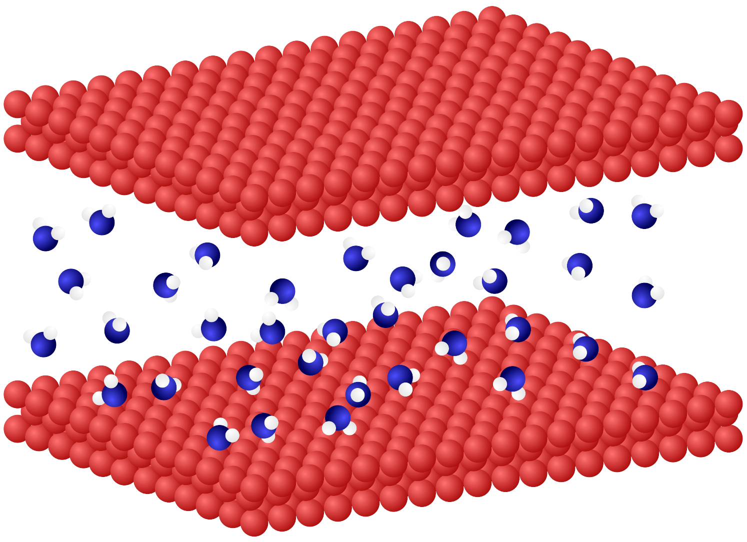
CFD for Micro and Nano flows

An Introduction to Molecular Dynamics and the lattice-Boltzmann Method

Tom-Robin Teschner



MSc in Computation Fluid Dynamics

School for Aerospace, Transport and Manufacturing

Cranfield University

Contents

1 Molecular Dynamics

1.1 Prerequisites and Assumptions

When we are dealing with fluid mechanics, no matter at which time and length scale, our governing equations are always constrained by various assumptions. The Navier–Stokes equations are only valid as long as the continuum hypothesis is valid. The continuum itself is assessed based on the Knudsen number, which is given as

$$\mathrm{Kn} = \frac{\lambda}{l_c}, \tag{1.1}$$

where λ is the mean free path and l_c is a characteristic length of the simulation domain. The mean free path is the distance travelled by an atom before it collides with another atom, at which point it changes its direction and energy level. For air at ambient pressure and temperature it is of the order of 10^{-8} meters and so for any macroscopic flow, the Knudsen number will be rather small. Usually, at around $\mathrm{Kn} = 0.001$, the continuum hypothesis is said to become invalid. This means that the characteristic length scale of the flow domain is now of the order of 10^{-5} meters for which the Navier–Stokes equations can still be used (for air at ambient pressure). In the range of $0.001 < \mathrm{Kn} < 0.1$, the no-slip boundary condition at solid walls is no longer valid and the flow starts to creep. The slip behaviour gradually becomes more important. We can still use Navier–Stokes here but have to make modifications at solid walls to account for the slip behaviour (and other surface effects which occur at boundaries, specific to the problem of interest). The transition regime is given between $0.1 < \mathrm{Kn} < 10$ in which the behaviour of the flow is changing to a pure particle-based one. Cases with $\mathrm{Kn} > 10$ are said to be free molecular flows and for even higher Knudsen numbers, the flow enters the rarefied gas regime. Here, particle interactions become important and need to be treated explicitly, hence the continuum is not longer valid and the Navier–Stokes equations can no longer be used.

Molecular Dynamics is a general particle model which is also valid for continuous flows. It is, however, computational rather expensive (see Section 1.7) so that it is only applicable to smaller domains. Just like the Navier–Stokes equations, Molecular Dynamics starts out with Newton's second law of motion which is then integrated for each atom where inter-particle forces determine the movement of each atom. Just as we have assumed the continuum to be valid for the Navier–Stokes equations, we also have to make assumptions about Molecular Dynamics. First, each particle is represented by a sphere. The sphere of each atom is charged and depending on the inter-particle distance, positive (attractive) or negative (repulsive) forces are exerted on the atom which depends on the distance to its neighbours. For very low temperatures (close to zero Kelvin), quantum effects become dominant which we will not include in our basic description on the method. However, it has to be kept in mind if the low temperature regime is of interest. There are other assumptions which will be addressed in the following sections in more detail.

1.2 The Equation of Motion

As stated in the previous section, Newton's second law of motion,

$$m_i \frac{\partial^2 \mathbf{r}_i}{\partial t^2} = \mathbf{F}(r_{ij}) \tag{1.2}$$

is integrated and the atoms are advanced in time. Here, m_i is the mass of atom i, $\mathbf{r}_i$ the position vector and $\mathbf{F}(r_{ij})$ is the inter-particle force. The force is always due to an interaction of an atom i and j which are separated by r_{ij} and we have $i \neq j$. It is usually more convenient to express the force in terms of the potential energy as

$$m_i \frac{\partial^2 \mathbf{r}_i}{\partial t^2} = \frac{\partial U(r_{ij})}{\partial \mathbf{r}}. \tag{1.3}$$

The exact form of the potential $U(r_{ij})$ is commonly approximated with a suitable model. The typical form is shown in Figure 1. Here, the potential itself is obtained by the superposition of the

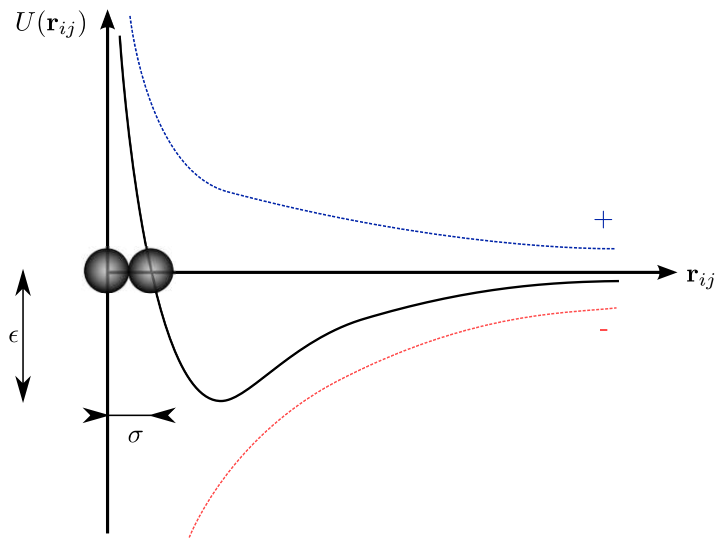


Figure 1: Inter-particle potential function. The superposition of the attractive (red) and repulsive (blue) forces yield the potential function shown in black. The minimum is at ϵ which is at the atomistic diameter of σ.

attractive and repulsive forces which has its maximum at $r_{ij} = \sigma$, where σ represents the atomistic diameter. The depth of the potential well is denoted by ϵ and it measures the strength of the atomistic interactions. Two atoms are schematically shown for scale. For monoatomic systems, the Lennard-Jones potential is the most widely used model which takes the form of

$$U(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]. \tag{1.4}$$

In order to obtain the forces on each atom, we need to differentiate $U(r_{ij})$ according to Eq.(1.3) for which we obtain

$$\frac{\partial U(r_{ij})}{\partial \mathbf{r}} = \mathbf{F}(r_{ij}) = -\frac{48\epsilon}{r_{ij}} \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \frac{1}{2} \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]. \tag{1.5}$$

