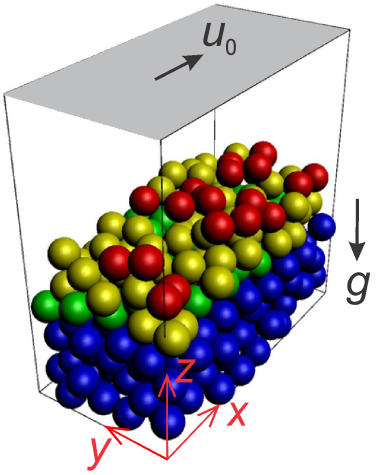


Figure 4.3 | Laminar shear flow geometry including coordinate system. Spherical particles colored according to their speed. Red: $|\mathbf{u_p}| \geq 0.05\dot{\gamma}_0 d$; yellow: $0.05\dot{\gamma}_0 d > |\mathbf{u_p}| \geq 0.005\dot{\gamma}_0 d$; green: $0.005\dot{\gamma}_0 d > |\mathbf{u_p}| \geq 0.0005\dot{\gamma}_0 d$; blue: $0.0005\dot{\gamma}_0 d > |\mathbf{u_p}|$.

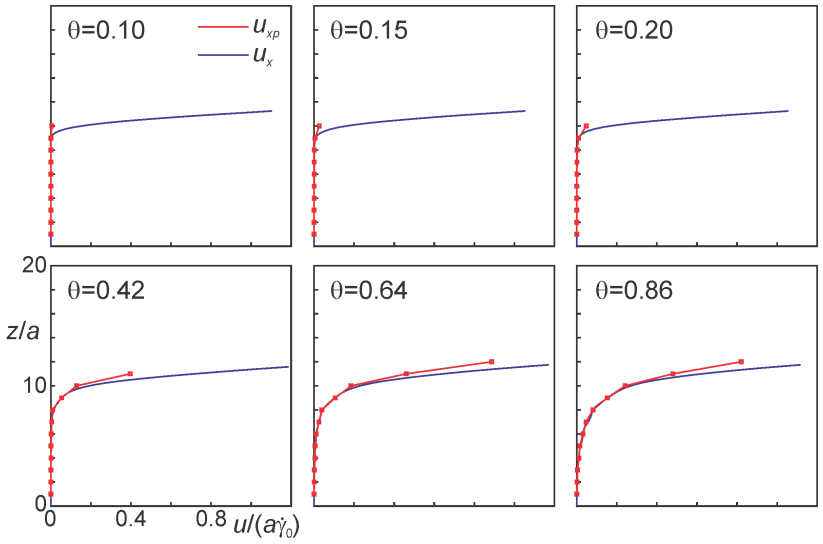


Figure 4.4 | Average solids and liquid stream wise velocity (u_{xp} and u_x respectively) for four variants for different θ and further Re=0.12, $\frac{\rho_P}{\rho} = 4$ and $\mu = 0.1$. The solids velocity is indicated only if locally the solids volume fraction exceeds 2.5%.

of the two. The spheres and cylinders have the same volume and the same density and the Shields

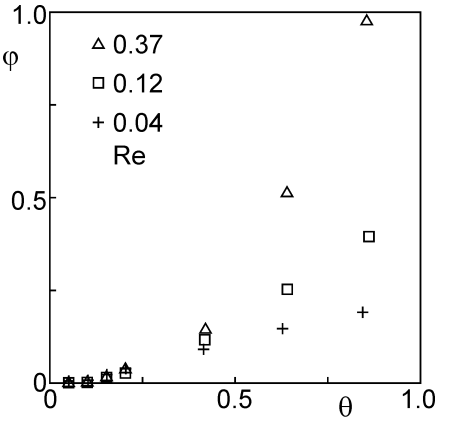


Figure 4.5 | Averaged dimensionless volumetric solids flux φ [12] as a function of the Shields number θ for three values of Re (as indicated) and $\frac{\rho_P}{\rho} = 4.0$.

parameter is the same in all three cases. First inspection of the results only shows a minor impact (less than 10%) of the composition of the granular bed on the solids transport rate.

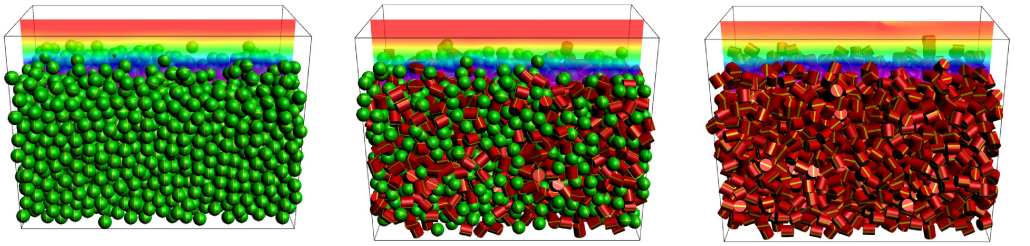


Figure 4.6 | Erosion of three types of granular beds by a laminar shear flow at θ =0.20. From left to right, spheres, a 50/50 mixture of cylinders (red) and spheres (green), cylinders with length over diameter $\frac{\ell}{d}$=1.

Incipient sediment transport under mildly turbulent flow

The same numerical procedure as in Section 4.2.4 was subsequently applied to a granular bed that is large enough to allow for turbulence to develop above it. Also different from the laminar case is that now the flow is generated by a body force f on the liquid in the flow (is x) direction. This body force can be interpreted as an overall pressure gradient in x-direction. The dimensions of the flow domain are $L \times W \times H$ in the stream wise (x), span wise (y) and vertical (z) direction respectively. With a granular bed height of h and a flat, free slip, top surface at $z = H$ the average shear stress at the bed surface is $\tau = f(H - h)$. This shear stress we use in the definition of the Shields parameter

for this flow system: $\theta = \frac{(H-h)f}{2ag(\rho_p-\rho)}$, for defining a wall shear velocity $u_\tau = \sqrt{\frac{f(H-h)}{\rho}}$, and translating length scale into wall units, e.g. for the particle radius: $a^+ \equiv \frac{u_\tau a}{\nu}$. With $W^+ \approx 250$ and $L^+ \approx 500$ the domain is sufficiently large to develop and sustain turbulence [31].

Throughout the simulations described in this sub-section, the density ratio is $\frac{\rho_p}{\rho} = 4.0$, the particles have size such that $a^+ \equiv \frac{u_\tau a}{\nu}$, the bed height is $h \approx 10a$, and $Re_\tau = \frac{u_\tau(H-h)}{\nu}$ =168. The Shields parameter θhas been varied in the range 0.02 to 0.15.

It was first established that with the lattice-Boltzmann method the well-known planar channel turbulence develops in a channel with the parameters given above (see Chapter 5 and [13]).

Impressions of systems with the granular bed in place for two values of θ are given in Figure 4.7 where we have colored the particles by their speed and show an instantaneous liquid velocity distribution in the end plane that demonstrate turbulent liquid flow. Clearly particle mobility gets reduced at low Shields parameters. The end planes ($x = L$) in Figure 4.7 show velocity contours and make clear the turbulent nature of the flow above the granular bed.

Figure 4.8 demonstrates the effect of θ on particle mobility. Different from granular beds under a laminar flow we – as of yet – were not able to create conditions with absolute zero solids motion. Down to $\theta = 0.02$ there consistently was some (albeit minute) average motion of particles in the streamwise direction.

4.2.5 | Liquid fluidization of rigid cylindrical particles

As already shown in Figure 4.6 in Section 4.2.4 on incipient motion of particles under laminar shear flow, our approach based on the lattice-Boltzmann method combined with an immersed boundary method is suited for particle-resolved simulations of solids of non-spherical shape. In this section we look at dense suspensions of rigid cylindrical particles fluidized by an upward vertical liquid flow. A significant set of results has been presented in [16] where the main parameter variations were the overall solids volume fraction ϕ and the length over diameter ($\frac{\ell}{d}$) aspect ratio of the cylinders. One feature that distinguishes cylinders from spheres are the orientational degrees of freedom of the former. In the case of fluidization with purely vertical flow, the orientation that mostly matters is the angle between the vertical and the centerlines of the cylinders.

Monosized rigid cylinders with aspect ratio $\frac{\ell}{d}$ and density ρ_p are placed in a fully periodic three-dimensional domain to a solids volume fraction ϕ along with a Newtonian liquid with density $\rho < \rho_p$ and kinematic viscosity ν in the remainder of the volume. The collection of cylinders feel a net gravity force $-\phi V g\,(\rho_p - \rho)$ with g gravitational acceleration, V the total domain volume and the minus sign to indicate the downward direction. In order to achieve an overall force balanced flow

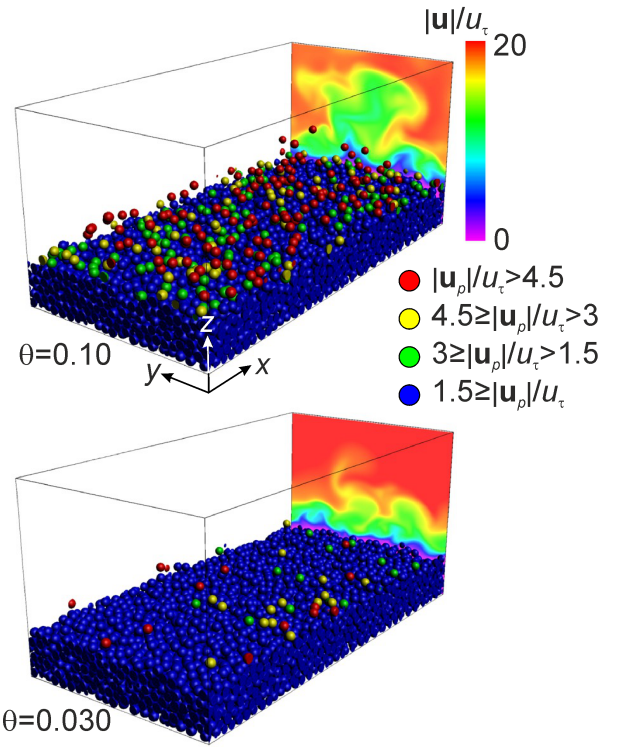


Figure 4.7 | Single realizations of erosion simulations at two different Shields numbers. Particles have been colored by their speed $|\mathbf{u_p}|$ as indicated. The contours in the back plane (x=L) denote the liquid velocity magnitude.

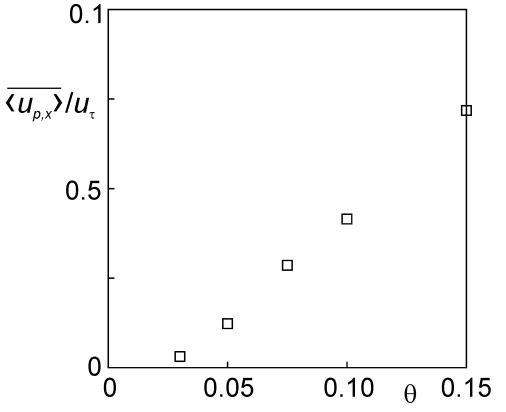


Figure 4.8 | Time and volume average particle velocity in the stream wise direction as a function of the Shields number.

system and reach a dynamic steady state, net gravity on the cylinders is balanced by an upward body force (force per unit volume) on the fluid of $\phi g\,(\rho_p - \rho)$.

Instantaneous realizations of relatively dense systems (ϕ =0.48) after reaching steady state are shown in Figure 4.9. The longer cylinders (with $\frac{\ell}{d}$=4) show a preference for a more vertical orientation, different from shorter ones ($\frac{\ell}{d}$=2). This qualitative observation has been quantified by calculating the distribution function of the angles the centerlines of cylinders make with the vertical, shown in Figure 4.10.

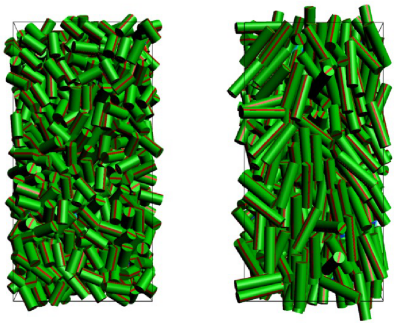


Figure 4.9 | Fluidized cylinders in a periodic domain at ϕ =0.48 after a dynamic steady state has been reached at an Archimedes number of Ar $= (\gamma - 1)\frac{gd^3}{\nu^2} =$ 1296 with $\gamma = \frac{\rho_p}{\rho}$. Left: $\frac{\ell}{d}$=2; right $\frac{\ell}{d}$=4.

