Thermal Lattice Boltzmann Method for a three populations plasma

Lorenzo Bonelli[1] and Federica Ravasio[2]

[1]Department of Nuclear engineering
[2]Department of Physics engineering

Contents

1 Introduction

The Lattice Boltzmann Method (LBM) is a numerical approach that aims to simulate a liquid by the use of the kinetic equation, also called the Boltzmann equation. Instead of solving the Navier-Stokes equations, as it is done in Computational Fluid Dynamics, the LBM aims to discretize the kinetic equation into a set of Lattice Boltzmann Equations (LBE) that are solved on a lattice grid. The use of LBM nowadays is very broad: it can be used, for example, in biomechanics or in the study of nanofluids. If coupled with a thermal equation, the so-called Thermal Lattice Boltzmann Method, it can be even used to describe heat transfer and thermo-physics problems.
When multiple populations are used inside the LBM, the use can be even wider, and it can also be used to describe Magnetofluid-like in fusion plasma physics, magnetic molten metal flows, and ion thruster optics.
The aim of this work is to study the general description of LBM and to approximate the problem, limiting it to our case. Then we want to realize a full code in C++ that embodies a three populations TLBM with different possible uses. In the end the full aim will be the optimization of the code in order to make it as fast and as memory efficient as possible.

2 Physical problem

The description has to start from the study of a multiple population plasma in an Electro-Magnetic field. The most general treatment that can be done is the "ab initio" one that describes every particle with its velocity and position and the EM field with the usual Maxwell equations:

$$\begin{aligned}
\frac{d\vec{v}_{i,a}}{dt} &= \frac{q_a}{m_a}(\vec{E} + \frac{\vec{v}_{i,a}}{c} \times \vec{B}) \\
\frac{d\vec{x}_{i,a}}{dt} &= \vec{v}_{i,a} \\
\vec{\nabla} \cdot \vec{E} &= 4\pi(\rho_{ext} + \sum_a \rho_a) \\
\vec{\nabla} \times \vec{B} &= \frac{4\pi}{c}(\vec{J}_{ext} + \sum_a \vec{J}_a) + \frac{1}{c}\frac{\partial \vec{E}}{\partial t} \\
\vec{\nabla} \cdot \vec{B} &= 0 \\
\vec{\nabla} \times \vec{E} &= -\frac{1}{c}\frac{\partial \vec{B}}{\partial t}
\end{aligned} \tag{1}$$

Them are reported here in Gaussian unit with the subscript a indicating the species and i the particle.

2.1 Kinetic equation

In order to retrieve the kinetic description, we associate a microscopic distribution function defined as:

$$f_a(\vec{x}, \vec{v}, t) = \sum_i \delta(\vec{x} - \vec{x}_{i,a})\delta(\vec{v} - \vec{v}_{i,a}) \tag{2}$$

that represents a point in the phase space (with coordinates of position and velocity) at a specific time. Here a and i indicate, respectively, the species and the different particle of the same species. From this definition we can get the Boltzmann equation that we need for the model:

$$\frac{\partial f_a}{\partial t} = \vec{v} \cdot \frac{\partial f_a}{\partial \vec{x}} + \frac{q_a}{m_a}(\vec{E} + \frac{\vec{v}}{c} \times \vec{B}) \cdot \frac{\partial f_a}{\partial \vec{v}} = -\frac{q_a}{m_a}(< \widetilde{\vec{E}} \cdot \frac{\partial \widetilde{f_a}}{\partial \vec{v}} > + \frac{\vec{v}}{c} \times < \widetilde{\vec{B}} \cdot \frac{\partial \widetilde{f_a}}{\partial \vec{v}} >) \tag{3}$$

The equation turns out to be rather complicated because of the tilde-terms indicating the microscopic oscillation of a macroscopic quantity for whom the average value in the phase space is zero. However, the right-hand side of the equation can be described as a collision term to have a much simpler form:

$$\frac{\partial f_a}{\partial t} = \vec{v} \cdot \frac{\partial f_a}{\partial \vec{x}} + \frac{q_a}{m_a}(\vec{E} + \frac{\vec{v}}{c} \times \vec{B}) \cdot \frac{\partial f_a}{\partial \vec{v}} = C_a \tag{4}$$

Here C_a indicates the collision term of the specie a. Its description and the approximations exploited are described and explained in section 2.3

In order to retrieve a self-consistent description, this equation has to be coupled with the Maxwell equation. The Maxwell-Boltzmann equations are the following:

$$\begin{aligned}
\frac{\partial f_a}{\partial t} = \vec{v} \cdot \frac{\partial f_a}{\partial \vec{x}} + \frac{q_a}{m_a}(\vec{E} + \frac{\vec{v}}{c} \times \vec{B}) \cdot \frac{\partial f_a}{\partial \vec{v}} &= C_a \\
\vec{\nabla} \cdot \vec{E} &= 4\pi(\rho_{ext} + \sum_a \rho_a) \\
\rho_a &= q_a * \int f_a d^3v \\
\vec{\nabla} \times \vec{B} = \frac{4\pi}{c}(\vec{J}_{ext} + \sum_a \vec{J}_a) + \frac{1}{c}\frac{\partial \vec{E}}{\partial t} & \\
\vec{J}_a &= q_a \int \vec{v} f_a d^3v \\
\vec{\nabla} \cdot \vec{B} &= 0 \\
\vec{\nabla} \times \vec{E} &= -\frac{1}{c}\frac{\partial \vec{B}}{\partial t}
\end{aligned} \tag{5}$$

With this set of equations the description is self-consistent if we choose an appropriate value or model for the collision.

