Computers & Geosciences 36 (2010) 115–122

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

LBflow: An extensible lattice Boltzmann framework for the simulation of geophysical flows. Part I: theory and implementation ☆

E.W. Llewellin

Department of Earth Sciences, Durham University, DH1 3LE, UK

ARTICLE INFO

Article history:
Received 24 October 2006
Received in revised form
20 July 2009
Accepted 18 August 2009

Keywords:
Lattice Boltzmann method
Geophysical fluid dynamics
Flow simulation
Flow visualization
Computational steering

ABSTRACT

This article presents LBflow, a flexible, extensible implementation of the lattice Boltzmann method. The code has been developed with geophysical applications in mind, and is designed to be usable by those with no specialist computational fluid dynamics expertise. LBflow provides a 'virtual laboratory' which can be used, rapidly and easily, to obtain accurate flow data for the geometrically complex, three-dimensional flows that abound in geophysical systems. Parameters can be 'steered' by the user at runtime to allow efficient and intuitive exploration of parameter space.

LBflow is written in object-oriented C++ and adopts a modular approach. Lattice Boltzmann algorithms for distinct classes of material are encoded in separate modules, which implement a standard interface, and which are linked to LBflow dynamically at runtime. This allows users with programming skill and expertise in the lattice Boltzmann method to create and share new LBflow modules, extending functionality. A companion application, LBview, provides a graphical user interface to LBflow and renders a user-configurable visualization of the output. LBflow's output can be piped directly to LBview allowing realtime visualization of steered flow. LBview also facilitates analysis of the data generated by LBflow.

This article presents an overview of the theory of the lattice Boltzmann method and describes the design and operation of LBflow. The companion paper, 'Part II', describes the practical usage of LBflow and presents detailed validation of its accuracy for a variety of flows.

1. Introduction

Over the last 20 years, a new class of computational fluid dynamic models has emerged, based upon the lattice Boltzmann equations (McNamara and Zanetti, 1988; Higuera and Jimenez, 1989; Higuera et al., 1989; Benzi et al., 1992; see Succi, 2001, for an overview). Intense research effort has been expended on the development of lattice Boltzmann algorithms with the result that a large inventory of model types is now available to the modeller. Algorithms exist for the simulation of single phase Newtonian and non-Newtonian fluids, for two-phase liquid–gas and liquid–solid mixtures and for many more-esoteric materials. Depending on the algorithm, creeping, inertial or turbulent flows may be simulated. Succi (2001) and Yu et al. (2003) provide accessible overviews.

In general, the lattice Boltzmann method (introduced in Section 2) is easier to implement than conventional computational fluid dynamic techniques, is highly amenable to parallelization and can deal with arbitrarily complex flow geometries without significant penalty. These attributes, combined with the flexibility of the lattice Boltzmann method, make it suitable for the investigation of a wide range of geophysical flow problems. The method is particularly well-suited to problems involving flow through complex geometries (e.g. permeating groundwater flow, melt segregation) and flows in which multiphase interactions are of interest (e.g. particle laden flows, bubble suspensions). Sophisticated lattice Boltzmann research-code implementations are available (e.g. PELABS: Dupuis and Chopard, 2000; and OpenLB[1]), however, since these are typically organized as libraries, their use requires programming expertise and time investment to learn the application programming interface. Commercial codes are also available.

Section 3 of this article introduces LBflow, an extensible lattice Boltzmann implementation designed with geophysical applications in mind. LBflow comprises a core executable, which controls simulation flow and lattice operations, and a suite of 'physics modules' which provide implementations of various lattice Boltzmann algorithms for different materials. The simulation parameters and required outputs are specified by the user via a plain-text simulation file, which is parsed by the core executable. This allows sophisticated numerical experiments to be conducted

☆ Code available from server at http://www.dur.ac.uk/ed.llewellin/lbflow/

E-mail address: ed.llewellin@durham.ac.uk

[1] http://www.openlb.org/

0098-3004/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2009.08.004

by those without programming experience. LBflow and its modules are written using object-oriented C++. Physics modules are called dynamically at runtime by the core executable and implement a specified interface, allowing users with programming experience to create further modules, adding functionality.