Eq.(1.5) gives the magnitude of the force but in order to integrate our governing equation, Eq.(1.2), we need to split the force contributions according to their components. We simply multiply by the normalised direction of the atoms as $\mathbf{r}_k / r_{ij}$, where $k = 1, 2, 3$ represents the three coordinate directions, to obtain

$$\mathbf{F}_x(r_{ij}) = -\tfrac{48\epsilon}{r_{ij}^2} \left[\left(\tfrac{\sigma}{r_{ij}} \right)^{12} - \tfrac{1}{2} \left(\tfrac{\sigma}{r_{ij}} \right)^{6} \right] r_x, \tag{1.6}$$

$$\mathbf{F}_y(r_{ij}) = -\tfrac{48\epsilon}{r_{ij}^2} \left[\left(\tfrac{\sigma}{r_{ij}} \right)^{12} - \tfrac{1}{2} \left(\tfrac{\sigma}{r_{ij}} \right)^{6} \right] r_y, \tag{1.7}$$

$$\mathbf{F}_z(r_{ij}) = -\tfrac{48\epsilon}{r_{ij}^2} \left[\left(\tfrac{\sigma}{r_{ij}} \right)^{12} - \tfrac{1}{2} \left(\tfrac{\sigma}{r_{ij}} \right)^{6} \right] r_z. \tag{1.8}$$

Since the force needs to be evaluated for each atom i and j, the calculation of the force is of the order of $\mathcal{O}(N^2)$ and therefore the most computationally expensive part. In reality, the force beyond a couple of atomistic diameter is negligible and so it is common to define a threshold beyond which the force evaluation is skipped. This is denoted by the cutoff radius r_{cut} and a typical value is around $r_{\text{cut}} = 2.5\sigma$.
The following code snippet shows how the force calculation is implemented in MatLab. The number of atoms is given by *natoms*, the coordinates are stored in $atom(natoms, 3)$, where the first index goes over all atoms and the second from one to three to account for all three coordinate directions. The length of the domain in x, y and z is given by lx, ly and lz, respectively. The cutoff radius is given by *rcut*.

```
 1 F(:,:) = 0;
 2 fx = 0;
 3 fy = 0;
 4 fz = 0;
 5 for i=1:natoms-1
 6   for j=i+1:natoms
 7     % inter atomic distance
 8     rx = atom(i,1) - atom(j,1);
 9     ry = atom(i,2) - atom(j,2);
10     ry = atom(i,3) - atom(j,3);
11     rij = sqrt(rx*rx + ry*ry + rz*rz);
12       % compute force if i and j atoms are close enough
13       if(rij<rcut)
14       % inter-particle force
15       fx = -(48.0*epsilon/(rij*rij))*((sigma/rij)^12-0.5*(sigma/rij)^6)*rx;
16       fy = -(48.0*epsilon/(rij*rij))*((sigma/rij)^12-0.5*(sigma/rij)^6)*ry;
17       fz = -(48.0*epsilon/(rij*rij))*((sigma/rij)^12-0.5*(sigma/rij)^6)*rz;
18       % sum up contributions
19       F(i,1) = F(i,1) + fx;
20       F(j,1) = F(j,1) - fx;
21       F(i,2) = F(i,2) + fy;
22       F(j,2) = F(j,2) - fy;
23       F(i,3) = F(i,3) + fz;
24       F(j,3) = F(j,3) - fz;
25     end
26   end
27 end
```

It is worth pointing out at this stage that our loop in i is going from 1 to $natoms - 1$ while the loop over j is going from $i + 1$ to $natoms$. This is in order to reduce the computational time spent in the force computation loop. Looping over all atomistic pairs in this way ensures that each pair is only visited once during the force update so we have reduced our computational time to $\mathcal{O}(N^2/2)$. Since we are visiting each pair only once, we have to apply Newton's third law in line 19–24 where we add and subtract the force on both the i and j atom. The force is then stored in the array F, which has the same dimensions as the array $atom$.

1.3 Time Integration Techniques

Methods for integrations are usually derived based on the requirements. These do not necessarily need to be the same for different methods. We saw that the force evaluation is the most time consuming part in the calculation procedure and any method which requires several force evaluations during a single timestep should be avoided if possible. We also seek to minimize the storage demand, i.e. having to store, ideally, only the positions, velocities and accelerations. This is, however, seldom possible. Higher order time integration schemes are an essential necessity of each Molecular Dynamics simulation. Think of two simulations with the exact same initial and boundary conditions. If you only change the initial position of a single atom in one simulation, it will change the outcome of the simulation in the sense that after a couple of hundred timesteps, you would not be able to see that both simulation had almost the same initial conditions. Each interaction between particles will accumulated and eventual cause the simulation to march in a complete different direction. This is also contributing (if not the sole factor) to the non-linear behaviour of the Navier–Stokes equations. Having said that, it becomes clear that deterministic simulations are of little interest. We are not interested in each atom's trajectory but rather in their averaged behaviour. Therefore, we have to make use of statistical methods and sample quantities of interest from our simulations. This also means that we have to perform a large number of timesteps in order to get statistical converged results. A higher order scheme which can further improve the stability of the solution may be advantageous as we can run larger timesteps compared to less stable schemes. Less force evaluations need to be carried out and so the computational efficiency increases. The cost of the integrations scheme itself is not critical, a slightly more complex one may drastically reduce computational time.
All this goes into the consideration of designing an efficient, robust and accurate scheme. We will

limit ourselves here to the third order Verlet algorithm which is simple and easy to understand. We start with a Taylor series expansion around the position vector $\mathbf{r}$ forward in time and obtain

$$\mathbf{r}(t+\Delta t) = \mathbf{r} + \frac{\partial \mathbf{r}}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 \mathbf{r}}{\partial t^2}(\Delta t)^2 + \mathcal{O}(\Delta t)^3. \tag{1.9}$$

The same can be done backward in time to obtain

$$\mathbf{r}(t-\Delta t) = \mathbf{r} - \frac{\partial \mathbf{r}}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 \mathbf{r}}{\partial t^2}(\Delta t)^2 + \mathcal{O}(\Delta t)^3. \tag{1.10}$$

Adding Eq.(1.9) and Eq.(1.10) together yields

$$\mathbf{r}(t+\Delta t) + \mathbf{r}(t-\Delta t) = 2\mathbf{r} + \frac{\partial^2 \mathbf{r}}{\partial t^2}(\Delta t)^2 + \mathcal{O}(\Delta t)^3. \tag{1.11}$$

Rearranging so that only knows are on the RHS, unknowns on the LHS and inserting Eq.(1.2) into Eq.(1.11) yields

$$\mathbf{r}(t+\Delta t) = 2\mathbf{r} - \mathbf{r}(t-\Delta t) + \frac{\mathbf{F}(r_{ij})}{m_i}(\Delta t)^2 + \mathcal{O}(\Delta t)^3. \tag{1.12}$$

We see that we have retained the third order accuracy from the Taylor series by keeping the acceleration which we substituted by our governing equation. We require to store the position vector at $t - \Delta t$ but we only evaluate the forces once. If we are starting the simulation, we have to use Eq.(1.9) for the first timestep and then switch to Eq.(1.12) for subsequent iterations. We note that we have eliminated the velocity, i.e. $(\partial\mathbf{r}/\partial t)\Delta t = \mathbf{v}\Delta t$ in Eq.(1.12). The velocity can be simply reconstructed using a second order central scheme as

$$\mathbf{v} = \frac{\mathbf{r}(t+\Delta t) - \mathbf{r}(t-\Delta t)}{2\Delta t}. \tag{1.13}$$

We are now facing a problem which has not come up before. While our governing equation (Eq.(1.2)) does conserve momentum, it does not conserve energy. To understand how we can circumvent this problem, we need to borrow findings from kinetic gas theory, which is the study of the bulk behaviour of atoms. From kinetic gas theory we know (without proof) that the temperature and velocities are just scaled values of the kinetic energy and we have $U_{\text{kin}} = (3/2)Nk_bT$ for the temperature and $U_{\text{kin}} = (1/2)m\mathbf{v}^2$ for the velocity. We can equate both to each other and solve for the temperature so that we get $T = (2/3)U_{\text{kin}}/(Nk_B)$. If we define now a reference temperature at which we want to keep our system (more on that in Section 1.6) which ensures the conservation of energy, we have to divide aforementioned equation by itself using a reference temperature which will yield a desired velocity $\mathbf{v}_d$, i.e.