2.2 Non relativistic limit

In our treatment, however, one of the main assumptions is that we are dealing with particles moving far from relativistic velocities (so we are assuming kinetic energy << rest mass energy) and that we

have no external magnetic field. One of the main outcome is that, in this approximation, the term $\frac{v}{c}$ is nearly zero and the Maxwell equations can be reduced to the simple Poisson equation. The new set of equations that need to be solved (written in SI system) is:

$$\begin{aligned} \frac{\partial f_a}{\partial t} + \vec{v} \cdot \vec{\nabla} f_a + \frac{q_a}{m_a} \vec{E} \cdot \vec{\nabla}_v f_a = C_a \\ \nabla^2 \phi = -\frac{\rho}{\epsilon_0} \end{aligned} \tag{6}$$

Where ϕ is the electric potential defined as usual as $\vec{\nabla}\phi = -\vec{E}$

2.3 Collisions

In order to treat the collision term we have to make some considerations and approximations. If we deal with different populations we have to consider a collision term associated to each one of them: $C_a = \sum_b C_{ab}$. Here the sum over b is conducted over all the population, a included. The terms C_{ab} will indicate inter-species collisions, while the terms C_a are associated with intra-species ones. Different models has been studied to treat this term we here report some of them:

- **Vlasov**: for a sufficiently hot, diluted and ionized plasma we can embed a collisionless description. By neglecting the collision term ($C_a = 0$) we get a much more simple equation that takes the name of Vlasov equation.

- **BGK**: The Buttnagar-Gross-Kruck approximation (or relaxation time approximation) is the most vastly used inside the LBM and it considers the collision term as the difference between the distribution function and the equilibrium one: $C_a = -\frac{f_a - f_a^{eq}}{\tau_a}$ [1] where τ_a is the specific relaxation time of the species and f_a^{eq} is the equilibrium distribution function that comes from the model used to describe the particle.
  In our limit there is no need to treat the Fermi-Dirac or Bose-Einstein distributions; it is enough to treat it as a Maxwell-Boltzmann distribution.
  We have
  $$f_a^{eq} = \frac{n_a}{(2\pi RT)^{D/2}} \exp -\frac{(\vec{v} - \vec{u}_a)^2}{2RT}$$
  [2] where D is the space dimension, R is the universal gas constant, T is the temperature, $\vec{v}$ is the velocity in the phase space, n_a is the mass density associated to the specie a (calculated in $[\frac{kg}{m^3}]$) and $\vec{u}_a$ is the macroscopic velocity associated to the specie a. In order to treat inter-specie collision we can refer to an inter-specie equilibrium distribution function
  $$f_{ab}^{eq} = \frac{n_a}{(2\pi RT)^{D/2}} \exp -\frac{(\vec{v} - \vec{u}_{ab})^2}{2RT}$$
  [2] where
  $$\vec{u}_{ab} = \frac{n_a \vec{u}_a + n_b \vec{u_b}}{n_a + n_b}$$
  is the barycentric velocity of binary collisions. The collision term will be $C_{ab} = -\frac{f_a - f_{ab}}{\tau_{ab}}$ where

τ_{ab} is the relaxation time of binary collisions. Of course we can retrieve $C_a = C_{aa}$ so we can use the most general case and get also single specie collisions

- There are also higher order collisions terms that can be used to describe the phenomena like the Landau collision operator or the Balescu-Lenard collision operator. However their treatment is much more complicated and it goes beyond the scope of this study.

3 Discretization

In order to treat the LBE, we have to discretize the velocities in the phase space. Depending on the number of dimensions and directions that we want to use, we can use different approaches that fall all under the name $DnQm$ where n stands for number of dimensions and m stands for number of directions. The most used are, in 2D, **D2Q5** or **D2Q9**, while, in 3D, **D3Q15** or **D3Q27**. In figure 1 we can see the different directions adopted for the **D2Q9** model. A general expression of the directions can be obtained as in [3]:

$$\vec{e}_\alpha = \begin{cases} 0 & \alpha = 0 \\ \sqrt{3}(cos((\alpha-1)\frac{\pi}{2}), sin((\alpha-1)\frac{\pi}{2}))c_s & \alpha = 1,2,3,4 \\ \sqrt{6}(cos((\alpha-5)\frac{\pi}{2}+\frac{\pi}{4}), sin((\alpha-5)\frac{\pi}{2}+\frac{\pi}{4}))c_s & \alpha = 5,6,7,8 \end{cases} \tag{7}$$

where c_s is the lattice sound speed. However, for the D2Q9 case, c_s is constant and equal to $c_s = \frac{c}{\sqrt{3}}$ with c "light speed" of the system that can be chosen since $c = \frac{\delta_x}{\delta_t}$ and often $\delta_x = \delta_t = 1$. This means that the vectors are simply:

$$\vec{e}_\alpha = (0,0); (1,0); (0,1); (-1,0); (0,-1); (1,1); (-1,1); (1,-1), (-1,-1) \tag{8}$$

as we can see in figure 1.

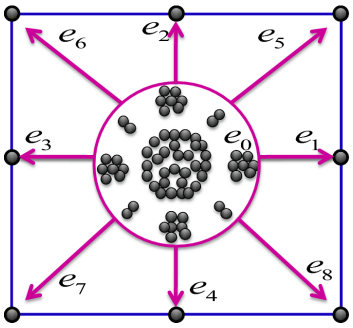


Figure 1: Directions in the D2Q9 model

Associated to the velocity, each model has a weight that indicates the probability for that velocity.