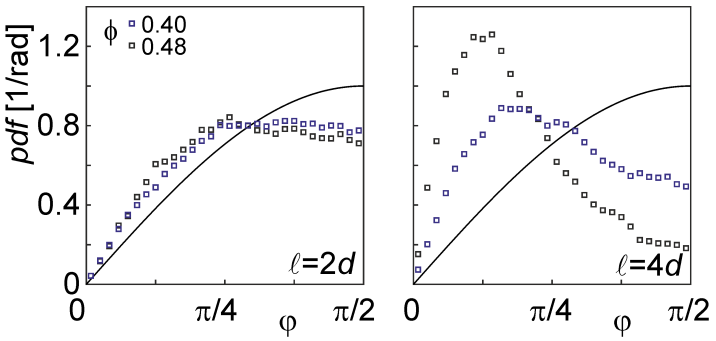


Figure 4.10 | Distributions of the angles φ between cylinder centerlines and the vertical for ϕ =0.48 and 0.40 and Ar =1296. Left: $\frac{\ell}{d}$=2; right $\frac{\ell}{d}$=4. The drawn black curve in each panel is sin φ which is representative for a random orientation distribution.

4.2.6 | Flexible cylindrical particle mechanics

Methodology

In a range of applications solid particles deform under the action of laminar or turbulent fluid flow. In chemical engineering processes one can think of biomass processing given that biomass often contains flexible fibrous components. Under fluidized conditions these fibers bend and entangle which has consequences for fluidizability. In environmental fluid dynamics flow over and through flexible vegetation canopies gives rise to wave-type phenomena (monami and honami) [45]. In this section we give examples of how to incorporate solids deformation in a particle-resolved lattice-Boltzmann simulation. The examples involves cylindrical particles that deform by bending.

The cylinders have diameter d, length ℓ, and density $\rho_p > \rho$. The particle volume is denoted as $V_p = \frac{1}{4}\pi d^2 \ell$ and mass as $m = \rho_p V_p$. The cylinders are bendable, i.e. they have a bending stiffness EI with E Young's modulus and I the moment of inertia of the cross sectional area of the cylinder ($I = \frac{\pi d^4}{64}$). Bending is the only allowed deformation; the cylinders cannot be stretched, compressed or twisted. The immersed boundary method (IBM) as well as the collision algorithm provide us with the distribution of forces over the surfaces of the particles at any moment in time. As we will show below, with this information the bending deformation of the cylinders can be determined.

In the terminology of structural mechanics [65], the cylinder is a beam with bending stiffness (EI) that is deflected by a distributed load (force per unit length along the beam) $a_2(x_1)$ and $a_3(x_1)$ in the two lateral directions x_2 and x_3 respectively (see Figure 4.11), with x_1 the coordinate along the centerline of the beam. The load in the x_1 direction is irrelevant for bending; it would be relevant for stretching or compression which are, however, deformations not considered in this study. The force distribution over the cylinder surface that is the result of the IBM and the particle contact algorithm has three consequences: (1) it accelerates the particle (in a linear and angular sense); (2) it opposes net gravity (if relevant); (3) it bends the particle. In the x_2 and x_3 direction this implies

$$b_2(x_1) = a_2(x_1) + \frac{m}{\ell}\left(\frac{du_{p2}}{dt} + x_1\frac{d\omega_{p3}}{dt} - g_2(\gamma - 1)\right) \tag{4.18}$$

$$b_3(x_1) = a_3(x_1) + \frac{m}{\ell}\left(\frac{du_{p3}}{dt} - x_1\frac{d\omega_{p2}}{dt} - g_3(\gamma - 1)\right) \tag{4.19}$$

with $b_2(x_1)$ and $b_3(x_1)$ the total force per unit length at axial location x_1 in the x_2 and x_3 direction respectively, u_{p2} and u_{p3} components of the linear velocity of the particle, ω_{p2} and ω_{p3} angular velocity components, g_2 and g_3 gravitational acceleration in the x_2 and x_3 direction respectively, and $\gamma = \frac{\rho_p}{\rho}$. Given that the particle acceleration (linear and angular) is solved separately in the

simulation procedure, Eqs. 4.18 and 4.19 allow – at every moment in time for each particle – the determination of the load distributions a_2 and a_3 that bend the particle.

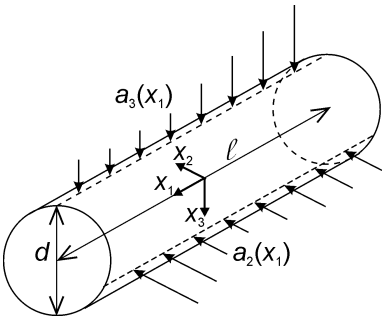


Figure 4.11 | The (x_1, x_2, x_3) coordinate system attached to each cylinder. The load (force per unit length) distributions responsible for bending are indicated by a_2 and a_3.

In a quasi-static approximation, the load distributions relate to bending moments M_2 and M_3 according to [65]

$$a_2 = \frac{d^2 M_2}{dx_1^2} \quad a_3 = \frac{d^2 M_3}{dx_1^2} \tag{4.20}$$

For a freely moving cylinder both ends are unconstrained. Then the boundary conditions for these two second-order ordinary differential equations (ODE's) are that at $x_1 = -\frac{1}{2}\ell$ and $x_1 = \frac{1}{2}\ell$ the bending moments are zero: $M_2\left(x_1 = \pm\frac{1}{2}\ell\right) = M_3\left(x_1 = \pm\frac{1}{2}\ell\right) = 0$.

The deflections of the beam (w_2 and w_3) obey [65]

$$M_2 = EI\frac{d^2 w_2}{dx_1^2} \quad M_3 = EI\frac{d^2 w_3}{dx_1^2} \tag{4.21}$$

in the limit of small deflections, i.e. if $|w_2| \ll d$ and $|w_3| \ll d$. Solving for w_2 and w_3 requires again two boundary conditions each. Since the overall translation and rotation of the cylinder are updated by solving the dynamic equations of the cylinder in its entirety, the deflections are not allowed to add additional overall translation or rotation. Therefore the average deflection as well as the average deflection gradient must be zero. For w_2: $\int_{\frac{-\ell}{2}}^{\frac{\ell}{2}} w_2 dx_1 = 0$ and $\int_{\frac{-\ell}{2}}^{\frac{\ell}{2}} \frac{dw_2}{dx_1} dx_1 = 0$. The latter implies $w_2\left(\frac{-\ell}{2}\right) = w_2\left(\frac{\ell}{2}\right)$. The same boundary conditions apply to w_3.

The sets of ODE's (Eqs. 4.20 and 4.21) are solved through finite differences. The cylinder is divided in n_s equally sized segments with length $\Delta x_1 = \frac{\ell}{n_s}$. The second derivatives are discretized according to a central scheme. As an example, for M_2 this reads $\left.\frac{d^2 M_2}{dx_1^2}\right|_i = \frac{M_{2,i+1}+M_{2,i-1}-2M_{2,i}}{\Delta x_1^2} + O\left(\Delta x_1^2\right)$ with $i = 1 \ldots n_s$ and the i-nodes located in the middle of each segment. For each of the ODE's this leads to a linear system of equations of size n_s in the nodal values of M_2, M_3, w_2, w_3 that is solved directly through Gauss elimination. The bending loads a_2 and a_3 are determined by first integrating the hydrodynamic and contact forces over each segment of the cylinder so as to calculate b_2 and b_3 and then apply Eqs. 4.18 and 4.19.

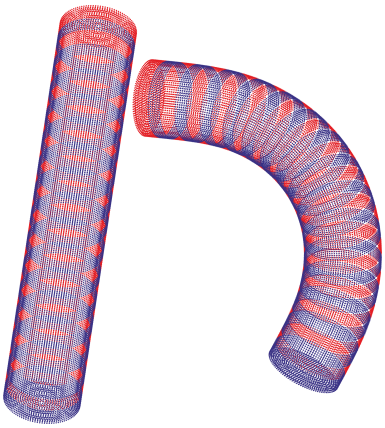


Figure 4.12 | Left: marker points on an undeformed cylinder, the alternating red and blue color indicates the segments; right: marker points on a deformed cylinder.

Once we have calculated the deflections w_2 and w_3 for a cylinder, its shape in terms of marker points needs to be adjusted in order to apply the IBM as well as collision detection and execution at the surfaces of the *deformed* cylinders. The marker points are grouped per segment and the deflection is dealt with as a translation and rotation per segment. Figure 4.12 illustrates how this has been implemented.

Verification

Inspired by experimental work on soft hair beds by Alvarado et al [1] in which they show non-linear behavior as a direct consequence of linear elastic cylinders ("hairs") bending under the influence of a laminar shear flow. In the experimental work [1], a Couette flow of a Newtonian liquid between two parallel solid surfaces was created with the flexible cylinders clamped to one of the surfaces in a regular pattern. The shear stress was measured as a function of the shear rate. The authors observed apparent shear-thinning behavior as a result of the hairs bending under the influence of the shear flow which then leaves more room for fluid flow above the hairs and thus a sub-linear

relation between shear rate and shear stress. These were experiments at low Reynolds numbers with a steady stimulus (shear rate) and steady response (shear stress).

Simulations mimicking the experimental system were carried out with the above described methodology. The response of the clamped flexible cylinders to shear flow is shown in Figure 4.13. One observes increased bending with increased rescaled velocity [1] $\widetilde{v} = \frac{\rho\nu\ell^2 U_w}{E\phi d^2 H}\left(1-\frac{\ell}{H}\right)^{\frac{-3}{2}}$ with U_w the upper wall shearing velocity, H the channel height and ϕ the cylinder area packing fraction on the bottom wall. Increased bending leaves more room for the shear flow above the cylinders. This reduces the shear rate and – as a consequence – the shear stress. This apparent shear-thinning effect as a function of geometrical and process parameters has been documented experimentally in great detail in [1]. Approximately universal behavior was observed when plotting the rescaled impedance $\widetilde{Z}$ as output parameter versus $\widetilde{v}$ as input parameter. Impedance has been defined as $Z \equiv \frac{\tau}{U_w}$ with τ the shear stress at the top surface which is a measurable quantity [1]. Rescaled impedance is $\widetilde{Z} \equiv \frac{Z-Z_\infty}{Z_0-Z_\infty}$ with Z_0 and Z_∞ the impedance under near-zero and infinite shear respectively [17].

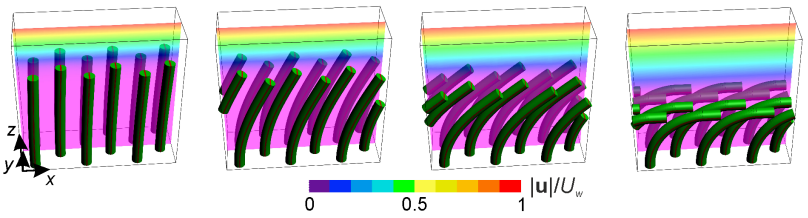


Figure 4.13 | Fibers in a hexagonal arrangement bending in simple shear. From left to right $\widetilde{v}$ =0.234, 2.34, 4.58, 11.7 respectively. All cases have ϕ =0.11, $\frac{\ell}{d}$ =10 and $\frac{\ell}{H}$ =0.73. The periodic condition in x-direction.

We plot the relationship between $\widetilde{v}$ and $\widetilde{Z}$ obtained from simulations in Figure 4.14 in the same manner as was presented in Alvarado et al. The figure also shows model results from [1] that were shown to closely follow the experimental data. The simulation results correctly show the transition towards a reduction of impedance that sets in at $\widetilde{v} \approx 1$ and for larger $\widetilde{v}$ scales as $\widetilde{Z} \sim \widetilde{v}^{\frac{-1}{2}}$. The set of simulations presented in Figure 4.14 include a number of parameter variations: a hexagonal versus a square arrangement of cylinders, variation in surface coverage ϕ as well as in aspect ratios. We observe that $\widetilde{v}$ as an overarching input parameter captures the overall trend in simulated impedance well, as it does in the experimental work.

We thus conclude that the simulations reproduce experimental results on cylinders bending under the influence of steady laminar fluid flow. Next we will explore situations where the flow gets unsteady.

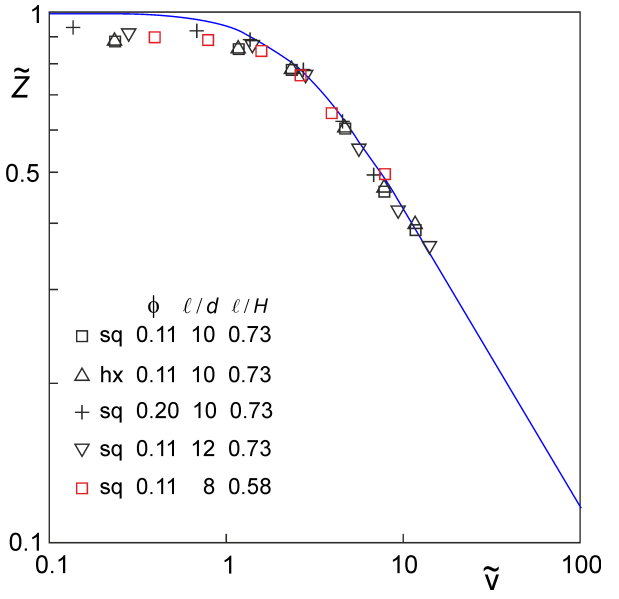


Figure 4.14 | Rescaled velocity $\tilde{v}$ versus rescaled impedance. The symbols represent simulations in various geometrical systems – *sq* is a square pattern of hairs and *hx* a hexagonal pattern. The blue curve is a model due to [1] where it has been shown that experiments closely follow the model curve.

