`pylbm`: A flexible Python package for lattice Boltzmann method

Loïc Gouarin[1] and Benjamin Graille[1]

[1]CMAP/CNRS, école polytechnique
[2]Université Paris-Saclay, CNRS, Laboratoire de mathématiques d'Orsay, 91405, Orsay, France

June 16, 2020

1 Summary

`pylbm` is an open-source package written in Python [1]. It proposes a leading-edge implementation of the lattice Boltzmann method for 1D/2D/3D problems and has been created to address various communities:

- **Mathematicians**: `pylbm` provides a very pleasant and efficient environment in order to test new schemes, understand their mathematical properties, such as stability and consistency and open towards new frontiers in the field such as mesh refinement, equivalent equations…
- **Physicists**: `pylbm` offers an ideal framework to conduct numerical experiments for various levels of modeling and schemes, without having to dive into the mathematical machinery of LBM schemes. The high level of the proposed algorithm can also allow to go beyond the first test and easily conduct large scales simulations thanks to its parallel capability.
- **Computer scientists**: In `pylbm`, the lattice Boltzmann method is not hard-coded. Advanced tools of code generation, based on a large set of newly developed computer algebra libraries, allow a high-level entry by the user of scheme definition and boundary conditions. The `pylbm` software then generates the resulting numerical code. It is therefore possible to modify the code building kernels to test performance and code optimization on different architectures (AA pattern and pull algorithm); the code can also be generated in different languages (C, C++, openCL, …).

The principle feature of `pylbm` is its great flexibility to build lattice Boltzmann schemes and generate efficient numerical simulations. Moreover, it has excellent parallel capabilities and uses MPI for distributed computing and openCL for GPUs.

The generalized multiple-relaxation-time framework is used to describe the schemes [2]. It's then easy to define your lattice Boltzmann scheme by providing the velocities, the moments, their equilibrium values, and the associated relaxation parameters. Moreover, multiple D_dQ_q schemes can be coupled in the simulation where d is the dimension and q the number of velocities. This formalism is used for example to simulate thermodynamic fluid flows as in the Rayleigh-Benard test case. But you can also experiment with new types of lattice Boltzmann schemes like vectorial schemes [3] or with relative velocities [4].

`pylbm` will offer in the future releases more tools to help the user to design their lattice Boltzmann schemes and make large simulations with complex geometries.

- **More examples**: we want to give access to various lattice Boltzmann schemes you can find in the literature. We will add multi-component flows, multi-phase flows, ... in order to have a full gallery of what we can do with LBM. We hope this way the users can improve this list with their own schemes.
- **Equivalent equations**: the hard part with the LBM is that you never write the physical equations you want to solve but the lattice Boltzmann scheme associated. We will offer the possibility to retrieve the physical equations from the given scheme by doing a Chapman-Enskog expansion for nonlinear equations up to the second order.
- **Complex geometries**: the geometry in `pylbm` can be described by a union of simple geometry elements like circle, triangle, sphere,... It's not realistic for industrial challenges and we will offer the possibility to use various CAD formats like STL.

In the following sections, we describe the main features of the `pylbm` module: the formal description of the scheme, the tools of analysis (to compute the equivalent equations and to visualize the stability), the possibility to implement your own boundary conditions.

2 A formal description of the lattice Boltzmann scheme

The greatest asset of `pylbm` is that the scheme is not hard-coded: it is described by a dictionary that contains all necessary information of the simulation. Moreover, schemes with multiple distribution functions are simply described with a list of elementary schemes that can be coupled by the equilibrium functions. Each elementary scheme is written in the MRT formalism proposed by d'Humière [2].

2.1 MRT formalism

Let us first consider a regular lattice L in dimension d with a mesh size dx and a time step dt. The scheme velocity λ is then defined by $\lambda = dx/dt$. We introduce a set of q velocities adapted to this lattice $\{v_0, \dots, v_{q-1}\}$, that is to say that, if x is a point of the lattice L, the point $x + v_j dt$ is also on the lattice for every $j \in \{0, \dots, q-1\}$.