In the D2Q9 case the weights are:

$$w_\alpha = \begin{cases} \frac{4}{9} & \alpha = 0 \\ \frac{1}{9} & \alpha = 1,2,3,4 \\ \frac{1}{36} & \alpha = 5,6,7,8 \end{cases} \tag{9}$$

3.1 Discretized Boltzmann Equation

From the discretization of the directions (and so also velocities) present inside the Boltzmann equation, we report the complete discretization of the Boltzmann equation as in [4]:

$$f_{\alpha,a}(\vec{x} + \vec{e}_\alpha \delta_t, t + \delta_t) = f_{\alpha,a}(\vec{x}, t) + C_{\alpha,a} + \delta_t F_{\alpha,a} \tag{10}$$

Where as before α indicates the direction while a indicates the specie. From this equation we now need to describe the collision term and the force discretized. If we use the BGK model we need the "discretized" $f^{eq}_{\alpha,a}$ that can be found in several papers [5] deriving from the f^{eq} expansion and the Chapman-Enskog approximation, only valid for low Mach numers ($M_a = \frac{u}{c_s}$):

$$f^{eq}_{\alpha,a} = w_\alpha n_a (1 + \frac{\vec{e}_\alpha \cdot \vec{u}_a}{c_s^2} + \frac{(\vec{e}_\alpha \cdot \vec{u}_a)^2}{2c_s^4} - \frac{\vec{u}_a^2}{2c_s^2}) \tag{11}$$

3.2 Force discretization

In the expression of the discretized LBE the only missing term is the Force one. In order to treat this, various authors have proposed different methods:

- **Guo** [6]: this is the most embedded in LB codes. The idea is to develop in series the force term, with respect to the particle direction, and truncate the series.

  $$F_{\alpha,a} = w_\alpha * n_a * (1 - \frac{1}{2\tau_a})(\frac{\vec{e}_\alpha - \vec{u}_a}{c_s^2} + \frac{\vec{e}_\alpha \cdot \vec{u}_a}{c_s^4}\vec{e}_\alpha) \cdot \vec{a}_a \tag{12}$$

  Here $\vec{a}_a$ is the acceleration due to external force. The latter is, in general, provided by the electric field (external and generated by the mutual interaction of particles).

- **He** [7]: this model is based on an approximation made in the continuum:

  $$\vec{\nabla}_{\vec{v}} f \approx \vec{\nabla}_{\vec{v}} f^{eq} = -\frac{\vec{v} - \vec{u}}{RT} f^{eq} \tag{13}$$

  If this approximation holds we can substitute in the continuum equation the result above and the discretize it. In this way we obtain a simpler expression that uses also the equilibrium distribution function as follows:

  $$F_{\alpha,a} = \frac{\vec{a}_a \cdot (\vec{e}_\alpha - \vec{u}_a)}{c_s^2} f^{eq}_{\alpha,a} \tag{14}$$

In our code we exploited the Guo description.

3.3 Poisson equation

In order to describe the acceleration, as we said, we need the Electric field. This can be described by $\vec{E} = \vec{E}_{ext} + \vec{E}_{int}$ where $\vec{E}_{ext}$ comes from the electric field imposed from external source (for example a capacitor), while $\vec{E}_{int}$ comes from the resolution of the Poisson equation. The Poisson equation is an elliptic partial differential equation that features a Laplace operator: $\vec{\nabla}^2\phi = -\rho$. In order to solve it we need a numerical procedure. We can feature different type of numerical procedure in order to obtain the result, here are the ones we implemented:

- **NONE**: Not solving the Poisson equation leads to a "fluid" driven only by an external force and, eventually, collisions. This is a good approximation for weakly ionized plasma, but it does not treat the most general case.
- **GAUSS-SEIDEL (GS)**: In this method we divide the domain in a regular grid and apply a 5-point difference stencil getting

$$\frac{\phi(x+1,y) + \phi(x-1,y) + \phi(x,y+1) + \phi(x,y-1) - 4\phi(x,y)}{(\delta_x)^2} = -\rho \tag{15}$$

  Where we assumed $\delta_x = 1$ as before. The GS interaction update the potential in-place as

$$\phi_{new} = \frac{1}{4}(\phi(x+1,y) + \phi(x-1,y) + \phi(x,y+1) + \phi(x,y-1) + \delta_x^2\rho) \tag{16}$$

  Then we repeat until convergence of $|\phi_{new} - \phi_{old}|$ up to a chosen small error value
- **SOR**: The Successive Over-Relaxation method is a refinement of the GS scheme, in which a relaxation parameter ω is introduced to obtain a faster convergence. By choosing a relaxation parameter ω such that $1 < \omega < 2$ we get a new update rule:

$$\phi_{new} = (1-\omega)\phi_{old} + \frac{\omega}{4}(\phi(x+1,y) + \phi(x-1,y) + \phi(x,y+1) + \phi(x,y-1) + \delta_x^2\rho) \tag{17}$$

  As in GS, the iteration is performed until the error falls below the chosen threshold.
- **FFT**: This method is based on a spectral approach relying on the Fast Fourier Transform (FFT) [8]. Indeed, in Fourier space, the Poisson equation takes a much simpler algebraic form:

$$\hat{\Phi}(k_x, k_y) = \frac{\hat{\rho_q}(k_x, k_y)}{k^2}$$

  Where the discrete Laplacian is now given by:

$$k^2 = 4\sin^2\left(\frac{\pi k_x}{N_x}\right) + 4\sin^2\left(\frac{\pi k_y}{N_y}\right)$$

  Since the $k = 0$ component (the constant mode) represents the total net charge, to avoid divergences we impose $\hat{\Phi}(0,0) = 0$ making sure that $< \rho_q >= 0$, that is, indeed, the case

for a neutral plasma as we can generally assume. Once we obtain $\hat{\Phi}(k_x, k_y)$ we perform the inverse transform (iFFT) to obtain the potential back to the real space. This method is the fastest among the ones presented in this work, however, it should be noted that it works only if periodic boundary conditions are applied, greatly limiting its usability.