Philosophically, LBflow is designed to be used as a 'virtual laboratory' in which complex natural flows can be simulated and analysed by those without specialist computational fluid dynamics training. Several features of LBflow exemplify this approach: output from LBflow can be visualized and analysed through a companion application, LBview, which has an intuitive graphical user interface; certain key parameters can be 'steered' by the user during execution and the results can be piped dynamically through LBview, allowing interactive exploration of parameter space; LBflow is 'dimensional', i.e. parameters are specified, and results are presented, in SI units.

This article is Part I of a two-part series. Part II (Llewellin, 2009) describes the practical usage of LBflow and presents detailed validation of the code against analytical solutions and experimental data for a range of three-dimensional flows. In both parts, in line with LBflow's experimental philosophy, considerable attention is paid to the quantification of errors and to practicalities, such as obtaining a pragmatic balance between accuracy, resolution and system resources.

2. The lattice Boltzmann method

Historically, the lattice Boltzmann method developed heuristically from attempts to improve the performance of the lattice gas cellular automaton (McNamara and Zanetti, 1988; Higuera and Jimenez, 1989). Subsequent analysis has put the lattice Boltzmann method on a sound theoretical footing by showing that, providing that certain conditions are met, the lattice Boltzmann equations are formally equivalent to the Navier–Stokes equations, discretized in time and space (He and Luo, 1997; Chen and Doolen, 1998). The literature contains several excellent and rigorous theoretical treatments of the lattice Boltzmann method (see earlier references) which this section does not aim to duplicate. Rather, the relevant equations are presented and their physical meaning is discussed, echoing the method's heuristic origins.

According to the kinetic theory of gases, a control volume of gas, with spatial dimensions larger than the molecular mean free path and smaller than the hydrodynamic lengthscale of the fluid, contains a large number of molecules moving in all directions and at all velocities and exchanging momentum through collisions with one another. The probability that a molecule within this volume, centred on position $\mathbf{r}$, at time t, has velocity $\mathbf{v}$, is described by the velocity distribution function $f(\mathbf{r},\mathbf{v},t)$. Collisions between the molecules act rapidly to bring the velocity distribution function to an equilibrium described by the Maxwell–Boltzmann distribution:

$$f^{\mathrm{eq}} = \frac{\rho}{(2\pi RT)^{3/2}} \exp\left[-\frac{(\mathbf{v}-\mathbf{u})^2}{2RT}\right], \tag{1}$$

where ρ is the gas density, R the universal gas constant, T the absolute temperature of the gas and $\mathbf{u}$ is the bulk velocity of the gas packet ($\mathbf{u}=0$ for a gas at rest). For each orthogonal component α of velocity, the Maxwell–Boltzmann distribution is a normal distribution about u_α. Note that the bulk velocity of the gas must be much smaller than the root mean square velocity of the molecules. Fig. 1a and b show, schematically, the Maxwell–Boltzmann distribution of velocities for a point in a two-dimensional flow around a cylinder; note that Fig. 1b shows the distribution in velocity space.