$$\frac{\frac{3}{2}Nk_BT_{\text{ref}}}{\frac{3}{2}Nk_BT} = \frac{\frac{1}{2}m\mathbf{v}_d^2}{\frac{1}{2}m\mathbf{v}^2}. \tag{1.14}$$

This is simplified to

$$\mathbf{v}_d = \mathbf{v}\sqrt{\frac{T_{\text{ref}}}{T}}. \tag{1.15}$$

After each timestep we have to rescale our temperature to ensure that the overall energy is not accumulating. There is a second, more sophisticated approach which makes use of so called thermostats. These are heat baths which are coupled to the governing equations. For example, the Berendsen thermostat modifies Eq.(1.2) to yield

$$m_i\frac{\partial^2 \mathbf{r}_i}{\partial t^2} = \frac{\partial U(r_{ij})}{\partial \mathbf{r}} + m_i\gamma\left(\frac{T_{\text{ref}}}{T} - 1\right)\mathbf{v}_i. \tag{1.16}$$

The addition of the heat bath term, where γ is the heat bath coupling strength (between zero and unity), allows the temperature to be weakly controlled. The energy is not exactly conserved but the system is reproducing more physical results (no step change in atomistic velocities).
Bringing now all together, the Verlet algorithm, making use of the rescaling procedure to conserve energy, is implemented in the following listing.

```
if(stp == 1) % if first time step
  for i=1:natoms
    % new coordinates in x, y and z
    atom(i,1) = atom_old1(i,1) + dt*vel(i,1) + 0.5*dt*dt*F(i,1);
    atom(i,2) = atom_old1(i,2) + dt*vel(i,2) + 0.5*dt*dt*F(i,2);
    atom(i,3) = atom_old1(i,3) + dt*vel(i,3) + 0.5*dt*dt*F(i,3);
  end
else
  % kinetic energy to rescale velocities
  ekin = 0;
  for i=1:natoms
  % new coordinates in x and yatom2
  atom(i,1) = 2*atom_old(i,1) - atom_old2(i,1) + dt*dt*F(i,1)/m(i);
  atom(i,2) = 2*atom_old(i,2) - atom_old2(i,2) + dt*dt*F(i,2)/m(i);
  atom(i,3) = 2*atom_old(i,3) - atom_old2(i,3) + dt*dt*F(i,3)/m(i);

  % new velocities
  vel(i,1) = (atom(i,1) - atom_old2(i,1))/(2*dt);
  vel(i,2) = (atom(i,2) - atom_old2(i,2))/(2*dt);
  vel(i,3) = (atom(i,3) - atom_old2(i,3))/(2*dt);

  % calculate kinetic energy
  ekin = ekin + 0.5*m(i)*(vel(i,1)*vel(i,1)+vel(i,2)*vel(i,2)+vel(i,3)*vel(i,3));
end

% system temperature based kinetic energy
T = (2.0/(3.0*natoms*kb))*ekin;

% scaling factor for velocities
scale_vel = sqrt(Tref/T);

% scale velocities
ekin = 0;
for i=1:natoms
  vel(i,1) = vel(i,1)*scale_vel;
  vel(i,2) = vel(i,2)*scale_vel;
  vel(i,3) = vel(i,3)*scale_vel;
end
```

1.4 Boundary Conditions

Every computational approach needs to implement boundary conditions and initial conditions. When we are dealing with atomistic simulations, these are not straight forward to implement and they need some special consideration. First of all consider the simulation domain, on the macroscale we are able to define a clear volume. On the microscale, where atoms do not have a volume but rather an electrical shell surrounding the nucleus, the volume definition becomes problematic. Defining clear lines and surfaces, where our simulation domain ends, is not possible because atoms may be penetrating such objects. How are we then able to impose any boundary conditions if in-fact the boundaries themselves are not well defined? We will consider the problem of the volume again in Section 1.6 but for now lets make a crude approximation and define a cubic volume with straight surfaces. In Figure 2 we have depicted a typical set up of a 2D nano channel flow of length L_x, where solid wall atoms are represented in red which are on a regular lattice while the blue fluid atoms are moving irregularly in-between. Let us first consider the fluid-wall interactions. Depending on the property of each atom group, their interaction potential, i.e. Eq.(1.4) may differ by the atomistic diameter σ and its potential of the interaction, ϵ. It is useful to introduce indices when dealing with multiple potentials and we will use ϵ_{ff} when we are dealing with fluid-fluid interactions (only blue atoms exert forces on each other, away from the wall) and ϵ_{fw} when we are dealing with fluid-wall interactions (blue with red atoms, close to the wall, i.e within r_{cut}). Now we can have different behaviours: If $\epsilon_{\text{ff}} < \epsilon_{\text{fw}}$, then the blue fluid atoms will be attracted towards the wall and the probability of finding fluid atoms there is enhanced. This will also contribute towards a no-slip condition at the wall. If $\epsilon_{\text{ff}} > \epsilon_{\text{fw}}$, however, then the wall

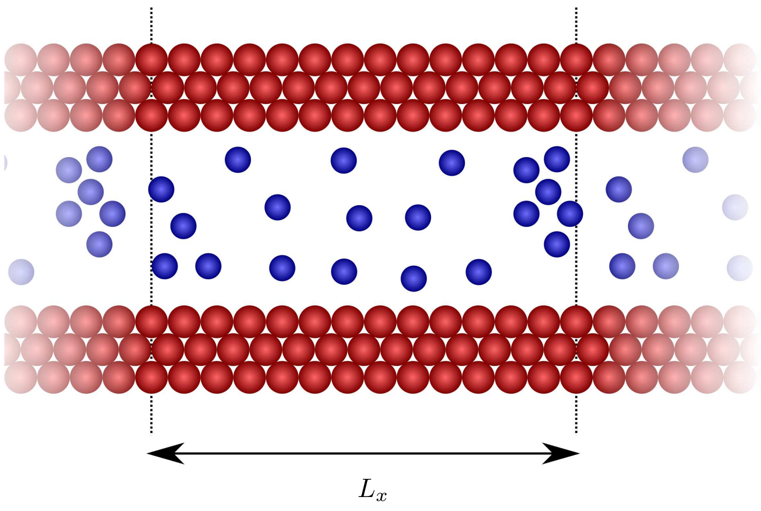


Figure 2: Channel flow on the microscale. Red atoms are representing solid wall boundaries while blue atoms move freely inside the channel of size L_x beyond which periodic images are created of the channel.

atoms may not posses enough energy to retain fluid atoms close to the wall and the slip behaviour is observed. In Figure 2, the red wall atoms are on a regular and perfect lattice structure which by default reduces drag at the wall but in reality we have a surface roughness which would trap fluid atoms to further enforce the no-slip condition. Now let us have a look at open, i.e. in- and outflow conditions. Physically speaking, no such boundary conditions exist in reality. All atoms are smoothly connected to other atoms throughout the universe which may only change at the edge of the universe. For all of our intents and purposes, we are only interested in small domains on the order of 10^{-9} meters. The problem with in- and outflow boundaries is two-folded; first, inserting and removing atoms will change the energy level and secondly, atoms near open boundaries will experience a reduced force due to missing particle beyond the boundary. The first problem can be addressed by either rescaling the velocities as discussed previously or to insert atoms at specific points in space where the total amount of energy does not change, i.e. where inserted atoms have the same energy as the removed atoms (but not necessarily at the same position). This is not a trivial process and goes beyond the scope of our discussion. We simply note that open boundaries, due to their non-physical nature, are difficult to implement. Furthermore, the missing interactions at the boundary would cause wrong particle trajectories and as we discussed in Section 1.3, it is vital that we capture the trajectories as close as possible due to the non-linear behaviour. This is commonly overcome by making use of periodic images of the simulation domain. It is similar to periodic boundaries but not only the information at the boundaries is exchanged, the whole simulation domain is copied at the boundaries so that atoms close to the boundary are able to perform their force calculation. If open boundaries are desired, it is common practice to first simulate the domain with periodic boundary conditions from which the force at the boundary is sampled. This can then be introduced for the open boundary case to account for the missing interactions.