Monami

Monami are wave instabilities in submerged vegetation (e.g. sea grass) induced by liquid flow [45] (the equivalent air flow induced phenomenon of e.g. waiving wheat fields is called honami). We here will focus on monami under laminar conditions whereas a significant part of the monami literature deals with turbulent flow. Stability analyses [57], however, also show clear evidence for the existence of monami due to laminar flow.

In describing the flow systems we will largely follow the notation and terminology as used in [57] where a stability analysis for monami is presented. The flow system consists of an open channel with identical flexible cylinders clamped to the bottom. The main flow is in the x-direction and is driven by a pressure gradient $\frac{dP}{dx}$. The bottom is a no-slip wall with the normal in the z-direction. The top surface is flat and has a free slip boundary condition so as to mimic an open channel without surface waves / deformations. The height of the channel is $2H$. Periodic conditions apply in the streamwise (x) and spanwise (y) directions. The length of the domain is L, its width is W.

The cylinders have diameter d and height ℓ and bending stiffness EI. A number of n cylinders is placed in a regular linear array along the x-direction with center-to-center spacing s and with $W = s$

and $L = ns$. Given the periodic conditions this then mimics a square lattice of cylinders with the flow aligned with the lattice. The working fluid is a Newtonian liquid with kinematic viscosity ν and density ρ. In defining dimensionless quantities we will be using H as the length scale. As in [57], the velocity scale has been chosen as $U_0 = \frac{\left|\frac{dP}{dx}\right| H^2}{(\rho\nu)}$. This defines a Reynolds number $Re = \frac{U_0 H}{\nu}$, with the onset of monami / wave instabilities associated with a critical Reynolds number Re_c [57].

As argued in [57], the length scale δ based on the average velocity and velocity gradient at the top of the bed, defined as $\delta = \left.\frac{U}{\frac{dU}{dz}}\right|_{z=\updownarrow}$, is a measure for the distance over which the flow penetrates the soft hair bed, with $U(z)$ the average fluid velocity profile. This distance decreases with the number of cylinders per unit area (N_g) increasing. With all cylinders having the same diameter d the surface coverage fraction is $\phi = \frac{\pi}{4} d^2 N_g$; for a square lattice of cylinders with spacing s, $\phi = \frac{\frac{\pi}{4} d^2}{s^2}$. The bending stiffness EI gives rise to a dimensionless flexibility parameter defined as $\chi = \frac{\rho U_0^2 \ell^4}{EI}$.

The periodic boundary condition in x-direction has important consequences for the development and characteristics of wave instabilities: The domain size dictates the wavelength λ with an integer number of waves (1 in most cases) fitting in the domain. The dimensionless wave number is defined as $k = \frac{2\pi}{\lambda} H$. The literature [57] reports on critical dimensionless wave numbers k_c with instabilities developing only if $k \leq k_c$. In all simulation cases the length L of the domain was such that $\frac{2\pi}{L} H < k_c$ so that the onset of monami is not expected to be hindered by limitations of the domain length.

Figure 4.15 shows simulation snapshots for one stable and two unstable cases. The cases only differ in terms of the spacing s of the cylinders. The results in Figure 4.15 demonstrate that only beyond a certain spacing, monami develop. The waving oscillations of the cylinder tips are sinusoidal with a very distinct frequency (Figure 4.16).

By performing a number of simulations varying the spacing of the cylinders, their bending stiffness EI and also their arrangement on the bottom wall (in square and in hexagonal patterns) we make a stability map that can be directly compared to the results of the theoretical analysis by [57], see Figure 4.17. Good agreement between the analytical and simulation results is observed.

As in the discussion on soft hair beds (Section 4.2.6) – which was a steady phenomenon – also the dynamic monami phenomenon is captured adequately by our simulation procedure the core of which is based on the lattice-Boltzmann method.

4.3 | Particle-unresolved simulations

Particle-resolved simulations are computationally expensive. Given that the lattice on which the fluid flow is solved needs to have a spacing at least one order of magnitude smaller than the particle

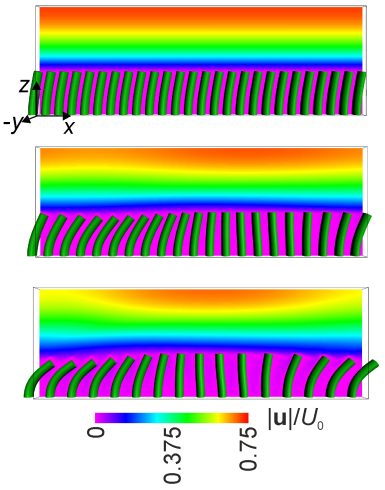


Figure 4.15 | Instantaneous realizations of regular linear arrays of cylinders bending under fluid flow and velocity magnitude color contours in the mid *xz*-plane. From top to bottom the spacing s between the cylinders increases: $s = 1.5d,\ 1.875d,\ 2.5d$ respectively. In all cases Re=1,760 and $\chi = 5.40 \cdot 10^3$. Aspect ratios are $\frac{\ell}{H} = 0.8$, $\frac{\ell}{d} = 5$, $\frac{W}{s} = 1$ and $\frac{L}{H} = 6$. In each panel the cylinder far left is the periodic copy of the one far right.

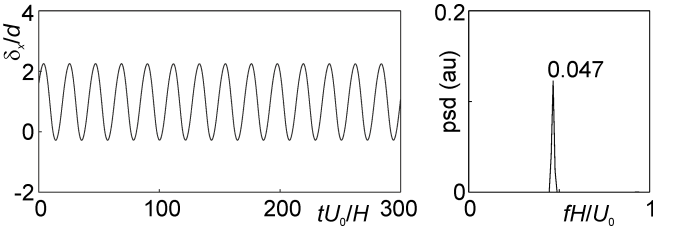


Figure 4.16 | Time series (left) and associated power spectral density (psd) (right) of the tip deflection in *x*-direction of the first cylinder from the left in the row of cylinders. Re=1,760, $\frac{s}{d} = 1.875$, $\chi = 5.40 \cdot 10^3$.

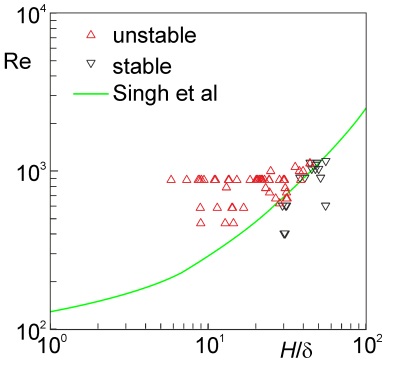


Figure 4.17 | Stability map showing the critical Reynolds number as a function of boundary layer thickness for $\frac{\ell}{H} = 0.8$ due to [57] and the outcome of simulations in terms of stable / unstable. Also in the simulation cases $\frac{\ell}{H} = 0.8$.

size, each particle in a particle-resolved simulation represents 10^3 to 10^4 lattice nodes. This then already makes for large simulations if they contain 10^4 to 10^5 particles. However, with millimetre size particles these only are centimetre size flow systems. If one gives in on the demand to resolve the flow around individual particles and would consider particles with a size comparable to the lattice spacing the number of particles in a single simulation is allowed to increase by a factor of 10^3 to 10^4.

In order for such unresolved-particle (aka "point-particle" or "discrete particle") simulations to be meaningful we need to have a realistic model for the hydrodynamic forces and torques on the individual particles as well as a way to couple back the presence of particles to the fluid flow. Mostly for simplicity we will limit ourselves to spherical particles in this section. Given that we do not resolve the flow around particles it seems – as of now – too ambitious to include particle shape in unresolved particle simulations.

4.3.1 | Volume-averaged Navier-Stokes equations

To account for the volume occupied by the solids in the flow domain we solve volume-averaged mass and momentum balances [54]:

$$\frac{\partial}{\partial t}\left(\rho\phi^c\right) + \nabla\cdot\left(\rho\phi^c\mathbf{u}\right) = 0 \tag{4.22}$$

$$\frac{\partial}{\partial t}\left(\rho\phi^c\mathbf{u}\right) + \nabla\cdot\left(\rho\phi^c\mathbf{u}\mathbf{u}\right) = \phi^c\nabla\cdot\pi + \mathbf{f} + \mathbf{f_s} \tag{4.23}$$

with $\phi^c \equiv 1-\phi$ the continuous phase (liquid) volume fraction, $\mathbf{u}$ the *interstitial* liquid velocity, π the liquid's stress tensor, and $\mathbf{f_s}$ the force per unit volume the solid particles exert on the liquid. Solving these volume-averaged equations requires an extension of the standard lattice-Boltzmann method as it was discussed in Chapter 2. This extension has been detailed in [61].

4.3.2 | Drag force correlations

A major simplification is that the only hydrodynamic force we will be considering in the particle-unresolved simulations is the drag force. For gas-solid systems it is well established that drag is the dominating hydrodynamic force; for liquid-solid systems – with density ratios of order one – this does not necessarily need to be the case. An additional simplification is that drag is assumed to only depend on the local solids volume fraction, and on the Reynolds number based on the slip

velocity of the particle in question $Re = \frac{(1-\phi)|\mathbf{u}-\mathbf{u_p}|d}{\nu}$. We thus do not include terms in the drag expression that depend on the granular temperature (as e.g. proposed in [71]).

In the literature there is extensive activity and debate about the manner in which the drag force depends on ϕ and Re. Much of the recent work is motivated by applications of gas-solid flow (e.g. gas-fluidization, pneumatic conveying) where Stokes numbers are high due to the high solid over fluid density ratios. In such cases, the fluid flow time scales are much shorter than the time scales over which particle configurations change so that one can view drag as the result of the gas flowing through static (and random) assemblies of particles. As shown in [52], the situation for liquid-solid systems that have low to intermediate Stokes numbers is very different. For liquid-solid systems the prevailing approach is the one based on the seminal experiments of solid particles settling in liquids by Richardson and Zaki [51]. In this school of thought, hindered settling is described as $\frac{(1-\phi)|\mathbf{u}-\mathbf{u_p}|}{u_\infty} = (1-\phi)^N$ with a well-established value for the exponent N at low Reynolds numbers: $N = 4.65$ [20], and relatively weak dependencies of Non Re [20] or Re_∞ for moderate Reynolds numbers.

The drag force relation that is the consequence of a Richardson-Zaki type approach can be written as

$$\mathbf{F_D} = 3\pi\rho\nu d\,(\mathbf{u}-\mathbf{u_p})\left(1+0.15Re^{0.687}\right)(1-\phi)^{-\beta} \tag{4.24}$$

with $\beta = N - 2$ and a Schiller-Naumann type [55] account for finite Reynolds number drag. The value of the exponent has been set to the fixed value of $\beta = 2.65$, i.e. independent of Re.

4.3.3 | Two-way coupling and mapping functions

In particle-unresolved simulations we need to transfer information back and forth between the Eulerian (fluid flow) part of the simulation and the Lagrangian (particles) part. For example, in Eq. 4.24 we need to "map" fluid velocity **u** from the Eulerian grid to the direct vicinity of a particle (Lagrangian). The drag force on a particle (Eq. 4.24, Lagrangian) needs to be fed back to the fluid flow (action equals minus reaction) as a body force distribution on the Eulerian grid – the term $\mathbf{f_s}$ in Eq. 4.23. For this we apply mapping functions.

Starting point for our mapping process is a "clipped fourth-order polynomial" [11] $\mu(\xi)$ which shows some resemblance to a Gaussian distribution but is computationally more efficient than a Gaussian. In one dimension:

$$\mu(\xi) = \begin{cases} \frac{15}{16}\left[\frac{\xi^4}{\lambda^5} - 2\frac{\xi^2}{\lambda^3} + \frac{1}{\lambda}\right] & \text{for } -\lambda \leq \xi \leq \lambda \\ 0 & \text{for } |\xi| > \lambda \end{cases} \tag{4.25}$$

The local average at location κ over an averaging length scale 2λ of a one-dimensional function $\alpha(\xi)$ is then determined as $\langle\alpha(\kappa)\rangle_\lambda = \int_{-\lambda}^{\lambda} \mu(\xi-\kappa)\alpha(\xi)d\xi$. In our simulations, $\alpha(\xi)$ is defined on an equidistant grid ξ_i with spacing Δ by values α_i, and we approximate $\alpha(\xi)$ in the integrant as a stair step function, i.e. $\alpha(\xi) = \alpha_i$ for $\xi_i - \frac{1}{2}\Delta \leq \xi < \xi_i + \frac{1}{2}\Delta$.