The aim of the *DdQq* scheme is to compute a distribution function vector $f = (f_0, \dots, f_{q-1})$ on the lattice L at discrete values of time. The scheme splits into two steps: the relaxation phase and the transport phase. That is, the passage from the time t to the time $t + dt$ consists in the succession of these two phases.

the relaxation phase

This phase, also called collision, is local in space: on every site x of the lattice, the values of the vector f are modified, the result after the collision being denoted by $f^\star$. The collision operator is a linear operator of relaxation toward an equilibrium value denoted f^{eq}.

This linear operator is diagonal in a peculiar basis called moments denoted by $m = (m_0, \dots, m_{q-1})$. The change-of-basis matrix M is such that $m = Mf$ and $f = M^{-1}m$. In the basis of the moments, the collision operator then just reads

$$m_k^\star = m_k - s_k(m_k - m_k^{\text{eq}}), \qquad 0 \leqslant k < q,$$

where s_k is the relaxation parameter associated to the k-th moment. The k-th moment is said conserved during the collision if the associated relaxation parameter s_k is zero. We introduce the diagonal matrix $S = \text{diag}(s_0, \dots, s_{q-1})$ and the collision operator reads

$$m^\star = m - S(m - m^{\text{eq}}).$$

In this previous formula, the equilibrium value of the k-th moment, m_k^{eq}, $0 \leq k < q$, is a given function of the conserved moments.

By analogy with the kinetic theory, the change-of-basis matrix M is defined by a set of polynomials with d variables $(P_0, \dots, P_{q-1})$ by

$$M_{ij} = P_i(v_j).$$

The relaxation phase consists then to compute on every site x of the lattice the new particle distribution functions $f^\star$:

- compute the vector of the moments $m = Mf$,
- compute the moments after the collision $m^\star = m - S(m - m^{\text{eq}})$,
- compute the vector of the particle distribution functions $f^\star = M^{-1}m^\star$.

the transport phase

This phase just consists in a shift of the indices and reads

$$f_j(x, t + dt) = f_j^\star(x - v_j dt, t), \qquad 0 \leqslant j < q.$$

2.2 Build your first scheme

A lattice Boltzmann scheme is described in `pylbm` by a dictionary that contains all the necessary information. For example, the following dictionary can be used to build a scheme to simulate the Burgers equation $\partial_t U + \partial_x U^2/2 = 0$:

```
scheme_cfg = {
    'dim': 1,
    'scheme_velocity': 1.,
    'schemes': [
        {
            'velocities': [1, 2],
            'conserved_moments': U,
            'polynomials': [1, X],
            'equilibrium': [U, U**2/2],
            'relaxation_parameters': [0, 1.5],
        },
    ],
}
```

Each elementary scheme is then given by the **stencil** of velocities, the **conserved moments**, the **polynomials** that define the moments, the **relaxation parameters**, the **equilibrium values** that are functions of the conserved moments.

At our knowledge, `pylbm` is then the only lattice Boltzmann tool for which the user can implement is own scheme without to code inside the module. All the description of the simulation is given in a script file.

2.3 Symbolic variables

2.4 relative velocities

References

[1] L. Gouarin and B. Graille, *pylbm*. 2018.

[2] D. d'Humière, "Generalized lattice-boltzmann equations," in *Rarefied gas dynamics: Theory and simulation*, 1992, vol. 159, pp. 450–458.

[3] B. Graille, "Approximation of mono-dimensional hyperbolic systems: A lattice boltzmann scheme as a relaxation method," *Journal of Computational Physics*, vol. 266, pp. 74–88, 2014.

[4] F. Dubois, T. Février, and B. Graille, "Lattice boltzmann schemes with relative velocities," *Communications in Computational Physics*, vol. 17, no. 4, pp. 1088–1112, 2015.