1.5 Reduced Units

When we deal with Navier–Stokes, we have the choice of using dimensional or non-dimensional units which have several benefits, however, dimensional units are just as commonly used as their non-dimensional counterpart. In Molecular Dynamics, this is also possible theoretically, but less practical. To appreciate this we need to look at the typical units of a molecular simulation. We have introduced the Boltzmann constant in section Section 1.3, its actual value is $k_B = 1.38064852 \cdot 10^{-23}$ m^2 kg s^{-2} K^{-1}. The typical timestep is of the order of femto (10^{-15}) seconds, the atomistic diameter of Ångström (10^{-10} meters), the simulation domain size between 10^{-10} and 10^{-7} meters and the temperature around 10^2 Kelvin. With increasing computational power, we are able to simulate ever so more molecules but for the moment the limit seems to be at around 10^9 atoms. Just from these considerations, we see that there are huge separations in orders of magnitude between the parameters used in the simulation. Therefore it is common to use so called *reduced units*, which are essentially non-dimensional units but get their name from the fact that atomistic properties are set to unity, i.e. $m = \epsilon = \sigma = k_B = 1$ so that the governing equation can be further simplified. For example, if the mass has a value of unity, the force becomes equal to the acceleration and we do not convert one to the other. The velocity will also be equivalent to the momentum. The main reason to opt for reduced units is that all parameters are close to unity and thus the numerical round-off error decreases (or we could say that we increase the signal-to-noise ratio). Specifically we have for some fundamental properties the following conversion, where variables marked with an asterisk are in reduced units:

$$
\begin{aligned}
n^* &= n\sigma^3 && (1.17)\\
T^* &= k_B T/\epsilon && (1.18)\\
E^* &= E/\epsilon && (1.19)\\
t^* &= (\epsilon/m\sigma^2)^{1/2} t && (1.20)\\
v^* &= \sigma/t && (1.21)\\
p^* &= p\sigma^3/\epsilon && (1.22)\\
F^* &= F\sigma/\epsilon && (1.23)\\
& && (1.24)
\end{aligned}
$$

Here, n is the number density, T the Temperature, E the (total) energy, t the time, v the velocity, p the pressure and F the force.

1.6 Simulation Ensembles

We have discussed the problem about the boundary conditions, that the volume is not something that we can easily fix. We further found that we are not able to conserve energy without introducing either a thermostat or the rescaling of velocity technique. We could say that Newton's second law alone, i.e. Eq.(1.2), is not enough to describe the system and that it is under-constrained. We need to fix some properties of the system in order to simulate it. There are a number of possibilities and these are named statistical ensembles. There are four main ensembles which we are going to briefly look at; the micro-canonical, the canonical, the grand-canonical and the isothermal-isobaric canonical ensemble.

In the micro-canonical (or constant NVE) ensemble, we fix the number of atoms N, the volume V and the total energy E. Conserving the number of atoms N is easily achieved with periodic boundary conditions, i.e. no atom is able to leave the domain. The volume is defined as we did previously by considering a three dimensional block (or rectangle in 2D). The number density is then calculated based on this fixed volume. The total energy is conserved by summing up the potential, kinetic and internal energy and ensuring that the value stays constant (by similar means that we used before, i.e. rescaling velocities or using a thermostat).

The canonical (or constant NVT) ensemble is very similar, however, here we are only interested in conserving the temperature. This is usually advantageous if we do not want to consider internal energy or we are just looking at the temperature. We are just conserving the total energy partly which is sufficient for certain applications.

The grand canonical (or μVT) ensemble conserves a chemical potential μ which does not necessarily conserve the number of atoms. Atoms may interact together and form new bonds but need to do so at a set chemical potential. In analogy, if we were not to control the temperature in the canonical ensemble, the atoms would start to accelerate until they reach non-physical speeds. Similarly, if we were not to control the chemical potential, reaction rates would increase unreasonably.

The final ensemble, the isothermal-isobaric (or constant NPT) one controls the temperature and pressure (via a so called barostat) while conserving the number of particles. Here the volume is not important and may change during the simulation. However, the pressure (and in effect the force) will be controlled which stop atoms from expanding indefinitely.

1.7 Reaching the limit

We know that Molecular Dynamics is computational expensive, but exactly how much time does it take? Let us assume we would like to compute the flow through a channel which is of dimension 10cm $\times$ 1cm $\times$ 1cm. We further assume ambient pressure of $p = 101325$ Pa and $T = 300$ Kelvin. From the equation of state we get the number of moles as $n_{mol} = (pV/RT) = (101325 \cdot 10^{-5})/(8.3144621 \cdot 300) \approx 0.4 \cdot 10^{-4}$ moles. We know that there are $6.022 \cdot 10^{23}$ atoms per mole, regardless of the composition (this is the famous Avogadro constant) so we have a total number of atoms as $n_{\text{atoms}} = 0.4 \cdot 10^{-4} \cdot 6.022 \cdot 10^{23} = 8.088 \cdot 10^{19}$. From experience we can say that the simulation of 10^7 atoms per timestep per CPU core takes 6.0 seconds, for 10^8 atoms 60.0 seconds and so on so that $8.088 \cdot 10^{19}$ atoms would take $48 \cdot 10^{12}$ seconds. This is equivalent to $1.5 \cdot 10^6$ years. However, this is only for one single timestep. If we want to simulate a flow of one second in duration and knowing that the typical timestep size is of the order of one femto second, the total computational time would be $1.5 \cdot 10^{21}$ years. We can conclude from this rough approximation that even with an efficient parallelisation strategy where we may be able to use up to 10^7 cores while still achieving a linear speed up, that results obtained on the macroscale obtained from molecular dynamics are out of reach.

2 The early days: Lattice Gas Cellular Automata (LGCA)

In the second half of the 20^{th} century, when computational resources allowed the simulation of more and more complex fluid systems, the diversity of research methods to obtain the same results increased. Molecular dynamics (MD) — establishing itself in the micro- and nanoflow community — paved the way for particle-based methods but was always limited to domain sizes of the order of 10^{-9} m and so remained limited to molecular length scales for future generations of researchers to come. This restriction still holds in the modern days of high-performance computing. At the same time, a new — but largely frowned upon — particle method was introduced by Bird, where the well established Boltzmann equation was combined with a stochastical collision operation, taking elements from the Monte Carlo method. It was termed the Direct Simulation Monte Carlo (DSMC) method and was rather elegantly describing the flow features at high Mach number and rarefied flows. Vanquishing initial scepticism, the DSMC mehtod has established itself as the de-facto standard method for rarefied flows and is of special interest to space applications.
The Navier–Stokes equations also saw their own renaissance when it got reformulated into a Lagrangian framework so that particles now adhered to the classical Navier–Stokes equations while being able to model micro- to macroscopic length-scales. This method has now become known as Smoothed Particle Hydrodynamics (SPH) but is less frequently used due to its problem of mass conservation and higher-order treatment. None-the-less, due to its particle nature, no computational mesh is required and so starting from an initial seed of particles, the flow domain is mapped soley based on their interactions with one another.
The discrepancies in length-scales between the Navier–Stokes and Molecular Dynamics method motivated the development of so called mesoscopic methods which are loosely defined as sitting in the middle of microscopic and macroscopic methods. The Dissipative Particle Dynamics (DPD) is one such representative which is essentially an up-scaled version of the MD method, following the same governing equations where single particles are lumped into larger groups which are represented by a larger particle. This particle is non-physical but due to its size increase, it is feasible to model much larger systems compared to classical MD simulations. The method lacks realistic thermal fluctuations (due to the grouping of smaller particles into larger ones) which have to be modelled through random thermal forcing terms. Despite this shortcoming, the DPD method has enjoyed a wide user community and is at the forefront of accepted and highly-used particle methods. In recent years, the DPD and SPH method were coupled to form the Smoothed Dissipative Particle Dynamics (SDPD) method. Despite its lack of popularity — attributed to its young age — it has removed the shortcomings of the DPD method and has formed a solid particle method for the mesoscale regime.