Such mappings can be readily extended to three dimensions:

$$\langle\alpha(\kappa)\rangle_\lambda = \int_{-\lambda}^{\lambda}\int_{-\lambda}^{\lambda}\int_{-\lambda}^{\lambda} \mu(\xi_1-\kappa_1)\,\mu(\xi_2-\kappa_2)\,\mu(\xi_3-\kappa_3)\,\alpha(\xi)d\xi_1 d\xi_2 d\xi_3 \tag{4.26}$$

Equation 4.26 is used to determine the liquid velocity $\mathbf{u}$ and solids volume fraction ϕ from grid values in the expression for the drag force (Eq. 4.24). The choice of the filter length λ is based on benchmarking with particle-resolved simulation results. For spherical particles with diameter d, $\lambda = 1.5d$ [14].

An inverse mapping operation is required to couple back the hydrodynamic force $-\mathbf{F_D}$ to the liquid. This involves a distribution of the force acting at the location of a particle over grid points surrounding that particle. A similar operation applies for determining the solids volume fraction on the grid: the volume of each particle will be distributed over surrounding grid cells.

Given the discrete nature of $\alpha(\xi)$, Eq. 4.26 can be written as $\langle\alpha(\kappa)\rangle_\lambda = \sum_i\sum_j\sum_k \eta_{ijk}\alpha_{ijk}$ with i, j, k discrete coordinates in x, y, and z-direction respectively, and η_{ijk} coefficients following from integrating the mapping function. The coefficients η_{ijk} are only non-zero on grid points within a volume of $(2\lambda)^3$ around κ, and $\sum_i\sum_j\sum_k \eta_{ijk} = 1$ since for α uniform $\langle\alpha\rangle_\lambda = \alpha$.

The same coefficients η_{ijk} are used to distribute particle properties to the grid. The drag force on one of the particles ($\mathbf{F_D}$) contributes to the body force on the fluid $\mathbf{f_s}$ (see Eq. 4.23) in grid cell i, j, k by an amount $-\frac{1}{\Delta^3}\mathbf{F_D}\eta_{ijk}$; that same particle contributes to the solids volume fraction in cell i, j, k by an amount $\frac{\pi d^3}{6\Delta^3}\eta_{ijk}$. For the total interaction force and the total solids volume fraction in each grid cell (i, j, k) contributions from all particles need to be added up.

For computational reasons we pre-calculate the mapping function integrals η_{ijk} on a subgrid lattice of 11^3 points so as to be able to efficiently execute $\langle\alpha(\kappa)\rangle_\lambda = \sum_i\sum_j\sum_k \eta_{ijk}\alpha_{ijk}$.

4.3.4 | Particle dynamics

The dynamics of the *spherical* solid particles is governed by Newton's equations of motion (Eqs. 4.27 and 4.28) and by the kinematic relation Eq. 4.29, with $\mathbf{u_p}, \omega_\mathbf{p}, \mathbf{x_p}$ the linear velocity, angular velocity, and center location of a spherical particle respectively (note that – because we are dealing with spheres – there is no need to track the angular "location" of the particles), $\mathbf{F_h}$ and $\mathbf{T_h}$ the hydrodynamic force and torque on a particle, and $\mathbf{F_c}$ is the contact force due to particle-particle collisions and lubrication effects. We set a zero contact torque thereby assuming smooth particle surfaces and neglecting tangential lubrication effects.

$$\rho_s \frac{\pi}{6} d^3 \frac{d\mathbf{u_p}}{dt} = \mathbf{F_h} + \mathbf{F_c} - \frac{\pi}{6} d^3 \left(\rho_s - \rho\right) g \mathbf{e_x} \tag{4.27}$$

$$\rho_s \frac{\pi}{60} d^5 \frac{d\omega_\mathbf{p}}{dt} = \mathbf{T_h} \tag{4.28}$$

$$\frac{d\mathbf{x_p}}{dt} = \mathbf{u_p} \tag{4.29}$$

The contact force $\mathbf{F_c}$ consists of two parts: soft-sphere collision forces $\mathbf{F_{ssc}}$ and lubrication forces $\mathbf{F_{lub}}$. Both forces are assumed to be radial forces. This means that they act on the line connecting the two sphere centers involved in a contact.

The soft-sphere collision force is a radial repulsive force proportional to the distance δ over which the spheres overlap: $|\mathbf{F_{ssc}}| = \left(\frac{\pi^2 m}{t_c^2}\right)\delta$ with $m = \frac{\pi \rho_s d^3}{6}$ the mass of a particle, and t_c a parameter that controls the typical time of contact between two particles [52]. Lubrication forces occur when two closely spaced particles move relative to one another. The radial component of the lubrication force (the only component considered here) is the result of a draining liquid film between two approaching particles, and a liquid film filling upon separation. For low Reynolds number film flow, the radial lubrication force on particle j due to particle i can be written as $\mathbf{F_{lub,j}} = F_{lub}\mathbf{n_{ij}}$ with $\mathbf{n_{ij}} = \frac{(\mathbf{x_{p,j}} - \mathbf{x_{p,i}})}{|\mathbf{x_{p,j}} - \mathbf{x_{p,i}}|}$ the unit vector along the line connecting the centers of the two particles, and $F_{lub} = -\frac{3}{8}\frac{\pi \nu \rho d^2 (\mathbf{u_{p,j}} - \mathbf{u_{p,i}})\cdot \mathbf{n_{ij}}}{s}$ with s the minimum distance between particle surfaces [35]. The force on particle i due to j is opposite: $\mathbf{F_{lub,i}} = -F_{lub}\mathbf{n_{ij}}$. In the simulations these expressions have been modified in two ways. (1) A cut-off distance of $0.1d$ has been introduced: for $s \geq 0.1d$ the lubrication force is zero, for $s < 0.1d$ $F_{lub} \propto \frac{1}{s} - \frac{10}{d}$. (2) The lubrication force saturates if $s \leq 10^{-3} d$.

Since collisions between particles and between particles and walls are smooth, the only source of rotation is the hydrodynamic torque $\mathbf{T_h}$ (in Eq. 4.28). It is determined according to a creeping flow approximation: $\mathbf{T_h} = \pi\rho\nu d^3\left(\frac{1}{2}\omega - \omega_\mathbf{p}\right)$ with ω the vorticity of the liquid in the direct vicinity of the particle. The hydrodynamic torque is not fed back to the liquid. As a result, the rotation of the particles has no impact on the overall dynamics of the two-phase flow.

Identifying particles at close range makes use of a link-list algorithm as alluded to in Algorithms A.1.

The particles equations of motion (Eq. 4.27 – 4.29) are solved by means of a split derivative time integration [22]. Such integration enhances stability which is useful in case of modest solid over liquid density ratios, as we have in this chapter.

4.3.5 | Examples of point particle simulations of dense systems

Work done so far with the approach outlined above has mainly focused on verification and validation of numerical aspects [14]. After these had been performed satisfactorily the method has been mainly applied to solid-liquid systems agitated by impellers in mixing tanks [72] which is an application with limited relevance for hydraulics. The verification studies, however, might bear relevance to hydraulics. They consist of sedimentation of uniformly sized spheres towards a flat bottom. The results demonstrate that (1) the method is able to capture a wide range of solids volume fractions, from a dense, packed bed above the bottom to clear liquid well above the bed; (2) shows grid-independent results; (3) provide guidance for choosing the width λ of the mapping function (Eq. 4.25).

In the simulations we create a homogeneous mixture of solids in liquid with a certain overall solids volume fraction $\langle\phi\rangle$. The domain has no-slip top and bottom walls and periodic conditions in the lateral (x and y) directions. Single realizations of liquid and solids velocities are shown in Figure 4.18 where the two panels show the same instant in time of the same (physical) system (including the initial locations of all particles), the only difference being the particle size relative to the lattice spacing $\frac{d}{\Delta}$. In the first place one observes that the interface between clear liquid and solids-laden liquid agrees between the two panels. Closer inspection also reveals good agreement between the flow structures indicating that $\frac{d}{\Delta} = 1.1$ already provides proper spatial resolution.

Numerical effects have been quantified in Figure 4.19. The strongest impact on the average hindered settling velocity ratio $\frac{u_s}{u_\infty}$ comes from the width of the mapping function relative to the particles size. One sees that once $\frac{\lambda}{d} \geq 1.5$ the results are approximately independent of $\frac{\lambda}{d}$. In further simulations [15] we have set $\frac{\lambda}{d} = 1.5$ given that velocity fluctuation levels agree best with results from particle-resolved simulations of this value [14].

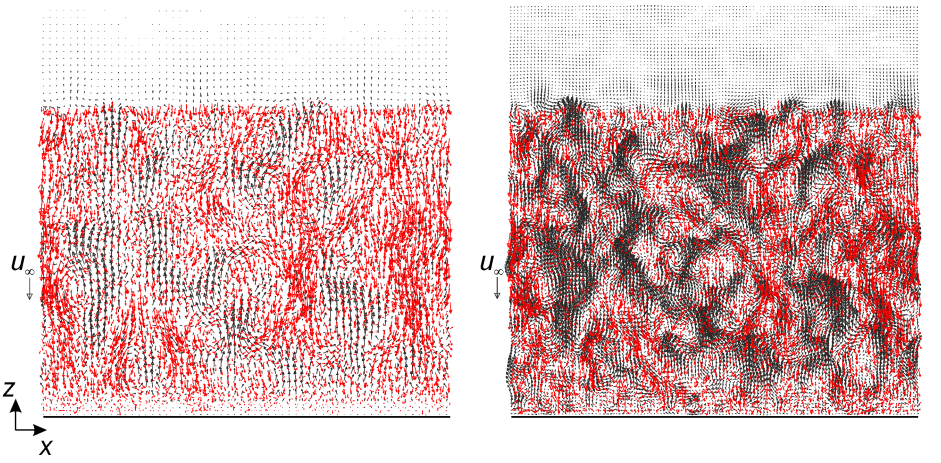


Figure 4.18 | Hindered settling at two resolutions: left $d = 1.1\Delta$, right $d = 2.2\Delta$. Instantaneous realizations at $\frac{tu_\infty}{d} = 13.4$ after start-up. Cross sections through the middle of the flow domain. Black vectors: interstitial liquid velocity; red vectors: velocity of particles in a $4d$ thick layer in the middle of the domain. The reference vector indicates the single-particle settling velocity u_∞.

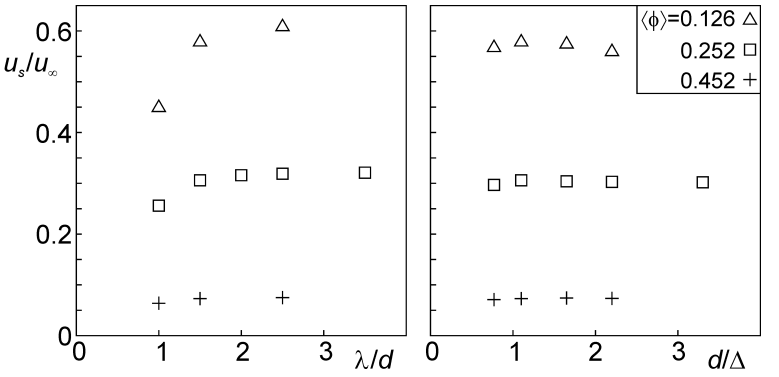


Figure 4.19 | Settling speed as a function of numerical parameters. Left: effect of the half-width of the mapping function (λ) for $d = 1.1\Delta$. Right: effect of particle size relative to grid spacing ($\frac{d}{\Delta}$) for $\lambda = 1.5d$. In all cases, Re_∞ =2.89, $\frac{\rho_s}{\rho} = 2.50$; $\langle\phi\rangle$ as indicated.

4.4 | Closing remarks

In this chapter we have discussed numerical methods for simulating liquid flows laden with solid particles. We restricted ourselves to Eulerian-Lagrangian (EL) methods that track individual particles. We did not consider Eulerian-Eulerian (EE) approaches that treat – next to the liquid – also the solids phase as a continuum. It is important to realize that EL simulations are more a research tool

than an engineering tool. Given the vast amounts of solid particles in e.g. coastal regions and in river beds it is impossible to simulate such systems at full scale in an EL fashion.

However, particle-resolved EL simulations can reveal in great detail the dynamic interactions between solids and liquid and probe dependencies of these interactions on, for instance, particle shape, solids volume fraction and (particle-based) Reynolds and Stokes numbers. With a high computational demand per-particle the domain size of particle-resolved simulations is limited (of the order of centimetres if the particles are order millimetres). Then the relevance of particle-resolved simulations mainly is in informing numerical approaches at larger scale.

Particle-unresolved simulations can accommodate three to four orders of magnitude more particles and thus deal with larger systems thus allowing to get a sense of (length) scale effects on the solids-liquid system behaviour. Eventually the information obtained at smaller scales finds its way – along with experimental data – in EE simulations for (close to) full-scale flow systems.

This chapter shows that the lattice-Boltzmann method in its role as (volume-averaged) Navier-Stokes solver is a fruitful approach for particle-resolved as well as particle-unresolved simulations of solids-liquid systems.