In our code we also implemented a GS based methods but with a 9-points stencil. This will indeed be useful later on to evaluate the evolution of the species parameters in 9 different grid points.
After having retrieved the potential with one of the previous methods we have to calculate the electric field $E = -\nabla\phi$. This can be easily done by using finite difference for the gradient so:

$$\begin{cases} E_x^{int} = -\frac{\phi(x+1,y)-\phi(x-1,y)}{2\delta_x} \\ E_y^{int} = -\frac{\phi(x,y+1)-\phi(x,y-1)}{2\delta_x} \end{cases} \tag{18}$$

In the end we can apply boundary conditions to the internal electric field evaluation and then get the total electric field just adding it to the external one: $\vec{E} = \vec{E}_{ext} + \vec{E}_{int}$.
In our implementation the external electric field it's considered as an impulsive field. This means that after the first iteration we're considering just the internal contribution.

3.4 Lattice units and Macroscopic quantities

It is hidden in the discretization step an important consideration regarding the units choice. In order to be consistent, every unit has to be intended as lattice unit. It is possible to retrieve from the lattice units quantity the physical quantity and do the other way around by imposing physical properties and then get the lattice units from them.
To switch from SI units to Lattice units, we defined a set of reference quantities for the fundamental physical dimensions: mass, length, time, charge and temperature. Specifically we selected plasma-characteristic values: respectively we choose the electron mass m_e, the Debye length λ_D, $\frac{\sqrt{3}}{\omega_p}$ (where ω_p is the plasma frequency), the elementary charge e and the initial electron temperature T_e^{init}. The choice of $\sqrt{3}$ in the time reference ensures that the sound speed c_s, which is typically defined as $c_s = \frac{v_{th}}{v_0}$ where v_{th} is the thermal velocity and v_0 a characteristic speed, results in $c_s = \frac{1}{\sqrt{3}}$, consistently with the assumption imposed by the Lattice Boltzmann model. The new units in lattice units will be obtained as: $F^{LU} = \frac{F^{SI}}{F^0}$ where F is a generic physical quantity and F^0 the reference one.
As an example, we can consider the electric field; in SI it is expressed as $E[\frac{N}{C}] = E[\frac{kg \cdot m}{s^2 \cdot C}]$, so to switch to lattice units we define

$$\tilde{E_0} = \frac{\tilde{M_0} \cdot \tilde{L_0}}{\tilde{t_0^2} \cdot \tilde{Q_0}}$$

and $E_{LU} = \frac{E_{SI}}{\tilde{E_0}}$.
In this context it is important to underline that the distribution function is not a macroscopic quantity but rather a microscopic one that can be associated to macroscopic quantity with specific

relations [6]:

$$\begin{aligned} n_a &= \sum_\alpha f_{\alpha,a} \\ n_a \vec{u}_a &= \sum_\alpha \vec{e}_\alpha f_{\alpha,a} + \frac{\delta_t \vec{F}_a}{2} \end{aligned} \tag{19}$$

3.5 Boundary conditions

In order to be able to perform a simulation we need to define the boundary conditions that the distribution function must satisfy. In order to have a general approach, we implemented different methods:

- **PERIODIC**: with the periodic boundary condition we assume that we are treating a small volume element of a bigger structure and that every cell is completely identical to the other. If that is the case the flow of particles from one side reenters from the other side with the same direction.
- **BOUNCE_BACK**: with this boundary condition we ensure that no particle can escape the site and that when it reaches the border it comes back with opposite direction.

4 Thermal equation

One of the most interesting methods to treat the temperature aspects in LB simulations is the so called Double Distribution Function (DDF) approach. Here the thermal field is described by a separate distribution function g that evolves according to its own LBE. In a simplified model (with respect to the total energy of the system considered in a more complete formulation presented by [9]) the thermal dynamics can be approximated by focusing only on the internal energy component. The evolution is thus governed by a LBE that includes a source term representing the dissipation effect. In a physical point of view this would represents a sort of energy conservation; indeed the loss of energy due to the collisions in the f distribution is now reintroduced as heat in the g distribution. Formally, according to [10], the two coupled LBE systems can be written, using our usual notation, as:

$$f_{\alpha,a}(\vec{x} + \vec{e}_\alpha \delta_t, t + \delta_t) = f_{\alpha,a}(\vec{x}, t) + C_{\alpha,a} + \delta_t F_{\alpha,a}$$

$$g_{\alpha,a}(\vec{x} + \vec{e}_\alpha \delta_t, t + \delta_t) = g_{\alpha,a}(\vec{x}, t) + C^g_{\alpha,a} + \delta_t S^g_{\alpha,a}$$

where $S^g_{\alpha,a}$ is the source term that represents the conversion of kinetic energy dissipated by f into internal thermal energy. This term is modeled as:

$$S^g_{\alpha,a} = F_{\alpha,a} + \frac{1}{2} dt \frac{\delta F_{\alpha,a}}{\delta t}$$

with

$$\frac{\delta F_{\alpha,a}(\vec{x}, t)}{\delta t} = \frac{F_{\alpha,a}(\vec{x}, t) - F_{\alpha,a}(\vec{x}, t - dt)}{dt}$$

$$F_{\alpha,a} = \omega_\alpha \frac{\varphi_{\alpha,a}}{c_v}[1 + \frac{\vec{e_\alpha} \cdot \vec{u}}{c_s^2}\frac{\tau_g - \frac{1}{2}}{\tau_g}]$$