This sets the scene for the discussion to follow. At the same time, around the 1970s, the Lattice Gas Cellular Automata (LGCA), or simply Lattice Gas Automata (LGA), was yet another method trying to bridge the microscopic and macroscopic world. Its approach was fundamentally different from any of the above. While all of the above methods were derived from physically considerations, the LGCA method was partly based on physics, but has a large computational aspect to it (which ultimately sealed its own fate). It is not in use anymore and was surpassed by the lattice-Boltzmann method, which was directly derived from the LGCA method and improved upon its shortcomings.

The basic idea of the method is shown in Figure 3. A computational mesh is placed in the domain of interest and particles are put onto nodes. Some nodes may be left empty, some will be filled, according to the initial conditions (particle density). If we assume that particles 1 and 2 are on a head on collision, it needs to be decided where particle 1 and 2 will go. This is done by the so called collision rules, derived from collision models. For the case on the left, particle 1 can either go to location 3 or 4 and same is true for particle 2. However, if particle 1 goes to location 3, then particle 2 has to go to location 4 and vice versa. A random choice is made which particle will go to which location and the other particle is then moved to the opposite location. During each timestep, particles are either stationary at a node (and only move after a collision) or are in constant motion. This depends on the collision model used. Not all allow for stationary particles. A more elaborate case is the one shown on the right of Figure 3. The collision rules quickly escalate

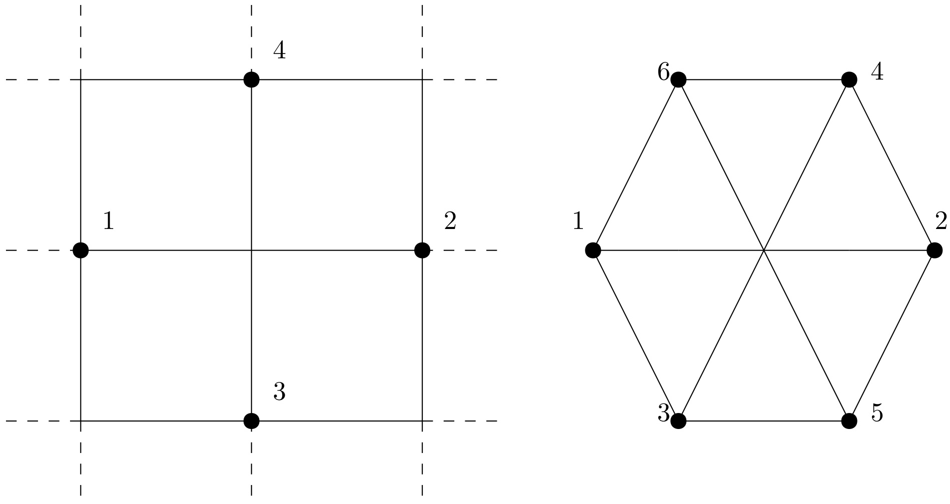


Figure 3: Left: Arrangement of particles on a square lattice, Right: Arrangement for a hexagonal lattice

and over a hundred different possible collisions need to be checked for each node. Not only does this increase computational time, but the randomness introduces numerical noise which was its biggest disadvantage. Furthermore, the extension to 3D is straightforward for the squared mesh on the left, but collision rules for more complex shapes in 3D become prohibitively large and unnecessarily complicated. Vortical flows appeared squared for coarser meshes and required a high amount of averaging over a large number of timesteps or an ensembled average of more than one simulation. With these shortcoming becoming evermore prominent and with the idea to replace particles by distribution functions, the LGCA method made way so that the lattice-Boltzmann method (LBM) could bring new research interest into a method which seemed to be on a declining path.

3 From LGCA to the lattice-Boltzmann method

The LBM and LGCA method share lot of similarities. A rather small tweak to the representations of the particles gave the method a whole new direction so that it became necessary to separate it from the LGCA method. The simple change included for the LBM is that the single-particle representation is exchanged for a Density Distribution Function (DDF). Instead of having particles stream and collide along the lattice (the lattice simply refers to the computational mesh), the DDF is streamed along the lattice and a collision is carried out if two DDFs collide at a node. This modification removed the numerical noise from the randomness of the collision but came at the cost of the collision itself; it is no longer possible to track the collision explicitly. A new formulation for the collision operator had to be found and, to this day, remains a problematic issue in the LBM, limiting it to low Reynolds number flows. In the next Sections, we will highlight each step using the LBM so to be able to implement it into a code.

3.1 Governing Equation

The governing equation for the lattice-Boltzmann method relies on heavy use of statistical mechanics, kinetic gas theory and the Boltzmann equation itself. For this brief introduction it would be too much to go into detail, we will limit ourselves here to a qualitative discussion. The equation for the LBM is

$$\frac{\partial f}{\partial t} + \mathbf{c} \cdot \nabla f = \Omega(f). \tag{3.1}$$

Here, f represents the DDF, c is the speed at which the DDF is streamed from one node to another and $\Omega(f)$ is the collision operator. Eq.(3.1) is thus a transport equation for the DDF. The knowledge of the DDF is not of interest and so we need to recover macroscopic properties from it,

which is discussed in Section 3.6. But comparing the governing equation of the LBM method to the Navier–Stokes equation, we immediately see that it is a single equation, not a system of equations, and that it is linear. This not only simplifies the calculation but also makes the LBM much faster than the Navier–Stokes equation approach. These advantages intrigued many researcher and the interest in the LBM increased exponentially. On top of that, particle-based approaches are known for their simplicity to include complex physical processes such as multiphase flows, turbulence, acoustics, porous media, electromagnetism. Although the LBM is an eulerian approach, its inherent particle description easily accommodates multi-physics treatment. It is a local approach, meaning each computational node is operating independently of each other (unlike conventional finite difference or finite volume approximations, where information of their neighbour nodes are needed). This means that parallelisation can be easily done of the algorithm. Another advantage is the ease at which complex geometries can be handled. Objects of any shape can be immersed in the flow domain and the flow calculation is carried out without complex coding needed.
This is the general description one reads when studying the LBM. There are, however, many drawbacks which are less frequently mentioned and require more complex considerations. In many cases, the Navier–Stokes equation become again more attractive. Common drawbacks are the lack of physical soundness in some models, making the LBM highly unstable for Reynolds numbers of any engineering importance. Modifications have been proposed with, however, only moderate success. It is restricted to low Mach number, isothermal flows. Extensions to compressible and non-isothermal cases have been attempted, but are not the primary area of applications of the LBM method. The extension to complex geometries may be straightforward, but there are no theoretical barriers on the Navier–Stokes side to adopt the same procedure. The reason for opting for a body-fitted, non-Cartesian approach using the Navier–Stokes equations its advantages in terms of reducing the computational mesh and increasing the accuracy for wall-bounded flows. However, the LBM has established itself for mesoscale flows, despite its relative short history.

Returning back to eq.(3.1), it consists of three parts. The first is the time derivative $\partial f/\partial t$ which indicates the temporal change of the DDF. The already mentioned collision operator $\Omega(f)$ has to be approximated by some form of suitable collision model. Note that the collision in the LGCA is exact, while in the LBM it is approximated. The only remaining term, $c \cdot \nabla f$, is the streaming step where the DDFs stream with the advection speed c from one node to another. Since the Cartesian mesh, on which the LBM operates, has a constant mesh spacing Δx, it follows that $\Delta x = \Delta t \cdot c$. The LBM uses, however, so called lattice units which are dimensionless. Thus, the same equation, in lattice units, is written $\delta x = \delta t \cdot c$. In dimensionless form, $c = 1$ and so $\delta x = \delta t$. This is also the reason why some literature refer to $\delta t = 1$ and instead of using the notation $t + \Delta t$ the notation $t + 1$ is used.