CHAPTER 5

Turbulence simulations with the lattice-Boltzmann method

5.1 | Introduction

In previous chapters, the lattice-Boltzmann method has been described as a flow solver (i.e. as a way to numerically solve the Navier-Stokes equations), and thus it should be capable of solving turbulent flow. In Chapter 4 we saw a glimpse of this capability when addressing sediment transport over a granular bed by mildly turbulent channel flow.

The objective of this chapter is to demonstrate that the LB method indeed agree with long-standing results of direct numerical simulations of turbulence, e.g. of channel flow, and that resolution requirements are similar to those for spectral or finite volume methods (also see [48]). Compared to the latter methods, basic LB simulations are disadvantaged when it comes to inhomogeneous and/or non-isotropic flows. Basic LB simulations use uniform, cubic lattices. For turbulent channel flow [34] the resolution in the wall-normal direction close to the walls needs to be higher than in streamwise and spanwise direction and further away from the walls. In e.g. finite volume grids this can be easily accommodated. In LB simulations, however, the high wall-normal resolution dictates the overall resolution of the simulation thereby over-resolving streamwise and spanwise directions and regions away from walls.

In this chapter we will consider simulations of single-phase turbulent flow with the lattice-Boltzmann method. In the first place we will show that direct simulations of turbulent channel flow produce results – in terms of turbulent statistics – in agreement with other numerical approaches.

We then discuss the suitability of the LB method for performing large-eddy simulations based on an eddy-viscosity subgrid-scale model and give an example of oscillatory flow with relevance for the flow under waves in coastal areas.

5.2 | Direct simulations of turbulent channel flow

We refer to Section 4.2.4 on sediment transport in a channel as a result of a turbulent flow over a granular bed. Prior to the simulations presented there we have ascertained that the single-phase (liquid-only) flow has the correct turbulence behaviour and statistics. We report on this here. Turbulence statistics are well documented in literature. Here comparisons are mainly made with [40]

since this paper is about a channel with a planar free-slip top surface (most turbulent channel flow simulations deal with a no-slip bottom and top surface). The channel has dimensions $L \times W \times H$ in stream wise (x), span wise (y) and wall-normal (z) direction. The flow is driven by a body force f in x-direction which gives rise to a wall shear velocity $u_\tau = \sqrt{\frac{fH}{\rho}}$ with ρ the fluid's density. This defines a Reynolds number as $Re_\tau = \frac{u_\tau H}{\nu}$ that was set to Re_τ =180. We made sure that the flow domain was large enough in stream and span wise direction as to sustain turbulence [31]: $L^+ = \frac{Lu_\tau}{\nu} \approx$500 and $W^+ = \frac{Wu_\tau}{\nu} \approx$250.

The (uniform, cubic) lattice has dimensions $nx \times ny \times nz$ =540×270×180. On the bottom wall (at $z = 0$) no-slip is enforced by the halfway on-grid bounce back boundary condition. The lattice node closest to this wall has z^+ =0.5 which is considered sufficient for resolving the boundary layer [31]. The flow system is evolved with a time step $\Delta t = 0.003 \frac{\nu}{u_\tau^2}$. As discussed in Section 3.2, the time step size is limited by the need to keep the Mach number well below 1 to achieve (near) incompressibility. As discussed in that same section, this is much more restrictive than the CFL criterion which makes for a less efficient numerical scheme. For this particular flow system, also the uniform cubic grid is hampering numerical efficiency significantly. Given that resolution requirements in z-direction are much higher than in the other two coordinate directions a grid of cuboids (instead of cubes) with $\Delta x > \Delta z$ and $\Delta y > \Delta z$ would help much with increasing computational efficiency. The uniformity of the grid thus drives up computational cost. Resolution requirements are highest in the boundary layer so that with a uniform grid the flow regions away from the boundary layer get over-resolved.

The above considerations make an LB simulation of single-phase channel flow a numerically inefficient endeavor. The reasons it is still presented here are twofold: (1) to demonstrate that correct turbulent flow behavior is obtained through the LB method; (2) to realize that when resolved particles are added to this flow simulation system, resolution needs to be high in all coordinate directions and a uniform cubic grid is desirable.

Figures 5.1 and 5.2 show sample results. The most important observation in Figure 5.1 relates to resolution. The figure show an instantaneous velocity vector field in a lateral cross section through the channel. One observes a mildly turbulent flow with flow structures almost as large as the height of the channel. These structures are well – if not overly well – resolved. We need the spatial resolution near the bottom wall for resolving the boundary layer there. It is clear we do not necessarily need the same resolution higher up (i.e. for larger z) in the channel.

Some of the flow's statistics obtained by averaging in time as well as in the homogeneous (x and y) directions are presented in Figure 5.2. The results are (within ~2%) the same as in [40]. For the stream wise average velocity we see a good resolution of the laminar sublayer, and the logarithmic layer.

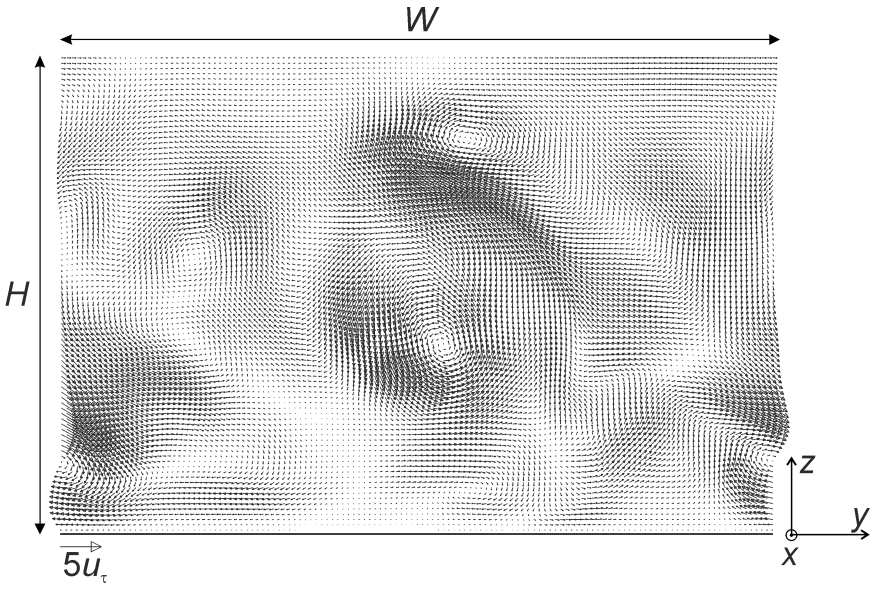


Figure 5.1 | Instantaneous realization of the velocity vector field a in lateral (*yz*) cross section.

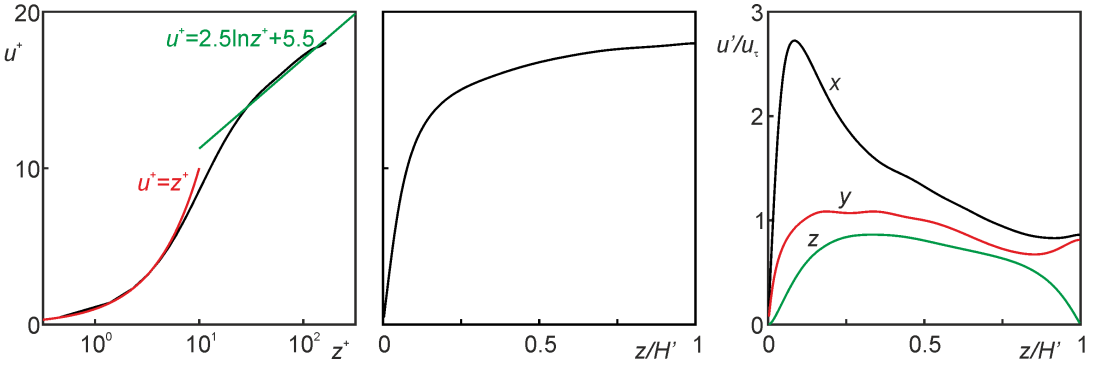


Figure 5.2 | Average wall-normal velocity profiles. Left: average velocity in terms of wall units; middle: average velocity; right: velocity fluctuations in the three coordinate directions as indicated.

The stream wise velocity fluctuation levels peak at $\frac{u'_x}{u_\tau} \approx 0.27$ where u' is the root-mean-square value of the deviations of the velocity component from their respective averages (in x, y and z direction as indicated).

5.3 | Large-eddy simulations with the lattice-Boltzmann method

5.3.1 | Background and implementation

The resolution requirements of direct simulation limits the (Reynolds number) range of its application. If we – for now – discard requirements for resolving wall boundary layers, the main requirement is resolving Kolmogorov eddies the size of which has an order of magnitude of $\eta_K \approx LRe^{-\frac{3}{4}}$ [64] with L a macroscopic length scale of the flow and Re a Reynolds number based on macro quantities.

Performing simulations on grids with spacings (much) wider than η_K need to – in some way – account for flow phenomena smaller than the grid spacing. The basic notion of a large-eddy simulation (LES) is that such subgrid-scale phenomena are diffusive in nature with a diffusivity that depends on the local strength of turbulence. Here it is the intention to discuss how to include a simple subgrid-scale model in a LB simulation, not so much to go over the broader topic of subgrid-scale modeling itself.

As that simple model we take the Smagorinsky model [58] with a constant coefficient c_S. It determines an eddy diffusivity (or kinematic viscosity) as

$$\nu_e = c_S \ell^2 \sqrt{2\bar{s}_{ij}\bar{s}_{ij}} \tag{5.1}$$

(summation over repeated indices i, j) with ℓ the "mixing length" and $\bar{s}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right)$ the deformation rate tensor of the *resolved* velocity field $(\bar{u}_x, \bar{u}_y, \bar{u}_z)$, with "resolved" meaning the large-eddy velocity field.

This model we now apply to an LB simulation on a uniform cubic grid with spacing Δ. The mixing length in Eq. 5.1 is set to $\ell = \Delta$. A peculiar and favorable feature of LB simulations is that $\bar{s}_{ij}$ is directly available in the simulation. It relates to the off-equilibrium part of the distribution function as will be demonstrated below. We thus do not need numerical differentiation of the velocity field to determine $\bar{s}_{ij}$ and subsequently the eddy diffusivity. Once ν_e has been determined the total viscosity is the sum of eddy viscosity and molecular viscosity $\nu_e + \nu$. With a BGK collision operator we then have a relaxation time $\tau = \frac{(\nu_e+\nu)}{3} + \frac{1}{2}$ that is not a constant but depends via Eq. 5.1 on local the local deformation levels and therefore turbulence levels.

To show how the off-equilibrium part of the distribution functions contain $\bar{s}_{ij}$ we refer to Eq. 2.45 that we repeat here as Eq. 5.2:

$$\sum_i c_{i\alpha} c_{i\beta} f_i^{(1)} = -\rho c_s^2 \tau \left[\frac{\partial^{(1)} u_\alpha}{\partial x_\beta} + \frac{\partial^{(1)} u_\beta}{\partial x_\alpha}\right] + \tau \frac{\partial^{(1)}}{\partial x_\gamma}\left(\rho u_\alpha u_\beta u_\gamma\right) \tag{5.2}$$

with $f_i^{(1)}$ the leading term in the non-equilibrium term of the distribution function (Eq. 2.31): $f_i = f_i^{eq} + \varepsilon f_i^{(1)} + \varepsilon^2 f_i^{(2)} + \ldots$. As argued in Chapter 2, the triple velocity term in Eq. 5.2 is negligible for low Mach numbers. And with $\varepsilon \frac{\partial^{(1)}}{\partial x_\beta} \approx \frac{\partial}{\partial x_\beta}$ we find

$$\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha} \approx -\frac{1}{\rho c_s^2 \tau} \sum_i c_{i\alpha} c_{i\beta} \varepsilon f_i^{(1)} \tag{5.3}$$

At any node location and any moment in time the distribution function f_i is calculated. From f_i we determine f_i^{eq} (Eq. 2.26). With $\varepsilon f_i^{(1)} \approx f_i - f_i^{eq}$ we then (with Eq. 5.3) are able to determine the deformation rate tensor and use this for determining the eddy viscosity (Eq. 5.1).

5.3.2 | Large-eddy simulations of oscillatory channel flow

Turbulent oscillatory flow occurs under surface waves, e.g. in coastal regions [46] . Here we deal with oscillatory flow for a single-phase (liquid only flow) application where it is understood that in most cases the relevance only begins when the flow is laden with solid particles. We use LES given the high Reynolds number (defined below) associated with this application.

We consider a sinusoidal oscillatory flow of water with a free stream velocity amplitude of U_a= 1 m/s and a period of t_p=5 s. For comparison we refer to experimental data obtained in a oscillatory flow tunnel [46]. It is not computationally feasible and not necessary to simulate flow in this entire tunnel. To limit computational cost we only simulate a small portion of the tunnel directly above the floor, see the schematic in Fig. 5.3. The height of the simulation domain (nz) is such that it captures roughly two times the boundary layer thickness at the moment of maximum free stream velocity (as observed in the experiments). The computations have periodic conditions in the streamwise (x) and spanwise (y) direction.