where c_v is the specific heat at constant volume and $\varphi_{\alpha,a}$ is the viscous heating term given by:

$$\varphi_{\alpha,a} = -\frac{2\mu}{3}(\nabla \cdot \vec{u})^2 + 2\mu S_{\alpha,a}^2$$

with $S_{\alpha,a}^2 = S_{1,2}^2 + S_{1,3}^2 + S_{2,3}^2$ and $S_{a,b} = \sum_\alpha e_{\alpha,a} e_{\alpha,b}(f_\alpha - f_\alpha^{eq})$ where $1, 2, 3$ are the a indices of the species.
The macroscopic temperature then can be evaluated in this scheme as $T = \frac{1}{n}\sum_i g_i$.
The treatment provided corresponds to the complete physical derivation. This could be implemented but it would involve even more computational effort than we have so far. In order to handle a more simple system, however, we can make further approximations considering the system's physics. In fact, as mentioned earlier, what we want to observe is just a transformation of the kinetic energy, lost during collisions, into thermal energy. This can be evaluated, after some calculations and approximations, by retrieving the kinetic energy lost by the f collisions as:

$$\Delta E_{\alpha,a} = \rho_a(u_{x,a}^2 + u_{y,a}^2)\sum_b \frac{2(1 - \frac{1}{\tau_{ab}})^2 - 2\rho_a(1 - \frac{1}{\tau_{ab}}) - Q\frac{f_{\alpha,ab}^{eq}}{\tau_{ab}}}{2(2(1 - \frac{1}{\tau_{ab}}) + Q\frac{f_{\alpha,ab}^{eq}}{\tau_{ab}})} \tag{20}$$

where Q stands for the number of directions considered in the model; 9 in our case. Then we insert it as a source term in the g distribution $\Delta T_{\alpha,a} = -\frac{\Delta E_{\alpha,a}}{K_b}$ with K_b the Boltzmann constant expressed in Lattice units. The final equation would simply be:

$$g_{\alpha,a}(\vec{x} + \vec{e}_\alpha\delta_t, t + \delta_t) = g_{\alpha,a}(\vec{x}, t) + C_{\alpha,a}^g + \ \Delta T_{\alpha,a} \tag{21}$$

Here we consider

$$T = \sum_i g_i \tag{22}$$

The relaxation term described by $C_{\alpha,a}^g$ is treated as the correspondent one in the f distribution, considering equal relaxation times τ_{ab}. To evaluate them one should consider many physical parameters including the cross section. However we can tune them simply considering that the higher the density (and the radius) of a specie the more probable is a collision event. This lead to a faster diffusion and so a smaller relaxation time. In this approximation so we can impose equal τ_{ab} for both the f and the g distribution.

5 Code structure

The code is organized in a modular way that follows a logical order. The aim is to write a code that is easier to read, maintain and update with respect to an unique block code. This will also be suitable for future implementation.
The functionalities are divided into some principal modules:

- **plasma.hpp**: This file contains the principal methods and all the parameters useful for the initialization and the principal structure of the code. What's fundamental is that in the header file we have the definition of the class LBmethod that is largely used.

- **plasma.cpp**: Here instead we have some fundamental methods:

    1. Initialize: In this step the aim is to define in the distribution function in the initial condition: $f_{\alpha,a} = w_\alpha n_a$ and $g_{\alpha,a} = w_\alpha T_a$ where w_α are defined in the eq. 9. In order to observe an expansion in the charged particle distributions, we defined them, at first, only in a central region of the whole space.
    2. ComputeEquilibrium: Here we calculate the equilibrium distribution functions that are useful for further calculation following the eq. 11.
    3. UpdateMacro: In this step we update, from the distribution function, the value of the macroscopic quantities from the eqs. 19 and 22.
    4. Run_simulation: This method iterates the simulation steps for how many time steps the user wants to perform.

- **poisson.cpp**: Here we have necessary to use one of the schemes proposed in the section 3.3 and compute the internal electric field generated from the charge distribution. In this file we use a dispatcher to chose between solver methods. The choice, however, can be easily done by the user in the main file.

- **collision.cpp**: It implements the relaxation step after having selected the BGK collision model reported in section 2.3. Here there's also the method used to evaluate the thermal LBE as eq. 21.

- **streaming.cpp**: It contains the functions responsible for the streaming part for both f and g distribution functions. Depending on our physical problem we can select one of the streaming procedure in section 3.5 in the main file and apply it to our code.

- **visualization.cpp**: It implements all the necessary for the visualization of the results. The videos used to evaluate the temporal evolution are created exploiting the C-native OpenCV library. This choice was made due to its faster performances with respect to some other python implementations.

- **main_plasma.cpp**: The main file is the one used by the user to define the controllable quantities such as the grid size or the time steps. Here we run the simulation calling the LBmethod class method "Run_simulation".

- **compile_and_run.sh**: This is the file that compiles the code and executes the simulation.