3.2 Speed models and the DnQm notation

Before proceeding, one important concept in the LBM needs to be introduced; the notion of a speed model which is commonly abbreviated with the $DnQm$ notation. Here, n represents the dimension used and m states how many links are used to construct a speed model. This is illustrated in Figure 4 for two different speed models. Links are directions on which the DDF can stream, so for a two-dimensional flow, having a $D2Q5$ speed model means that the DDF can stream to the north, south, east and west plus resting on the node, hence five directions (including the resting state). Using the diagonal components, one gets an additional four possible directions (north-east, north-west, south-east, south-west) equating to nine possible streaming directions. Hence, the model is named $D2Q9$. This is by far the most popular model as the $D2Q5$ suffers from similar drawbacks as the LGCA (square vortices) and so the $D2Q9$ model is the standard 2D model. Since the Cartesian grid is used irrespective of the speed model, extensions to 3D are easily achieved.

Each link in the speed model is now defined by its local velocities (hence the name speed model) and is given in terms of unit normal vectors. For the the $D2Q9$ model, we have for the x and y velocity:

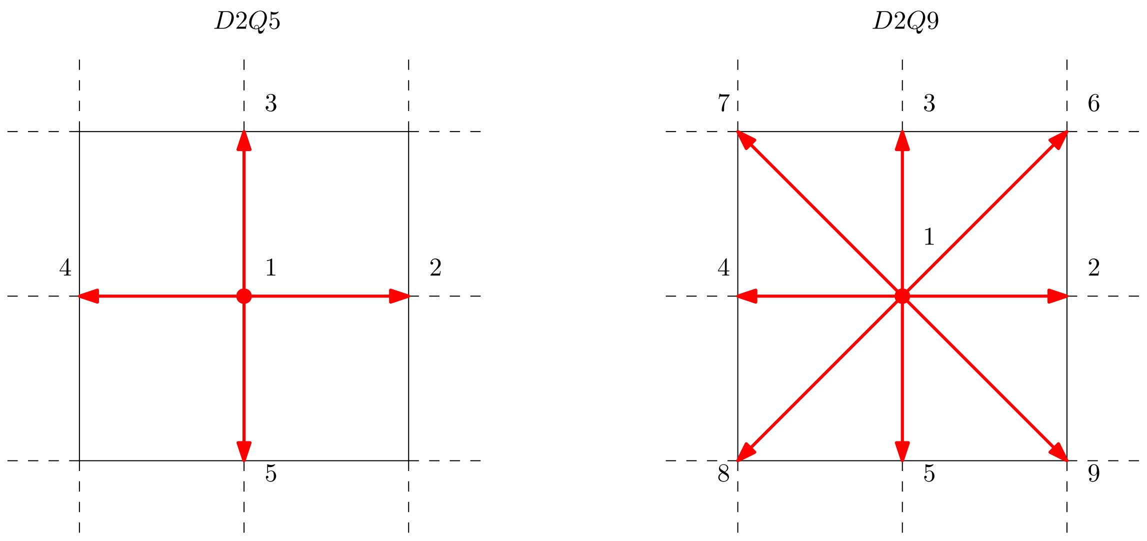


Figure 4: Two examples of a speed model for two-dimensional flows; Left: D2Q5 and Right: D2Q9

$$c_x = [0, 1, 0, -1, 0, 1, -1, -1, 1] \tag{3.2}$$
$$c_x = [0, 0, 1, 0, -1, 1, 1, -1, -1] \tag{3.3}$$

The first set of pairs, $c_x(1)$ and $c_y(1)$, are both zero, which represents the resting node. The second set is $c_x(2) = 1$ and $c_y(2) = 0$ which points in the x direction. Similar arguments results in the other directions as well and are numbered in Figure 4. Each direction has a different length to travel and so their influence has to be weighted. The weights for the $D2Q9$ model are given by

$$w_i = \begin{cases} 4/9, & i = 1 \\ 1/9, & i = 2, 3, 4, 5 \\ 1/36, & i = 6, 7, 8, 9 \end{cases} \tag{3.4}$$

Each weight corresponds to a link in the speedmodel. The largest weight is the first one, as the resting speed model does not travel any distant. The horizontal and vertical links have the same weights and are larger than the diagonal weights (which are also the same), as it travels a shorter distance. The determination of the weights is somewhat arbitrary but there are a set of rules which needs to be followed. The most important one is that the weights need to add up to unity so that $\sum w = 1$.

The speed model dictates how many DDFs there are at a single node. For the $D2Q9$ model, nine individual DDFs are present at each given node. This means that at each node, nine streaming and collision operations need to be carried out. This will be the discussion of the next two Sections.

3.3 A simple estimate for the collision operator

Even before the LBM was used, kinetic gas theory faced similar questions to that of the LBM itself. Bhatnagar, Gross and Krook put forward a collision model — named after the authors the BGK model — which provided a simple, isotropic approach, to approximate the collision of a collection of particles. It is given as

$$\Omega(f) = -\frac{1}{\tau}(f - f^{eq}). \tag{3.5}$$

Here, τ is the relaxation time (the time needed to return to equilibrium after a perturbation) and f^{eq} is the DDF at equilibrium. The derivation of f^{eq} is given in Section 3.5. This can now be inserted into eq.(3.1) to yield

$$\frac{\partial f}{\partial t} + c \cdot \nabla f = -\frac{1}{\tau}(f - f^{eq}). \tag{3.6}$$

With the knowledge of f^{eq}, eq.(3.6) can be solved. The equation can be written in discretised form

$$\frac{f(\mathbf{x}, t + \Delta t) - f(\mathbf{x}, t)}{\Delta t} + \mathbf{c}\frac{f(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t) - f(\mathbf{x}, t + \Delta t)}{\Delta\mathbf{x}} = -\frac{1}{\tau}(f(\mathbf{x}, t) - f(\mathbf{x}, t)^{eq}). \tag{3.7}$$

The streaming is evaluated at its final position, i.e. when the streaming has occurred (analogous to an implicit treatment), while the collision is treated explicitly. Making use of the fact that $\Delta\mathbf{x} = \Delta t \cdot \mathbf{c} = 1$, introducing $\omega = \Delta t/\tau = 1/\tau$, the equation can be simplified to yield

$$f(\mathbf{x}, t + \Delta t) - f(\mathbf{x}, t) + f(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t) - f(\mathbf{x}, t + \Delta t) = \omega(f(\mathbf{x}, t) - f(\mathbf{x}, t)^{eq}). \tag{3.8}$$

Collecting terms and rearranging yields the final form of the LBM using the BGK approximation as

$$f(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t) = (1 - \omega)f(\mathbf{x}, t) + \omega f(\mathbf{x}, t)^{eq}. \tag{3.9}$$

Since ω is the inverse of the relaxation time, it is termed the relaxation frequency and can be calculated as

$$\omega = \frac{1}{3\nu_l + 0.5}. \tag{3.10}$$

Here, ν_l is the lattice viscosity and can be converted to a physical viscosity. Setting this value too low, results in oscillation and ultimately divergence. A lower limit of $\nu_l = 0.1$ is usually sufficient. It is a user defined variable and so eq.(3.9) can be solved.
Note that eq.(3.9) is solving for $f(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t)$ (which in computational notation would be f_{i+1}^{n+1}). This is due to the underlying particle nature of the LBM method where we are interested in the evolution of the DDF in time and space, rather than just time.