The thickness of the boundary layer is – derived from experiments – estimated as δ =15 mm [46]. The height of the domain is represented by 45 lattice spacings (45Δ) and as a result Δ =0.67 mm. The size of the domain in streamwise and spanwise direction is 60 mm and 30 mm respectively. We characterize the flow by two dimensionless numbers: $Re = \frac{\delta U_a}{\nu}$ and $S = \frac{t_p U_a}{\delta}$. With $\nu = 1 \cdot 10^{-6}$ m^2/s, Re =15,000 and S =330. The Reynolds number suggests turbulent flow. Since the boundary layer thickness is captured by only 22.5 lattice spacings we need to apply turbulence modelling for which we use LES with a Smagorinsky subgrid scale model with $c_S = 0.1$.

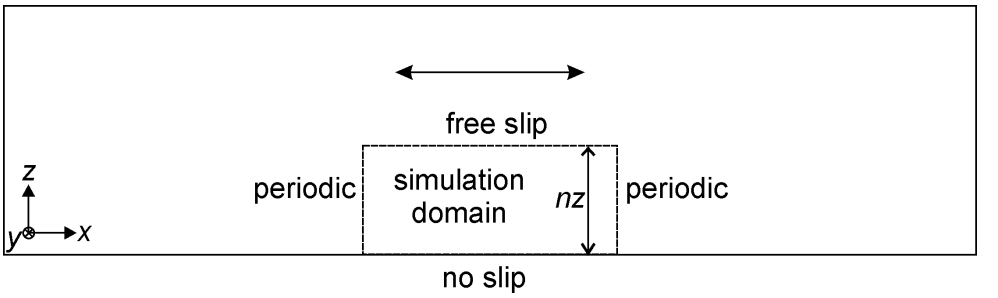


Figure 5.3 | Schematic of the simulation setup for oscillatory flow. The simulation domain is a small volume at the bottom of the much larger measurement section of the much larger flow tunnel.

In the LB simulations we achieve Re =15,000 and S =330 by setting U_a=0.15 (lu) to keep the Mach number low so as to simulate near-incompressible flow. Given that δ =22.5 (lu) the kinematic viscosity is set to $\nu = 2.25 \cdot 10^{-4}$. With $t_p = \frac{S\delta}{U_a}$ =4,500 (lu) which is the number of time steps for one period in the simulation.

The flow if forced by a body force f in x-direction. Where the body force in unidirectional flow was constant (as in Section 5.2), we now control the force. We target a symmetric, sinusoidal free stream velocity as a function of time: $U(t) = U_a \sin\left(2\pi \frac{t}{t_p}\right)$. Every time step we determine the average velocity in the free-stream part of the flow domain outside the boundary layer, i.e. for $z > \delta$ and compare it to the desired, sinusoidal velocity. We adjust the force f in a control loop to keep the actual free stream velocity in check with the aimed for sinusoidal one.

Sample impressions and results are given in Figures 5.4 and 5.5. The flow field realization in the top-left panel of Figure 5.4 shows a phase lag of the near-bottom velocity: where the free stream velocity is at the point of reverting its direction from positive x to negative x, the near-bottom flow is still in positive x-direction. The main flow normal (yz) vector plots indicate relatively strong turbulence near the bottom.

The latter is further explored in Figure 5.5. The phase lag is also observed in the average streamwise velocity profiles at zero free stream velocity (profiles 1 and 5). There are profound effects when it comes to turbulence. The overall levels of turbulence very much depend on time, that is where we are in the cycle with – not surprising – the strongest turbulence at maximum (absolute) mean stream velocity. It is surprising that turbulence dies out quickly away from the peak free-stream velocity levels. At these moments the turbulent kinetic energy is very highly concentrated in the lower part of the boundary layer.

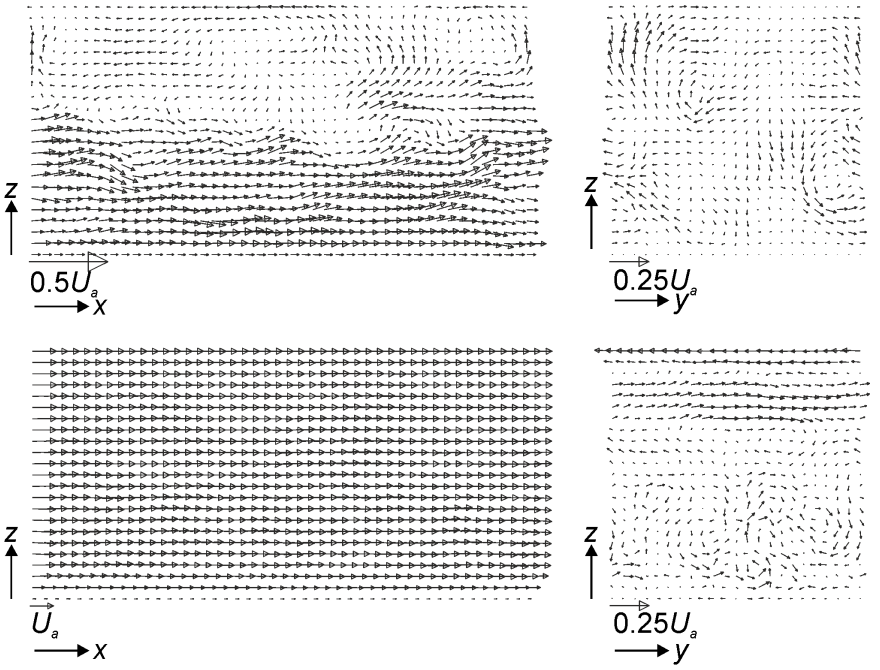


Figure 5.4 | Instantaneous realizations of the velocity field in the form of vectors. Top: the moment when the free stream velocity is zero (on average) coming from a flow in positive *x*-direction; bottom the moment when the free stream has maximum velocity. Left panels: *xz*-plane; right *yz*-plane (the plane normal to the main flow direction).

5.4 | Closing remarks

We have been looking at direct and large-eddy simulations performed based on the lattice-Boltzmann method. While the LB method gives reasonable results for single-phase direct simulations, it is a computational unpractical method for this application when using uniform cubic grids (as in the basic mode of the LB method) given that this leads to over-resolution away from wall boundaries and the inability to stretch grids in certain directions.

Similar remarks as above for large-eddy simulations. On the positive side for LES, part of the solution vector of the LB method directly contains deformation rates which simplifies subgrid-scale modelling, at least for eddy viscosity models.

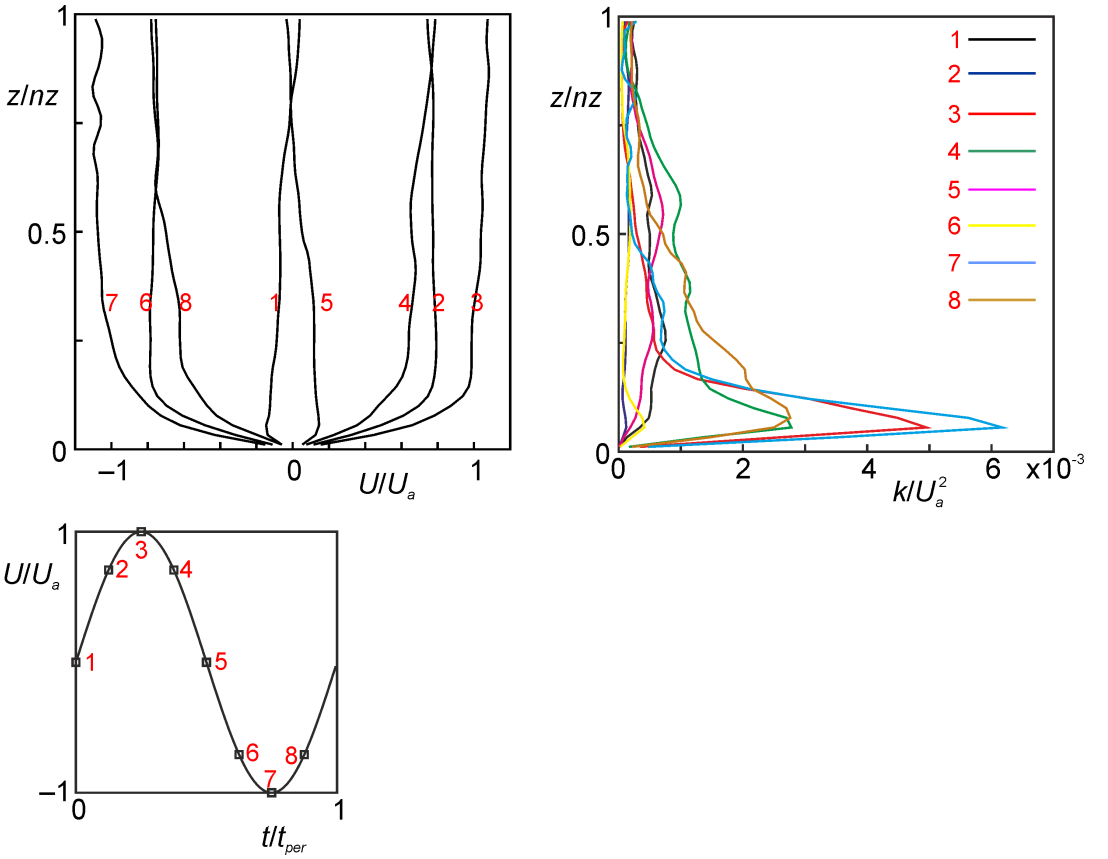


Figure 5.5 | Wall-normal profiles of the streamwise velocity (top left) and turbulent kinetic energy (top right) at eight different moments in the cycle (bottom). The averages are calculated by averaging over the homogeneous (x and y) directions.

CHAPTER 6

Summary & conclusions

The lattice-Boltzmann (LB) method is lattice gas automaton for fluid flow simulations and related physics (such as heat and mass transfer). Its inception we can put at 1986 [25] with lattice gas automata and at 1989 [43] with the lattice Boltzmann equation. Since then it has branched out over physics, mathematics and engineering. In this monograph we have touched upon a few examples of applications in civil engineering. With many approaches to computational mechanics, and more specifically computational fluid dynamics (CFD) such as finite element or finite volume analyses, one way of approaching the LB method is to ask the question why one should use the method in the first place. What are its advantages and disadvantages? As we saw in the monograph, there is no objective answer to this question. In the first place there are personal preferences and experiences. Putting these aside, the second consideration relates to the infrastructure available to engineers and researchers; infrastructure in terms of soft and hardware. And then, thirdly, there is the question what needs to be achieved by computational analyses within the boundaries of feasibility. For example, accuracy, speed or design guidelines of devices involving fluid flow? Speed is an interesting example. In some parts of medical practice a computation would need to be faster than real life so as to make decisions of about surgical intervention impacting on blood flow patterns [53]. In such cases the linear scaling and parallelism of the LB method are assets.

6.1 | Reasons to avoid the lattice-Boltzmann method

There are limits to the accuracy of the method. In terms of spatial discretization, usually 2^{nd} order accuracy is achieved [36] which is generally lower than spectral methods that, in principle, can go to any order [5]. Boundary conditions in the LB method might deteriorate 2^{nd} to 1^{st} order, specifically those boundary conditions based on immersed boundaries.

CFD codes are usually very flexible when it comes to local mesh refinement: refine grids in regions of high velocity gradients. LB methods are less flexible in this respect. However, in specific situations high spatial resolution is required in most of the flow domain, think of particle-resolved suspension simulation in sediment transport or fluidization where particles can be anywhere and thus high resolution is required anywhere and then limited possibilities of local grid refinement are not that much of a disadvantage.

We noted that the lattice Boltzmann fluid is compressible. In many cases, however, it is used to solve incompressible flow – specifically in hydraulics. This is achieved by setting a low Mach number. The consequence of this is that the physical time step needs to be set to a small value, a value typically one order of magnitude below a time step size that would be needed to satisfy the CFL condition in a finite volume or finite difference scheme [23].

6.2 | Reasons to embrace the lattice-Boltzmann method

A core LB code, thanks to its mostly local operations, scales linearly which means that the computational effort increases linearly with the number of degrees of freedom (and thus grid nodes). This is harder to achieve with codes that involve whole-domain operations such as pressure correction schemes in CFD. The local and near-neighbour operations also makes that parallel code only requires communication over subdomain boundaries when a domain decomposition is chosen as the parallelization strategy leading to a near linear (i.e. ideal) speed-up of parallel code.

Other advantages are not so much in the realm of hydraulics but are worth mentioning here. Thermodynamically consistent formulations of LB are beneficial when it comes to flows involving interfaces that exhibit surface tension [62].

6.3 | Lattice-Boltzmann codes & writing code from scratch

An important consideration when thinking of using the LB method in your research is writing your own code, using open-source code or using a commercial code.

A comprehensive list of open-source LB codes is available at [4]. In my perception the codes OpenLB [47] and Palabos [67] are the most versatile and most used codes.