Each module includes only the necessary and the interface between modules is managed through headers and forward declaration.
The simulation is based on a temporal loop structure, in which, at each step, the fundamental operations of the LBM are performed in sequence:

1. UpdateMacro
2. ComputeEquilibrium
3. Collisions
4. Streaming
5. SolvePoisson

All these functions are managed by the class LBmethod and by modular external functions.
Some relevant architectural choices we want to highlight are:

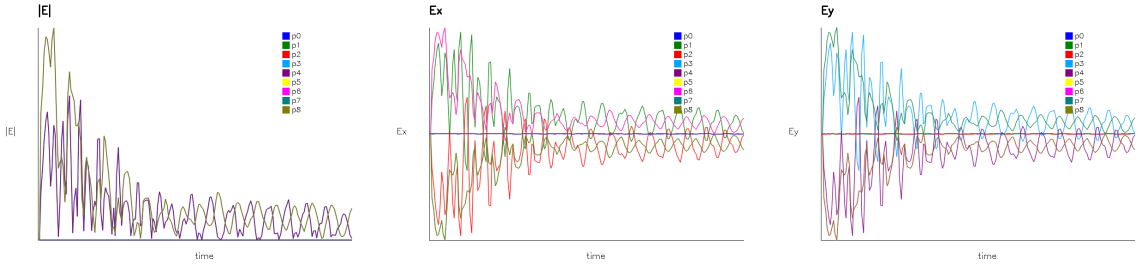
- Unique LBmethod class: it contains the entire simulation state, granting a centralized control and an efficient memory access. The f and g distribution are contained in 1D vectors in order to have good performances on CPU and a readable code.
- The parallelization is performed by OpenMP in order to have fast performances in multicore CPU. The number of cores used can be manually selected from the terminal when executing the compile_and_run.sh file.
- In many function some temporal buffers are used (temp_*). These are swapped at the end of every step through *std* :: *swap*. The aim here is to reduce write conflicts and ensures the proper preservation of information along with a fast process.
- Through some internal parameters (as bc_type for example) it is possible to run tests on different domains and boundary conditions without modifying the source code.

6 Numerical Results

The results of the simulation are automatically saved in the build folder. Here we have two subfolders, graphs and video, and the file simulation_time_plasma_details.csv which is particularly useful to evaluate the scalability of the code.

6.1 Physical results

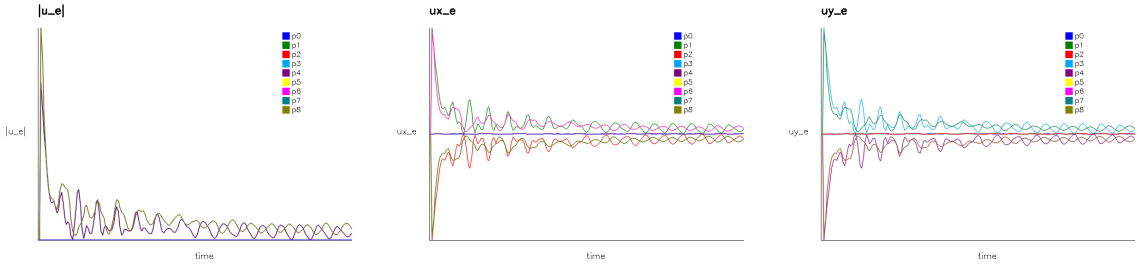
To access the effectiveness of the simulation we first focus on the content of the graphs and videos folders, which provide a visual representation of the main physical quantities as they evolve over time. In particular the graphs subfolders contains plots that track the time evolution of the main variables at 9 selected grid cells, symmetrically distributed. This allows for a localized yet representative analysis of the system dynamics.



(a) Magnitude (b) X component (c) Y component

Figure 2: Electric field

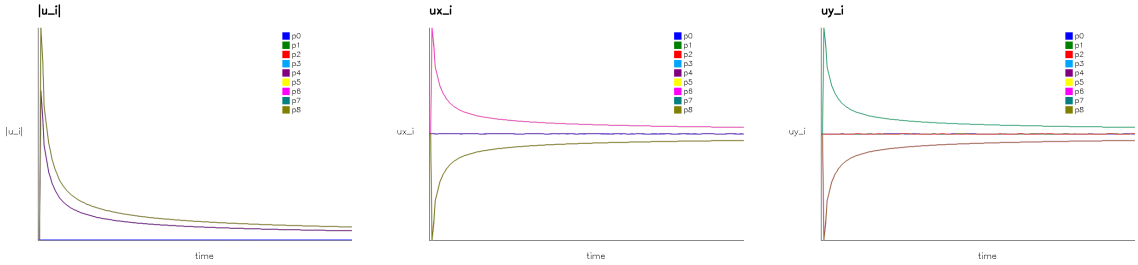
The effect of the oscillating electric field, shown in figure 2, is particularly visible for electrons, due to their much smaller mass compared to ions and neutral species. This is clearly reflected in their velocity evolution, reported in figure 3.



(a) Magnitude (b) X component (c) Y component

Figure 3: Electron velocity

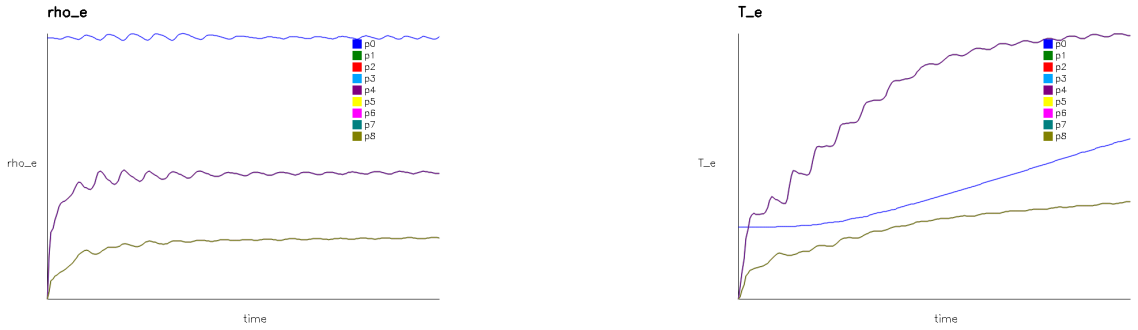
In both cases, we observe an oscillatory behaviour that can be interpreted as a collective phenomenon in plasma known as plasma oscillations or Langmuir oscillations. This is a self-sustained dynamic, arising from the coupling between charge density and electric field described by the Poisson equation (see subsection: 3.3). Then we notice a sort of exponential relaxation that is related to the collisions. Without them we expect the plasma to oscillate always at the same frequency: the plasma frequency. The velocities profiles of more massive species don't show oscillations due to their slower dynamics as we can see in 4. However we can see the overall relaxation phenomena that follows the same dynamic of electrons if we neglect oscillations.