3.4 Streaming process of the Density Distribution Function

As has been pointed out in Section 3.2, that each link of the speed model represents one streaming direction. During this process, each distribution function f is streamed on its current lattice link to the neighbour node. This process is illustrated in Figure 5. Take for example the node at i, j. The speed model at this link is indicated in red. The DDF at link 2 (refer back to Figure 4) is streaming from i, j to $i + 1, j$, so that the streaming for that direction is simply accomplished by setting $f_2(i + 1, j) = f_2(i, j)$. This is done for every direction and at every node i, j.
At boundaries, there are three incoming directions which can not be found from the streaming operation. To obtain these three missing links, appropriate boundary conditions need to be defined which is the subject of Section 3.7.

3.5 The density distribution function at equilibrium

We have already encountered the DDF at equilibrium in eq.(3.9). This function is an approximate version the Maxwell–Boltzmann distribution function of molecular velocities. It is given as

$$f_k(i, j)^{eq} = \rho(i, j)w(k)\left[1 + \frac{\mathbf{u} \cdot \mathbf{c}}{c_s^2} + \frac{(\mathbf{u} \cdot \mathbf{c})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2}\right]. \tag{3.11}$$

Here, ρ and $\mathbf{u}$ are the macroscopic density and velocity (how to obtain those will be discussed in the next Section), while $\mathbf{c}$ are the microscopic velocities of the speed model, i.e. eq.(3.2–3.3). The lattice speed of sound is $c_s = 1/\sqrt{3}$ and is depending on the speed model. To derive the DDF at equilibrium, one needs to discretise the Maxwell-Boltzmann velocity distribution function for the discrete velocities of $\mathbf{c}$ through a Taylor-Series approximation to arrive at eq.(3.11).

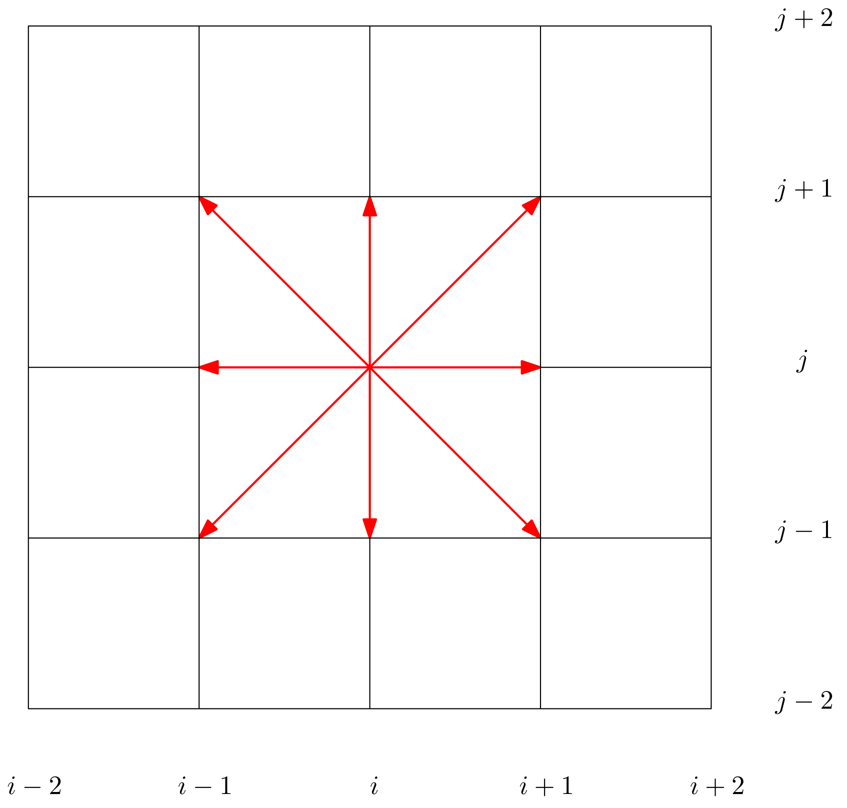


Figure 5: Streaming of the Density Distribution Function

3.6 Recovering macroscopic properties

Macroscopic quantities, primarily the density and momentum, are recovered directly from the DDF in the following way

$$\rho(i,j) = \sum_{k=1}^{m} f_k(i,j), \tag{3.12}$$

$$\rho(i,j)\mathbf{u}(i,j) = \sum_{k=1}^{m} f_k(i,j)\mathbf{c}_k. \tag{3.13}$$

The final velocity components of **u** are then recovered by dividing the momentum (eq.(3.13)) by the density (eq.(3.12)). The velocity component and density, however, are in lattice units and need to be transformed, in one way or another, to make statements about their true value. This is shortly introduced in section 3.8.

3.7 Microscopic and macroscopic boundary conditions

Boundary conditions for the LB method are probably the most difficult elements to impose on the system. For a classical Navier–Stokes approach, we solve for macroscopic quantities such as velocity, density, pressure and temperature directly, while in the LB method, we solve for the DDF and hence have to impose boundary conditions for the DDF itself. In a usual computations, we are not interested in the DDF by itself, but rather the macroscopic quantities which can be used to relate to dimensionless numbers like the Reynolds number. Hence, we have little knowledge on what the DDF should be at the boundaries. The next sections will cover boundary conditions of increasing complexity and show how to implement simple periodic to complex open boundary conditions. For this, we consider Figure 6 for the rest of the discussion where open and closed / solid boundary conditions are depicted. The unknown contributions of the DDF at the boundaries

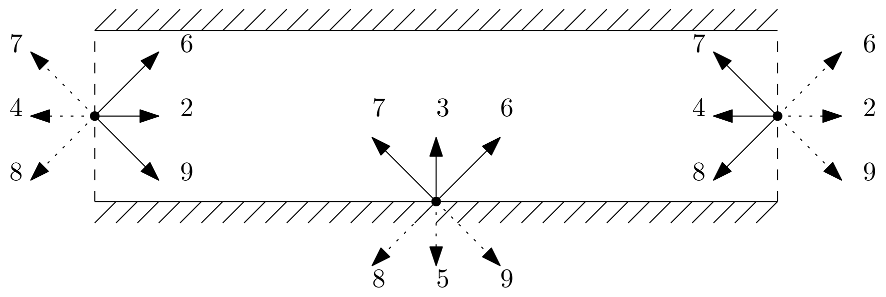


Figure 6: Sample geometry with open and closed (solid) boundary conditions. Solid boundaries are located at the north / south interface (solid line) while open / periodic boundary conditions are found at the east / west interfaces (dashed lines).

are denoted by solid lines while those known through streaming at open / periodic boundaries are shown as dotted arrows.

3.7.1 Periodic boundary conditions

The simplest form of boundary conditions are those that are periodic. Referring to Figure 6, links 2,6,9 at the west boundary are found by imposing the values of the links 2,6,9 at the east boundaries. The problem with those boundaries are that only a limited amount of problems can be simulated. Usually, fully periodic boundary conditions are limited to study homogeneous, isotropic decaying turbulence in a box while more sophisticated problems relevant to engineering disciplines require a mix of open, closed and even periodic boundary conditions.

3.7.2 Bounce Back

The bounce back scheme is the simplest of boundary conditions to model solid walls. They can either be used to model no-slip (wall) or slip (symmetry) boundary conditions. At the south face, we have the unknown DDF at links 3,6,7. The bounce back scheme assumes that particles bounce back at the wall so that we can find the missing DDFs by setting $f_3 = f_5$, $f_6 = f_8$ and $f_7 = f_9$. This may seem counter-intuitive for some as it would be expected that the particles would continue to travel in the direction from which they are coming form. However, in order to achieve a no-slip velocity component at the wall, it is necessary that the DDFs bounce back in the opposite direction so that the net velocity at the wall is equal to the wall velocity. On the other hand, if one wants to create a slip velocity, i.e. impose a solid boundary which does not constrain the velocity at the wall (symmetry boundary conditions), the opposite needs to be done. Here, we set $f_3 = f_5$, $f_6 = f_9$ and $f_7 = f_8$, now a slip velocity can occur.