Writing one's own code might be much less daunting than it looks like. LB operations are very basic and require only algebraic calculations. The streaming step (see Section 3.5) is simple as well. The advantage of writing-your-own-code is the learning-by-doing aspect. It forces one to live through the operations and directly observe how the code behaves. It also makes one to know exactly what is going on in the code. In addition one is free to design their own data structures and results files to be used for post-processing. To quote Richard P. Feynman "What I cannot create, I do not understand."

6.4 | Research directions

This monograph is intended to introduce the LB method and to show its potential for hydraulics applications. This makes that it sometimes only scratches the surface of an active research area of research that is in continuous development. In this section I would like to highlight a few recent developments regarding methodological and computational aspects.

Graphical Processing Units (GPU) as a computational platform have shown great potential in comparison to CPU-based systems because of the inherent parallel nature of GPU systems and their processors being designed to do the same computations on different input data in parallel [36].

In explaining the method in this text we have limited ourselves to the single-relaxation time (BGK) *collision operator*. Developing collision operators has been and is an active research area with TRT (two-relaxation time) and MRT (multiple relaxation time) collision operators as highlights. They improve stability at low viscosity and are therefore beneficial for high-Reynolds number flows. As compared to BGK-based schemes they come at higher computational cost per lattice-update. The entropic LB method [8] enhances stability through high-precision calculation of equilibrium distributions. It is particularly suited for schemes with large velocity sets that – for that reason – are highly isotropic.

On the side of hydraulics *applications*, LB-based methods for shallow water flows [75]

as well as for free surface flows [41] need mentioning. The latter combines volume-of-fluid surface capturing with an LB approach to study tsunami's over complex terrain.

Appendix

Algorithms

A.1. Creating a link-list of particles that helps identifying particles nearby a reference particle

The particles are a list of length npt of coordinates pcx,pcy,pcz and are in a 3D domain nx ny nz. Define an integer array fe(i,j,k) and an integer array nxt(m). Loop over all particles npt; if particle m falls in cell i,j,k and the cell is empty (no particle in there yet) then change fe(i,j,k) from 0 to m; if the cell is not empty, fe(i,j,k)=m1. Use the array nxt (stands for "next") to make m the next particle in cell i,j,k.

```
do k=1,nz
  do j=1,ny
    do i=1,nx
      fe(i,j,k)=0
    enddo
  enddo
enddo
c
do m=1,npt
  nxt(m)=0
  i=int(pcx(m))+1
  j=int(pcy(m))+1
  k=int(pcz(m))+1
  if (fe(i,j,k).eq.0) then
    fe(i,j,k)=m
  else
    m1=fe(i,j,k)
    do while (nxt(m1).ne.0)
      m1=nxt(m1)
    enddo
    nxt(m1)=m
  endif
enddo
```

Now loop over the mesh i,j,k:

```
do k=1,nz
  do j=1,ny
    do i=1,nx
      m=fe(i,j,k)
      do while (m.ne.0)
        do k1=-sr,sr
          kpk1=k+k1
          do j1=-sr,sr
            jpj1=j+j1
            do i1=-sr,sr
              ipi1=i+i1
              do while (m1.ne.0)
                if (m1.gt.m) then
                  c... identify nearby particles m and m1
                  c... calculate interaction forces / collisions
                  ......
                  m1=nxt(m1)
                enddo
              enddo
            enddo
          enddo
          m=nxt(m)
        enddo
      enddo
    enddo
  enddo
```

A.2. Compute forces so as to prescribe velocity at a set of marker points — immersed boundary method

We have npt particles; each particle has namp(v) marker points with v the type of particle (e.g. sphere or cylinder). The first triple loop is interpolation (Eq. 4.1); the second triple loop is extrapolation (Eqs. 4.2 and 4.3).

```
do m=1,npt
  c... m is the particle counter; v is the particle type
  v=1
  if (m.gt.np(1)) v=2
  do u=1,namp(v)
    c... xp,yp,zp are marker points coordinates in
    c... inertial frame of reference
    c... mp1,mp2,mp3 are marker points in particle
    c... frame of reference
    c... s is the rotation matrix (see Eq. 4.17)
    xp=s(1,1,m)*mp1(u,v)+s(1,2,m)*mp2(u,v)+s(1,3,m)*mp3(u,v)+
        > pcx(m)
    yp=s(2,1,m)*mp1(u,v)+s(2,2,m)*mp2(u,v)+s(2,3,m)*mp3(u,v)+
        > pcy(m)
    zp=s(3,1,m)*mp1(u,v)+s(3,2,m)*mp2(u,v)+s(3,3,m)*mp3(u,v)+
        > pcz(m)
    i=int(xp+0.5d0)
    j=int(yp+0.5d0)
    k=int(zp+0.5d0)
    xpl=xp-1.0*i+0.5
    ypl=yp-1.0*j+0.5
    zpl=zp-1.0*k+0.5
    c... the coefficients routine cff3d is listed below
    call cff3d(xpl,ypl,zpl,p3)
    xs=0.0
    ys=0.0
    zs=0.0
    rho=0.0
    do k1=0,1
      kpk1=k+k1
      do j1=0,1
        jpj1=j+j1
        do i1=0,1
          ipi1=i+i1
          xs=xs+p3(i1,j1,k1)*(x(2,ipi1,jpj1,kpk1)+
              > 0.5*fxo(ipi1,jpj1,kpk1))
```

```
        ys=ys+p3(i1,j1,k1)*(x(3,ipi1,jpj1,kpk1)+
            > 0.5*fyo(ipi1,jpj1,kpk1))
        zs=zs+p3(i1,j1,k1)*(x(4,ipi1,jpj1,kpk1)+
            > 0.5*fzo(ipi1,jpj1,kpk1))
        rho=rho+p3(i1,j1,k1)*x(1,ipi1,jpj1,kpk1)
      enddo
    enddo
  enddo
c
c...linear motion & rotational surface motion of particles
c
vr1=om2(m)*mp3(u,v)-om3(m)*mp2(u,v)
vr2=om3(m)*mp1(u,v)-om1(m)*mp3(u,v)
vr3=om1(m)*mp2(u,v)-om2(m)*mp1(u,v)
vsx=s(1,1,m)*vr1+s(1,2,m)*vr2+s(1,3,m)*vr3+vpx(m)
vsy=s(2,1,m)*vr1+s(2,2,m)*vr2+s(2,3,m)*vr3+vpy(m)
vsz=s(3,1,m)*vr1+s(3,2,m)*vr2+s(3,3,m)*vr3+vpz(m)
c
xs=xs-rho*vsx
ys=ys-rho*vsy
zs=zs-rho*vsz
c
do k1=0,1
  kpk1=k+k1
  do j1=0,1
    jpj1=j+j1
    do i1=0,1
      ipi1=i+i1
      fxo(ipi1,jpj1,kpk1)=alfa*fxo(ipi1,jpj1,kpk1)-
          > beta*p3(i1,j1,k1)*xs
      fyo(ipi1,jpj1,kpk1)=alfa*fyo(ipi1,jpj1,kpk1)-
          > beta*p3(i1,j1,k1)*ys
      fzo(ipi1,jpj1,kpk1)=alfa*fzo(ipi1,jpj1,kpk1)-
          > beta*p3(i1,j1,k1)*zs
    enddo
  enddo
```

```
    enddo
  enddo
enddo
subroutine cff3d(a,s,z,p)
c
c...calculate coefficients for interpolation:
c...input are a(=eta), s(=ksi) and z(=zeta);
c...output is the coefficient matrix p(0:1,0:1,0:1)
c
integer*4 i,j,k
real*8 a,s,z,p(0:1,0:1,0:1),
    > f1
c
do k=0,1
  do j=0,1
    do i=0,1
      p(i,j,k)=f1(i,a)*f1(j,s)*f1(k,z)
    enddo
  enddo
enddo
end
function f1(l,eta)
real*8 f1,eta
integer*4 l
if (l .eq. 0) f1=1.0-eta
if (l .eq. 1) f1=eta
end
```

A.3. Quaternion updates

Refer to exact solution paper [37]. Update quaternions q0...q3 based on angular velocities om1,om2,om3.

```
nrm=dsqrt(om1(m)*om1(m)+om2(m)*om2(m)+om3(m)*om3(m))
dt=nrm
if (nrm.ge.1d-8) then
```

```
  o1=om1(m)/nrm
  o2=om2(m)/nrm
  o3=om3(m)/nrm
else
  o1=0d0
  o2=0d0
  o3=0d0
  dt=0d0
endif
a0=dcos(0.5d0*dt)
sn=dsin(0.5d0*dt)
a1=o1*sn
a2=o2*sn
a3=o3*sn
q0n=q0(m)*a0-q1(m)*a1-q2(m)*a2-q3(m)*a3
q1n=q0(m)*a1+q1(m)*a0+q2(m)*a3-q3(m)*a2
q2n=q0(m)*a2-q1(m)*a3+q2(m)*a0+q3(m)*a1
q3n=q0(m)*a3+q1(m)*a2-q2(m)*a1+q3(m)*a0
c nrm=dsqrt(q0n**2+q1n**2+q2n**2+q3n**2)
nrm=1d0
q0(m)=q0n/nrm
q1(m)=q1n/nrm
q2(m)=q2n/nrm
q3(m)=q3n/nrm
enddo
```

A.4. Pre-calculate mapping function integrals according to Eq. 4.26**.** For this we create in each i2,j2,k2 cell a three dimensional grid of 11^3 points where we calculate $\eta_{i2,j2,k2}$. The routine mp performs the integral of the function sg. The latter has been expressed in Eq. 4.25.

```
diam=2d0
r0=1.5d0*diam
do k=1,11
  do j=1,11
    do i=1,11
      x0=0.1d0*(1d0*i-1d0)
```

```
      y0=0.1d0*(1d0*j-1d0)
      z0=0.1d0*(1d0*k-1d0)
      call mp(x0,y0,z0,r0,mvl)
      do k2=-mpr,mpr
        do j2=-mpr,mpr
          do i2=-mpr,mpr
            mmvl(i,j,k,i2,j2,k2)=mvl(i2,j2,k2)
          enddo
        enddo
      enddo
    enddo
  enddo
enddo
c... write the pre-calculated coefficients in a file
c... for use in the actual simulation algorithm
do k=1,11
  do j=1,11
    do i=1,11
      write(99) (((mmvl(i,j,k,i2,j2,k2),i2=-mpr,mpr),
          > j2=-mpr,mpr),k2=-mpr,mpr)
    enddo
  enddo
enddo
c
end
subroutine mp(x0,y0,z0,r0,mvl)
implicit none
integer*4 mpr
parameter (mpr=3)
integer*4 i,j,k
real*8 x0,y0,z0,r0,lb,ub,ss,
    > sz(-mpr:mpr),sy(-mpr:mpr),sx(-mpr:mpr),
    > mvl(-mpr:mpr,-mpr:mpr,-mpr:mpr)
do k=-mpr,mpr
  sz(k)=0d0
  lb=1d0*k
```

```
  ub=1d0*(k+1)
  if ((z0-r0.lt.ub).and.(z0+r0.gt.lb)) then
    if (z0-r0.gt.lb) lb=z0-r0
    if (z0+r0.lt.ub) ub=z0+r0
    call sg(lb-z0,ub-z0,r0,ss)
    sz(k)=ss
  endif
enddo
c
do j=-mpr,mpr
  sy(j)=0d0
  lb=1d0*j
  ub=1d0*(j+1)
  if ((y0-r0.lt.ub).and.(y0+r0.gt.lb)) then
    if (y0-r0.gt.lb) lb=y0-r0
    if (y0+r0.lt.ub) ub=y0+r0
    call sg(lb-y0,ub-y0,r0,ss)
    sy(j)=ss
  endif
enddo
c
do i=-mpr,mpr
  sx(i)=0d0
  lb=1d0*i
  ub=1d0*(i+1)
  if ((x0-r0.lt.ub).and.(x0+r0.gt.lb)) then
    if (x0-r0.gt.lb) lb=x0-r0
    if (x0+r0.lt.ub) ub=x0+r0
    call sg(lb-x0,ub-x0,r0,ss)
    sx(i)=ss
  endif
enddo
c
do k=-mpr,mpr
  do j=-mpr,mpr
    do i=-mpr,mpr
```

```
      mvl(i,j,k)=sx(i)*sy(j)*sz(k)
    enddo
  enddo
enddo
end
subroutine sg(lb,ub,r0,ss)
implicit none
real*8 lb,ub,r0,ss
ss=15d0*(0.2d0*(ub**5-lb**5)/(r0**5)-
    > 2d0*(ub**3-lb**3)/(3d0*(r0**3))+
    > (ub-lb)/r0)/16d0
end
```

References

1 | J. Alvarado, J. Comtet, E. de Langre, and A. E. Hosoi. Nonlinear flow response of soft hair beds. *Nature Physics*, 13:1014–1019, 2017.

2 | G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, Cambridge, 1967.

3 | P. L. Bhatnagar, E. P. Gross, and M. Krook. A model for collision processes in gases. I: small amplitude processes in charged and neutral one-component system. *Physical Review*, 94:511–525, 1954.