(a) Magnitude (b) X component (c) Y component

Figure 4: Ion velocity

Always focusing on the more dynamic behaviour of electrons, figure 5 shows the evolution of their density and temperature. It is evident how the electron population expands from the grid centre to the boundaries, progressively approaching an asymptotic value given by the equilibrium and collisional processes. The temperature behaviour, instead, shows an increase over time until reach an asymptotic value. This is due to the physical approximation that neglect any form of temperature dissipation. In fact, in our framework, the kinetic energy is completely converted into thermal energy. As a result we observe a gradual decrease in the velocities and a rise in temperatures. A more accurate and complete model would include dissipative mechanism to provide a more realistic temperature evolution.



(a) Density (massive) (b) Temperature

Figure 5: Electrons variables

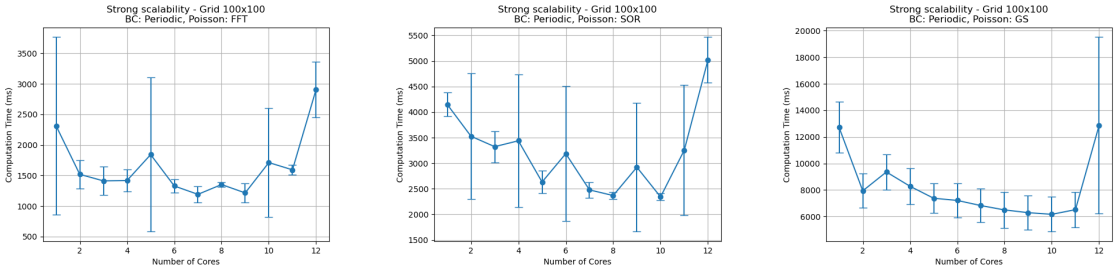
Videos describing the step-by-step evolution of the three species are then generated in the video subfolder. These animations shows the same oscillating behaviour in the velocity fields as before. This is enhanced in electrons because of their smaller masses but we could manually boost the ion velocity to see also there those oscillations.
Regarding the temperature evolution depicted in the videos, it is important to notice once again that in our model we include only a simplified treatment (see section 4), rather than a full physical description of energy dynamics. Within this framework, we observe the formation and expansion of a hot shell in the ion temperature plot T_i. This can be explained considering that electrons, being much

lighter, accelerate more rapidly than ions. As a result, the source term, that is linked to the energy loss during collisions, becomes more significant in areas where there is ion accumulation. These regions have a higher collision rate, which locally increases the temperature. This interpretation is supported even by the neutral species temperature T_n plot that mirrors, to some extent, this behaviour. If we consider the collision probability for neutral species, indeed, we'll have a higher collision rate with ions with respect that with electrons.

6.2 Code performances

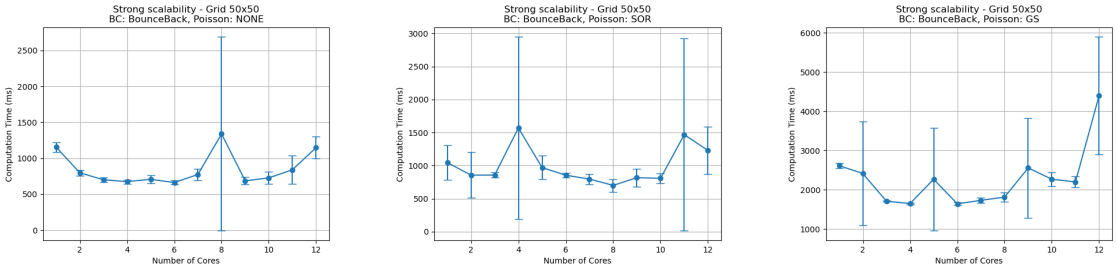
The efficiency of the code can be evaluated through different scalability analyses:

STRONG SCALABILITY: This test evaluates how the total computational time varies when increasing the number of processing cores while keeping the problem fixed. We performed this analysis for various Poisson solvers and boundary conditions, in order to observe how different combinations impact the trend.



(a) FFT method (b) SOR method (c) GS method

Figure 6: Strong scalability for Periodic Boundary Conditions



(a) No method for Poisson (b) SOR method (c) GS method

Figure 7: Strong scalability for Bounce-Back Boundary Conditions

As we can see from figure 6 the FFT method is clearly the most performant, exhibiting a significantly lower computational time compared to other solvers. In figure 7 it is evident then that omitting the Poisson equation the computational time would be further reduced, as we could expect.

WEAK SCALABILITY: In this case, both the domain and the number of cores are proportion-

ally increased in order to keep a fixed load for each core used. We reported for simplicity only the configuration with periodic boundary conditions and a FFT based Poisson solver, reporting the performances results for different fixed loads. The analysis, however, could be done for each configuration. Ideally the computational time should remain constant when the load is fixed. However, a gradual increase is observed in figure 8, especially for heavier loads. Despite this, the FFT solver maintains a relatively stable performance trend, making it suitable for parallel applications.