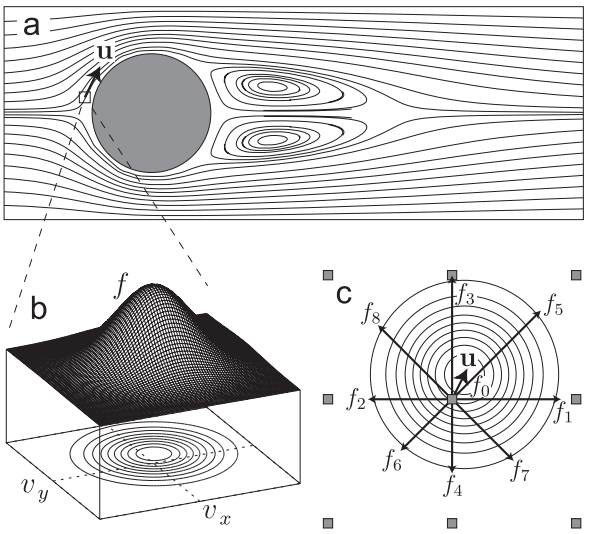

Fig. 1. Diagrammatic representation of velocity distribution function for a small packet of flowing fluid. (a) Two-dimensional flow around a cylinder; (b) velocity distribution, in velocity space, for packet of fluid with mean velocity $\mathbf{u}$; (c) same distribution discretized onto a two-dimensional, nine-velocity lattice.

The lattice Boltzmann method discretizes space into a regular lattice of nodes with spacing Δx. The lattice must have a high degree of symmetry and, typically, lower-dimension projections of a four-dimensional face-centred hypercubic lattice are used. All of the fluid molecules within a volume Δx^3 centred on a node are treated as if they exist at the node. Each node, therefore, is analogous to the control volume discussed above, and has a velocity distribution associated with it. Since time is also discretized, into timesteps of duration Δt, velocity space is also discrete. It is important to note that there are two distinct sets of units that are used in the numerical implementation of the lattice Boltzmann method: physical units and simulation units (denoted by the symbol '$\hat{}$' above a quantity). For simplicity the simulation units are usually chosen as $\Delta\hat{x}=1$ and $\Delta\hat{t}=1$. Distances in the simulation are, therefore, represented in units of lattice-spacings and times in units of timesteps. The mapping between simulation and physical units, and vice versa, is discussed later in this section.

In a three-dimensional lattice, each node has 26 immediate neighbours. If we impose the constraint that information may only propagate from a node to its nearest neighbours in a single timestep, this gives 27 'allowed' velocities (including a 'rest velocity' for stationary molecules). The velocities are denoted $\hat{\mathbf{e}}_i$ ($i=0,\ldots,n$), where n is the number of allowed velocities. On a lattice, the continuous velocity distribution $f(\mathbf{r},\mathbf{v},t)$ can be discretized into components f_i, which can be thought of as the fraction of the total mass of fluid at a node that is moving with each of the allowed velocities. Fig. 1c shows the discretization of the continuous Maxwell–Boltzmann distribution of velocities for a two-dimensional model, with nine allowed velocities.

Although each node in a three-dimensional lattice has 26 nearest neighbours, the most commonly used three-dimensional lattice Boltzmann models have only 15 or 19 allowed velocities. Fig. 2 shows the allowed velocities for a so-called D_3Q_{15} lattice (three-dimensional, 15 velocity), which is used by LBflow. The 15 velocities are:

$$\begin{array}{lll}
\hat{\mathbf{e}}_0=[0,0,0] & \hat{\mathbf{e}}_5=[0,0,1] & \hat{\mathbf{e}}_{10}=[1,-1,-1] \\
\hat{\mathbf{e}}_1=[1,0,0] & \hat{\mathbf{e}}_6=[0,0,-1] & \hat{\mathbf{e}}_{11}=[-1,-1,1] \\
\hat{\mathbf{e}}_2=[-1,0,0] & \hat{\mathbf{e}}_7=[1,1,1] & \hat{\mathbf{e}}_{12}=[1,1,-1] \\
\hat{\mathbf{e}}_3=[0,1,0] & \hat{\mathbf{e}}_8=[-1,-1,-1] & \hat{\mathbf{e}}_{13}=[1,-1,1] \\
\hat{\mathbf{e}}_4=[0,-1,0] & \hat{\mathbf{e}}_9=[-1,1,1] & \hat{\mathbf{e}}_{14}=[-1,1,-1].
\end{array} \tag{2}$$