3.7.3 Extrapolation

For open boundaries, the missing DDFs may be simply found by extrapolating the DDF's from the inside of the domain onto the boundaries. At the east boundary, the missing DDFs may be found through the following, first-order extrapolation as

$$\begin{aligned} f_4(i_{max}, j) &= f_4(i_{max} - 1, j), \\ f_7(i_{max}, j) &= f_7(i_{max} - 1, j), \\ f_8(i_{max}, j) &= f_8(i_{max} - 1, j), \end{aligned} \tag{3.14}$$

or with a second-order extrapolation as

$$\begin{aligned} f_4(i_{max}, j) &= 2f_4(i_{max} - 1, j) - f_4(i_{max} - 2, j), \\ f_7(i_{max}, j) &= 2f_7(i_{max} - 1, j) - f_7(i_{max} - 2, j), \\ f_8(i_{max}, j) &= 2f_8(i_{max} - 1, j) - f_8(i_{max} - 2, j). \end{aligned} \tag{3.15}$$

In a similar fashion, the boudnaries at the west, north or south boundary can be treated.

3.7.4 Generalised Boundaries: The method of Zou and He

Most of the LB solver in use and developed today rely on the Boundary conditions of Zou and He. This method allows to treat open and solid boundaries in the same way. Without proof, the boundaries at the four different boundaries can be computed as

North:

$$\begin{aligned}
\rho_N &= \frac{1}{1+v_N}[f_1+f_2+f_4+2(f_3+f_7+f_6)],\\
f_5 &= f_3 - \frac{2}{3}\rho_N v_N,\\
f_8 &= f_6 - \frac{1}{2}(f_4-f_2) - \frac{1}{6}\rho_N v_N - \frac{1}{2}\rho_N u_N,\\
f_9 &= f_7 + \frac{1}{2}(f_4-f_2) - \frac{1}{6}\rho_N v_N + \frac{1}{2}\rho_N u_N.
\end{aligned} \tag{3.16}$$

South:

$$\begin{aligned}
\rho_S &= \frac{1}{1-v_S}[f_1+f_2+f_4+2(f_5+f_8+f_9)],\\
f_3 &= f_5 + \frac{2}{3}\rho_S v_S,\\
f_6 &= f_8 + \frac{1}{2}(f_4-f_2) + \frac{1}{6}\rho_S v_S + \frac{1}{2}\rho_S u_S,\\
f_7 &= f_9 - \frac{1}{2}(f_4-f_2) + \frac{1}{6}\rho_S v_S - \frac{1}{2}\rho_S u_S.
\end{aligned} \tag{3.17}$$

East:

$$\begin{aligned}
\rho_E &= \frac{1}{1+u_E}[f_1+f_3+f_5+2(f_2+f_6+f_9)],\\
f_4 &= f_2 - \frac{2}{3}\rho_E u_E,\\
f_8 &= f_6 - \frac{1}{2}(f_5-f_3) - \frac{1}{6}\rho_E u_E - \frac{1}{2}\rho_E v_E,\\
f_7 &= f_9 + \frac{1}{2}(f_5-f_3) - \frac{1}{6}\rho_E u_E + \frac{1}{2}\rho_E v_E.
\end{aligned} \tag{3.18}$$

West:

$$\begin{aligned}
\rho_W &= \frac{1}{1-u_W}[f_1+f_3+f_5+2(f_4+f_7+f_8)],\\
f_2 &= f_4 + \frac{2}{3}\rho_W u_W,\\
f_6 &= f_8 + \frac{1}{2}(f_5-f_3) + \frac{1}{6}\rho_W u_W + \frac{1}{2}\rho_W v_W,\\
f_9 &= f_7 - \frac{1}{2}(f_5-f_3) + \frac{1}{6}\rho_W u_W - \frac{1}{2}\rho_W v_W.
\end{aligned} \tag{3.19}$$

First, the density is calculated on the boundaries which depends on the perpendicular velocity component to that boundary. With the density, the missing DDF components can be computed. The values of u_l and v_l, $l = N, E, S, W$ are set by the user in the same way they would be imposed for a Navier–Stokes based simulation, i.e. these are the macroscopic velocity components at the

boundaries. In this way, inflow and moving walls can be simulated.
If one wishes to simulate an open boundary with unknown velocities, for example a velocity outlet, the usual way is to set the parallel velocity component to zero (to ensure a straight outflow) while the the perpendicular, or normal, velocity component is recovered from the density equation by solving it for the unknown velocity. For example, if we consider a channel flow with an outlet at the east boundary, the modified density boundary condition would become

$$u_E = \frac{[f_1 + f_3 + f_5 + 2(f_2 + f_6 + f_9)]}{\rho_E} - 1. \tag{3.20}$$

Now the density needs to be set which is which can be arbitrarily done. Usually a value of unity is employed. Thus, u_E is known and together with a straight outflow assumption ($v_E = 0$), the set of eq.(3.18) can be solved.

3.8 Converting between lattice and real units and the limits of the LBM

There are two ways to convert from lattice units to physical units. The first one involves finding a mapping of the lattice units directly to the physical units which is less commonly done. Thus, we concentrate on the second path which is done through dimensionless numbers.
The most important one is the Reynolds number, which is defined as

$$Re = \frac{n_c u_l}{\nu_l}. \tag{3.21}$$

Here, n_c is not the characteristic length, as is usually the case for defining the Reynolds number, but rather the number of cells along the characteristic length. The macroscopic, lattice velocity and viscosity are denoted by u_l and ν_l, respectively. We already mentioned that the lattice viscosity is user defined and has a lower limit for stability, which is due to the collision approximation. A simple BGK collision approximation may only allow the lattice viscosity to go as low as $\nu = 0.1$ while other collision approximation, most notably the multiple-relaxation time, or MRT, collision approximation may reduce that value even further. There is, however, also an upper limit on the lattice velocity. Examining any of the density formulations in eq.(3.16–3.19), one sees that the fraction of $1/(1 \pm \mathbf{u})$ may reach infinity (and thus an infinite density) at boundaries for values of $\mathbf{u}$ close to unity. In-fact, oscillation may occur much earlier and so to minimise boundary induced numerical artefacts it is suggested to keep $\mathbf{u} < 0.5$. As a simple example, consider a lid driven cavity with 100 cells in both directions with BGK collision approximation, on sees that to achieve a stable results, Reynolds numbers above $Re = 500$ are not feasible. In order to increase the Reynolds number, the number of cells need to be increased which has a direct effect on the computational speed but also the convergence rate as the timestep is equivalent to the mesh spacing.

Another important dimensionless number is the Mach number which may be computed using

$$Ma = \frac{1}{n_c\sqrt{3}}\left(\omega - \frac{1}{2}\right) Re. \tag{3.22}$$

Here, $1/\sqrt{3}$ is the lattice speed of sound and we see again the dependence of n_c on the Mach number. In general, the LB method is only valid for isothermal, low-speed flows up to a Mach number of $Ma < 0.1$. Sticking to the previous example of the lid-driven cavity, the Mach number with a lattice viscosity $\nu_l = 0.1$, 100 cells along the characteristic directions and a Reynolds number of $Re = 100$ yields a Mach number of $Ma = 0.43$ which is already rather high. In order to reduce the Mach number we need to increase the number of cells or increase the lattice viscosity, in either way, it will also have an effect on the Reynolds number.
This brief example shows that it is difficult to achieve high Reynolds number flows on standard computational meshes. However, the LBM method has been designed to be a mesoscopic approach which happens to work on the macroscopic scale as well. Using the method for industrial or engineering applications, however, needs a thourough review on the collision approximation, speed-models and the DDF at equilibrium, as the ones presented in this brief introduction to the LBM are only sufficient for low Re-number flows.