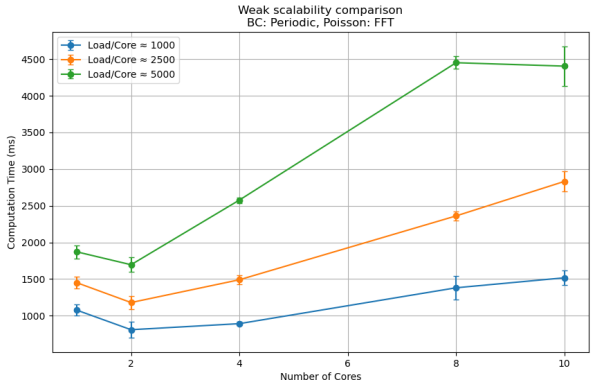
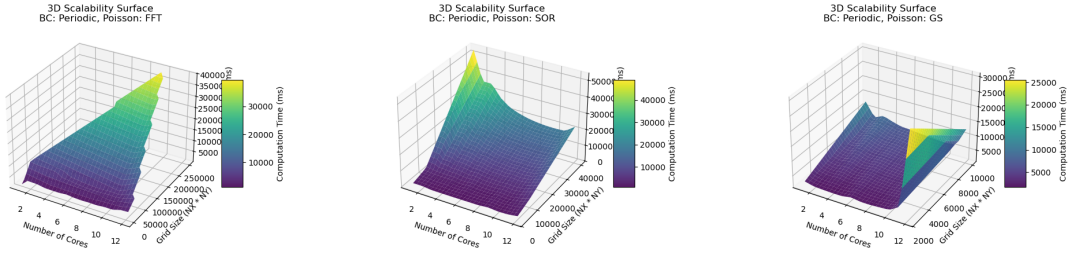


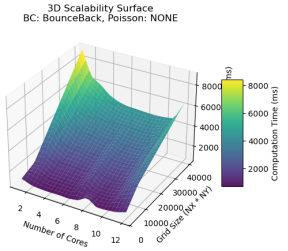
Figure 8: Weak scalability for different load values

GRID SIZE IMPACT: We then studied how the computational time is affected by the grid size. We can evaluate these results even in function of the core number exploiting a 3D heat surface:

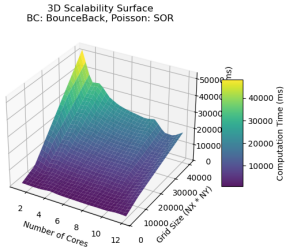


(a) FFT method (b) SOR method (c) GS method

Figure 9: Grid Size impact for Periodic Boundary Conditions



(a) No method for Poisson



(b) SOR method

Figure 10: Grid Size impact for Bounce-Back Boundary Conditions

Among all methods, the FFT one consistently outperforms the others, exhibiting a lower computational times even increasing the grid size. In contrast, SOR and GS methods always show poorer time performances, but a better scalability. Notably, in the absence of the Poisson solver, the computational time remains lower and scales efficiently with both grid size and core number.

7 Conclusions

The code works as expected under controlled and well-defined simulation conditions. However, the physical model relies on several critical approximations. As a consequence, modifying key parameters, such as species densities or the magnitude of the external electric field, without proper consideration can lead to significantly different physical regimes. In such cases, some corrections must be applied, mainly focusing on the rescaling of the relaxation time parameters. This is crucial for both the physical accuracy and numerical stability of the simulation. Improper tuning of these relaxation times may result in unphysical behaviour or even lead to numerical instabilities that can invalidate the simulation results. A careful calibration of these parameters should be considered before the configuration change.

A future development could be to implement adaptive or self-scaling relaxation times $\tau_{a,b}$ based on the evolving physical conditions. However such a feature would require a huge conceptual effort as it implies a deep understanding of all possible regimes and their associated constraints.

Other possible future implementation, considering real laboratory plasma systems, could be to include AC/DC electric field or a pressure control mechanism. The ultimate goal would be to implement a realistic plasma discharge, ignited by an applied voltage in the presence of a gas flux and a pressure regulation. This latter would emulate the dynamics of a plasma in a pump-controlled environment.

References

1. Bhatnagar PL, Gross EP, and Krook M. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. Physical review 1954;94:511.

2. Zhang X, Deguchi Y, and Liu J. Numerical simulation of laser induced weakly ionized helium plasma process by lattice Boltzmann method. Japanese Journal of Applied Physics 2012;51:01AA04.
3. Peng Y, Shu C, and Chew Y. Simplified thermal lattice Boltzmann model for incompressible thermal flows. Physical Review E 2003;68:026701.
4. Krause MJ, Kummerländer A, Avis SJ, et al. OpenLB—Open source lattice Boltzmann code. Computers & Mathematics with Applications 2021;81:258–88.
5. Li H and Ki H. Lattice Boltzmann simulation of weakly ionized plasmas and fluid flows using physical properties of fluids. Journal of Physics A: Mathematical and Theoretical 2009;42:155501.
6. Guo Z, Zheng C, and Shi B. Discrete lattice effects on the forcing term in the lattice Boltzmann method. Physical review E 2002;65:046308.
7. He X, Shan X, and Doolen GD. Discrete Boltzmann equation model for nonideal gases. Physical Review E 1998;57:R13.
8. Sköllermo G. A Fourier method for the numerical solution of Poisson's equation. Mathematics of Computation 1975;29:697–711.
9. Li Q, Luo K, He Y, Gao Y, and Tao W. Coupling lattice Boltzmann model for simulation of thermal flows on standard lattices. Physical Review E—Statistical, Nonlinear, and Soft Matter Physics 2012;85:016710.
10. Mai HC, Lin KH, Yang CH, and Lin CA. A thermal lattice Boltzmann model for flows with viscous heat dissipation. Computer Modeling in Engineering & Sciences(CMES) 2010;61:45–62.