Chapter 5

Lattice Boltzmann Method for Sparse Geometries: Theory and Implementation

Tadeusz Tomczak
Wrocław University of Science and Technology, Poland

ABSTRACT

This chapter presents the challenges and techniques in the efficient LBM implementations for sparse geometries. The first part contains a review of applications requiring support for sparse geometries including industry, geology, and life sciences. For each category, a short description of a geometry characteristic and the typical LBM extensions are provided. The second part describes implementations for single-core and parallel computers. Four main methods allowing for the reduction of memory usage and computational complexity are presented: hierarchical grids, tiles, and two techniques based on the indirect addressing (the fluid index array and the connectivity matrix). For parallel implementations, the advantages and disadvantages of the different methods of domain decomposition and workload balancing are discussed.

INTRODUCTION

One of the features of many LBM implementations is the use of uniform grids for the computational domain. This, connected with the spatial and temporal locality of the Boltzmann transport equation, allows for the use of simple data structures (multidimensional matrices), an automated mesh preparation even for complex geometries, and for trivial parallelization of computations. Unfortunately, the uniform
grid requires special techniques to correctly and efficiently handle geometries with borders of complex shape or with many areas without fluid (sparse geometries), especially on parallel machines. Since many real geometries belong to both the above-mentioned classes, these issues are especially important considering that even current desktop systems are based on processors performing computations in a parallel, single-instruction-multiple-data (SIMD) way.

Two main issues must be treated to efficiently simulate sparse geometries: a proper boundary conditions treatment for complex shapes and an efficiency of implementation (memory usage, performance and scalability on multiprocessor machines). This work concerns primarily the later issue, especially in the context of the geometry impact. In fact, almost all LBM implementations specialised for sparse, complex geometries focus primarily on decreasing the performance penalties and an additional memory usage caused by storing and processing of solid nodes.

One of the main geometry parameters affecting performance is a geometry porosity defined as

\[ \varphi = \frac{\text{fluid volume}}{\text{total volume}}. \] (1)

In practice, due to the space discretization, the porosity is computed as a ratio of the number of fluid LBM nodes to the total number of nodes. In addition to porosity, also its complementary coefficient, solidity, equal to

\[ s = \frac{\text{solid volume}}{\text{total volume}}, \] (2)

may be useful. Notice, that both \( \varphi, s \in (0,1) \) and \( \varphi + s = 1 \).

The range of porosity values of real sparse geometries can vary greatly (see Fig. 1). The porosity of engineering materials is usually below 70% and in many cases can be much lower; for example, the typical porosity of hydrocarbon-bearing sandstones is between 5% and 35% (O’Connor & Fredrich, 1999). In medical geometries (vascular systems) the porosity may be even as low as 0.3% (Godenschwager, Schornbaum, Bauer, Köstler, & Rüde, 2013).

Besides porosity, many other parameters can be found for a sparse geometry including permeability, dispersion, tortuosity, percolation and surface/volume ratio to name a few. Nevertheless, their impact on implementation performance or memory usage is not known yet.

The geometry porosity may be used to decide whether an implementation specialised for sparse geometries should give higher performance, but it may be
Code Clone Detection and Analysis in Open Source Applications
www.igi-global.com/chapter/code-clone-detection-and-analysis-in-open-source-applications/192915?camid=4v1a