4 | Sthavishtha Bhopalam. Curated list of open source codes employing lattice Boltzmann methods, 2024. Accessed: 12 August 2024.

5 | C. Canuto, M. Yousuff Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, Berlin Heidelberg, 1988.

6 | S. Chapman and T. G. Cowling. *The Mathematical Theory of Non-Uniform Gases*. Cambridge University Press, Cambridge, 2nd edition, 1952.

7 | S. Chen and G. D. Doolen. Lattice–Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30:329–364, 1998.

8 | S. S. Chikatamarla, S. Ansumali, and I. V. Karlin. Entropic lattice Boltzmann models for hydrodynamics in three dimensions. *Physical Review Letters*, 97:010201, 2006.

9 | K. Connington and T. Lee. A review of spurious currents in the lattice Boltzmann method for multiphase flows. *Journal of Mechanical Science and Technology*, 26:3857–3863, 2012.

10 | C. Crowe, M. Sommerfeld, and Y. Tsuji. *Multiphase Flows with Droplets and Particles*. CRC Press, Boca Raton, 1998.

11 | N. G. Deen, M. van Sint Annaland, and J. A. M. Kuipers. Multi-scale modeling of dispersed gas–liquid two-phase flow. *Chemical Engineering Science*, 59:1853–1861, 2004.

12 | J. J. Derksen. Simulations of granular bed erosion due to laminar shear flow near the critical Shields number. *Physics of Fluids*, 23:113303, 2011.

13 | J. J. Derksen. Simulations of granular bed erosion due to a mildly turbulent shear flow. *Journal of Hydraulic Research*, 53:622–632, 2015.

14 | J. J. Derksen. Assessing Eulerian–Lagrangian simulations of dense solid-liquid suspensions settling under gravity. *Computers & Fluids*, 176:266–275, 2018.

15 | J. J. Derksen. Eulerian–Lagrangian simulations of settling and agitated dense solid-liquid suspensions – achieving grid convergence. *AIChE Journal*, 64:1147–1158, 2018.

16 | J. J. Derksen. Liquid fluidization with cylindrical particles: Highly resolved simulations. *AIChE Journal*, 65:e16594, 2019.

17 | J. J. Derksen. Bending and low-frequency vortex shedding of flexible cylinders in laminar shear flow. *AIChE Journal*, 67:e17268, 2021.

18 | J. J. Derksen and S. Sundaresan. Direct numerical simulations of dense suspensions: wave instabilities in liquid-fluidized beds. *Journal of Fluid Mechanics*, 587:303–336, 2007.

19 | J. J. Derksen and H. E. A. Van den Akker. Large eddy simulations on the flow driven by a Rushton turbine. *AIChE Journal*, 45:209–221, 1999.

20 | R. Di Felici. The voidage function for fluid-particle interaction systems. *International Journal of Multiphase Flow*, 20:155–159, 1994.

21 | Z. Feng and E. Michaelides. The immersed boundary-lattice Boltzmann method for solving fluid-particles interaction problems. *Journal of Computational Physics*, 195:602–628, 2004.

22 | Z. Feng and E. Michaelides. Robust treatment of no-slip boundary condition and velocity updating for the lattice-Boltzmann simulation of particulate flows. *Computers & Fluids*, 38:370–381, 2009.

23 | J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer, 1999.

24 | D. Frenkel and B. Smit. *Understanding Molecular Simulation*. Academic Press, London, 2nd edition, 2001.

25 | U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier–Stokes equations. *Physical Review Letters*, 56:1505–1508, 1986.

26 | M. Gardner. The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223:120–123, 1970.

27 | D. Goldstein, R. Handler, and L. Sirovich. Modeling a no-slip flow boundary with an external force field. *Journal of Computational Physics*, 105:354–366, 1993.

28 | F. Huang, Z. Chen, Z. Li, Z. Gao, J. Derksen, and A. Komrakova. Numerical study of drop behavior in a pore space. *Chemical Engineering Science*, 233:116351, 2021.

29 | Z. Huang, Y. Cheng, J. Wu, W. Diao, and W. Huai. Fsi simulation of dynamics of fish passing through a tubular turbine based on the immersed boundary-lattice Boltzmann coupling scheme. *Journal of Hydrodynamics*, 34:135–147, 2022.

30 | T. Inamuro, T. Ogato, S. Tajima, and J. Konishi. A lattice Boltzmann method for incompressible two-phase flows with large density differences. *Journal of Computational Physics*, 198:628–644, 2004.

31 | J. Jimenez and P. Moin. The minimal flow unit in wall turbulence. *Journal of Fluid Mechanics*, 225:213–240, 1991.

32 | W. Kauzmann. *Kinetic Theory of Gases*. Dover Books, Chatham, 1966.

33 | A. G. Kidanemariam and M. Uhlmann. Formation of sediment patterns in channel flow: minimum unstable systems and their temporal evolution. *Journal of Fluid Mechanics*, 818:716–743, 2017.

34 | J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully-developed channel flow at low Reynolds number. *Journal of Fluid Mechanics*, 177:133–166, 1987.

35 | S. Kim and S. J. Karrila. *Microhydrodynamics: Principles and Selected Applications*. Butterworth-Heinemann, 1991.

36 | T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, S. Silva, and E. M. Viggen. *The Lattice Boltzmann Method: Principle and Practice*. Springer, 2017.

37 | J. B. Kuipers. *Quaternions and Rotation Sequences*. Princeton University Press, Princeton, 1999.

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

39 | A. J. C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 2. Numerical results. *Journal of Fluid Mechanics*, 271:311–339, 1994.

40 | K. Lam and S. Banerjee. On the condition of streak formation in a bounded turbulent flow. *Physics of Fluids A*, 4:306–320, 1992.

41 | G. Liu, J. Zhang, Q. Zhang, and X. Yu. Lattice Boltzmann modeling of tsunami run-up over complex terrain with free surface capturing. *Ocean Engineering*, 329:121096, 2025.

42 | J. Y. Liu and K. J. Hanley. CFD–DEM modelling of the infiltration of non-spherical slurry particles in granular soils. *Computers and Geotechnics*, 164:105845, 2023.

43 | G. R. McNamara and G. Zanetti. Use of the Boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, 61:2332–2335, 1989.

44 | W. J. Moore. *Physical Chemistry*. Prentice Hall, Englewood Cliffs, NJ, 1972.

45 | H. M. Nepf. Flow and transport in regions with aquatic vegetation. *Annual Review of Fluid Mechanics*, 44:123–142, 2012.

46 | T. O'Donoghue and S. Wright. Concentration in oscillatory sheet flow for well sorted and graded sand. *Coastal Engineering*, 50:117–138, 2004.

47 | OpenLB Project. OpenLB – open source lattice Boltzmann code, 2024. Accessed: 12 August 2024.

48 | Y. Peng, W. Liao, L.-S. Luo, and L.-P. Wang. Comparison of the lattice Boltzmann and pseudo-spectral methods for decaying turbulence: Low-order statistics. *Computers & Fluids*, 39:568–591, 2010.

49 | C. Peskin. Flow patterns around heart valves: a numerical study. *Journal of Computational Physics*, 10:252–271, 1972.

50 | C. M. Pooley, H. Kusumaatmaja, and J. M. Yeomans. Contact line dynamics in binary lattice Boltzmann simulations. *Physical Review E*, 78:056709, 2008.

51 | J. F. Richardson and W. N. Zaki. Sedimentation and fluidisation. Part 1. *Transactions of the Institution of Chemical Engineers*, 32:35–53, 1954.

52 | G. J. Rubinstein, J. J. Derksen, and S. Sundaresan. Lattice Boltzmann simulations of low-Reynolds-number flow past fluidized spheres: effect of Stokes number on drag force. *Journal of Fluid Mechanics*, 788:576–601, 2016.

53 | S. Sankaran, D. Lesage, R. Tombropoulos, N. Xiao, H. J. Kim, D. Spain, M. Schaap, and C. A. Taylor. Physics driven real-time blood flow simulations. *Computer Methods in Applied Mechanics and Engineering*, 364:112963, 2020.

54 | K. Sankaranarayanan and S. Sundaresan. Lattice Boltzmann simulation of two-fluid model equations. *Industrial & Engineering Chemistry Research*, 47:9165–9173, 2008.

55 | L. Schiller and A. Naumann. Über die grundlagenden Berechnungen bei der Schwerkraftaufbereitung. *Verein Deutscher Ingenieure Zeitschrift*, 77:318–320, 1933.

56 | X. Shan and H. Chen. Lattice Boltzmann model for simulating flows with multiple phases and components. *Physical Review E*, 47:1815–1819, 1993.

57 | R. Singh, M. M. Bandi, A. Mahadevan, and S. Mandre. Linear stability analysis for monami in a submerged seagrass bed. *Journal of Fluid Mechanics*, 786:R1, 2015.

58 | J. Smagorinsky. General circulation experiments with the primitive equations: 1. The basic experiment. *Monthly Weather Review*, 91:99–164, 1963.

59 | S. Succi. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Clarendon Press, Oxford, 2001.

60 | K. Suga, Y. Kuwata, K. Takashima, and R. Chikasue. A D3Q27 multiple-relaxation-time lattice Boltzmann method for turbulent flows. *Computers & Mathematics with Applications*, 69:518–529, 2015.

61 | R. Sungkorn and J. J. Derksen. Simulations of dilute sedimenting suspensions at finite-particle Reynolds numbers. *Physics of Fluids*, 24:123303, 2012.

62 | M. R. Swift, E. Orlandini, W. R. Osborn, and J. M. Yeomans. Lattice Boltzmann simulations of liquid–gas and binary fluid systems. *Physical Review E*, 54:5041–5052, 1996.

63 | A. Ten Cate, J. J. Derksen, L. Portela, and H. E. A. Van den Akker. Fully resolved simulations of colliding spheres in forced isotropic turbulence. *Journal of Fluid Mechanics*, 519:233–271, 2004.

64 | H. Tennekes and J. L. Lumley. *A First Course in Turbulence*. MIT Press, Cambridge, MA, 1972.

65 | S. P. Timoshenko and J. M. Gere. *Mechanics of Materials*. Van Nostrand Reinhold, New York, 1973.

66 | M. Uhlmann. An immersed boundary method with direct forcing for the simulation of particulate flows. *Journal of Computational Physics*, 209:448–476, 2005.

67 | University of Geneva. Palabos – parallel lattice Boltzmann solver, 2024. Accessed: 12 August 2024.

68 | E. Van Vliet, J. J. Derksen, H. E. A. Van den Akker, and R. O. Fox. Numerical study on the turbulent reacting flow in the vicinity of the injector of an LDPE tubular reactor. *Chemical Engineering Science*, 62:2435–2444, 2007.

69 | E. M. Viggen. Acoustic equations of state for simple lattice Boltzmann velocity sets. *Physical Review E*, 90:013310, 2014.

70 | J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, 1953.

71 | J. J. Wylie, D. L. Koch, and A. J. C. Ladd. Rheology of suspensions with high particle inertia and moderate fluid inertia. *Journal of Fluid Mechanics*, 480:95–118, 2003.

72 | J. Xie, L. Zhang, and J. J. Derksen. Detailed simulations of particle breakage as a result of agitation. *Chemical Engineering Research and Design*, 168:298–306, 2021.

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

74 | H. Yu, L.-S. Luo, and S. S. Girimaji. LES of turbulent square jet flow using an MRT lattice Boltzmann model. *Computers & Fluids*, 35:957–965, 2006.

75 | J. G. Zhou. *Lattice Boltzmann Methods for Shallow Water Flows*. Springer, Berlin, 2004.

The author demonstrates deep expertise in both the numerical method and the physical problems addressed, making this a valuable resource for researchers and students in the field.

ABOUT THE AUTHOR

Professor Jos Derksen was educated in the Netherlands. He received an MSc degree in mechanical engineering from the University of Twente and a PhD degree in engineering physics of the Technical University of Eindhoven. After a stint at the Netherlands aerospace laboratory (NLR) he became assistant professor at the Applied Physics Department of Delft University of Technology. He then moved to Alberta, Canada, to take up a professorship in the department of Chemical and Materials Engineering at the University of Alberta. Subsequently he came back to Europe to become a Chair in Engineering at the University of Aberdeen (UK). He teaches (computational) fluid dynamics, thermodynamics, and numerical methods. His research focuses on detailed simulations of multi-phase flow, mostly solid-liquid systems. He has published extensively in leading engineering and physics journals and delivered multiple invited and keynote lectures on the topic of this monograph. Currently he leads the Fluid Mechanics Research Group at the University of Aberdeen.

ISBN 978-90-835589-9-8

IAHR Global Secretariat
iahr@iahr.org

Madrid Office
Paseo Bajo Virgen del Puerto, 3
28005 Madrid, SPAIN
T +34 91 335 7908
F +34 91 335 7935

Beijing Office
A-1 Fuxing Road, Haidian District
100038 Beijing, CHINA
T +86 10 6878 1128
F +86 10 6878 1890

IAHR.org