Simulation of 2-dimensional flows in Modelica with the Casacaded Digital Lattice Boltzmann Method

Thomas Bäuml Helmut Kühnelt
AIT Austrian Institute of Technology GmbH
Mobility Department, Electric Drive Technologies
Giefinggasse 2, 1210 Vienna, Austria

Keywords: Modelica, Cascaded Digital Lattice Boltzmann, 2-dimensional flows

Abstract

This paper deals with the implementation of a general methodology for modeling two-dimensional fluid flows in Modelica applying the Cascaded Digital Lattice Boltzmann Method. This approach models fluid flow as collective dynamics of fictitious particles on the nodes of a regular lattice. The various elements needed for simulation are described in Modelica and generic test cases are set up. The method is able to deal with simple scenarios where the powerful capabilities of advanced CFD tools are not needed.

1 Introduction

Calculating the dynamics of fluid flows is an important topic in the field of simulation. Common practice is to simulate complex scenarios by utilizing Computational Fluid Dynamics (CFD). Despite its capability of representing fluid flows in a very detailed way it has the drawback of compatibility. Coupling with other physical domain simulations is only possible by co-simulation. In this contribution a general methodology for modeling two-dimensional fluid flows in Modelica is shown. Whereas in [1] the Navier-Stokes equations are solved by a finite volume method, this work deals with modeling them with a Lattice Boltzmann Method (LBM).

2 Theory

The Lattice Boltzmann method is a relatively new simulation technique for fluid systems that has attracted interest as alternative to the discretization of the Navier-Stokes equations. Instead of discretizing the Navier-Stokes equations to solve the conservation equations of macroscopic quantities (i.e., mass, momentum, and energy), LBM is a mesoscopic approach for modeling macroscopic fluid dynamics based on the Boltzmann kinetic equation which describes the statistical behavior of a non-equilibrium thermodynamic system. In the LBM, the fluid motion is based on the collective dynamics of fictitious particles on the nodes of a regular lattice. The dynamics of these particles is designed to obey the basic conservation laws ensuring hydrodynamic behavior in the continuum limit. The basic quantity is the particle distribution function $f_i(\vec{x},t)$ that represents the probability of finding a fluid particle density i at a location $\vec{x}$ and at a time t traveling with a discrete speed $\vec{c}_i$.

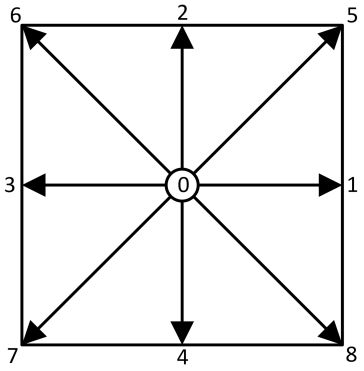

Figure 1: Lattice Boltzmann D2Q9 grid element

The mass density ρ and the momentum density $\rho\vec{v}$ are given by:

$$\rho(\vec{x},t) = \sum_{i=0}^{n} f_i(\vec{x},t) \tag{1}$$

$$\rho(\vec{x},t)\,\vec{v}(\vec{x},t) = \sum_{i=0}^{n} f_i(\vec{x},t)\,\vec{c}_i \tag{2}$$

The motion of the particles is restricted to the node positions of a regular lattice. In 2D, commonly a 9-speed, quadratic lattice (D2Q9) with mesh spacing Δx is applied, where the discrete velocities $\vec{c}_i$ connect lattice nodes to first and second neighbors and which has a rest particle f_0, see Figure 1. Here, $c_x = [0,1,0,-1,0,1,-1,-1,1]^T$ and $c_y = [0,0,1,0,-1,1,1,-1,-1]^T$.

The spatial and temporal evolution of the particle distribution function is described by an explicit discretization of the Boltzmann equation, given by the following equation:

$$f_i(\vec{x}+\vec{c}_i\Delta t, t+\Delta t) = f_i(\vec{x},t) - \omega\left(f_i(\vec{x},t) - f_i^{eq}(\vec{x},t)\right) \quad (3)$$

There, the left-hand side represents the molecular free streaming from one lattice node to the other, whereas the right-hand side represents the molecular collisions via a single-time relaxation towards local equilibrium f_i^{eq} on a typical timescale $\tau = 1/\omega$. τ is related to the macroscopic kinematic viscosity $\nu = c_s^2\Delta t\,(\tau - 1/2)$, where $c_s = 1/\sqrt{3}$ is the speed of sound. Commonly $\Delta t^{(LB)} = 1$ in lattice units, thus rendering the grid spacing $\Delta x^{(LB)} = 1$. Results in physical units can be obtained by applying the scaling $u^{(phys)} = u^{(LB)}\sqrt{3}c_s^{(phys)}$.

The local equilibrium is typically a second-order expansion in the fluid velocity of a local Maxwell distribution,

$$f_i^{eq} = w_i\left[\rho + 3\vec{c}_i\cdot\vec{v} - \frac{3}{2}\vec{v}^2 + \frac{9}{2}(\vec{c}_i\cdot\vec{v})^2\right], \quad (4)$$

where w_i is a set of weights normalized to unity. The single-relaxation-time (SRT) LBM, (3), recovers the weakly-compressible, athermal Navier-Stokes equations at low Mach numbers (Ma < 0.3) with second order accuracy in space and time.

Nevertheless, the SRT-LBM method shows instabilities when the viscosity is reduced to small values, in order to reach high Reynolds numbers at low Mach numbers. To enhance the stability, the multiple-relaxation-times (MRT) collision operator was proposed by [2, 3]. Instead of relaxing the particle distribution functions themselves towards equilibrium, as in the SRT-LBM, in the MRT-LBM, they are transformed from velocity space into the corresponding moment space, where the moments are relaxed towards their equilibrium values. The moment space of the D2Q9 model has nine velocity moments. The conserved moments are the density (1) and the flow momentum (2), the non-conserved moments include the energy, the stress tensor components, the energy square and the energy fluxes, for which different relaxation time scales are specified in order to decouple physical from higher order moments, thus improving the numerical stability. The post collision particle distributions f_i^{new} of the MRT-LBM are then given by the following expression:

$$f_i^{new} = f_i + M^{-1}S\left(m_i - m_i^{eq}\right), \quad (5)$$

where M is the orthogonal transformation matrix, $m_i = M f_i$ are the moments of the system and S is a diagonal matrix of the relaxation rates.

A specific MRT variant, adopted in this work, is the Cascaded-Digital-Lattice-Boltzmann (CDLB) algorithm [4], that allows virtually any viscosity value without loss of stability within the low Mach number limit. It adopts central moments in the reference frame moving with the macroscopic velocity and a generalized local equilibrium which is a function of both conserved and non-conserved hydrodynamic moments. The post collision distributions $\vec{f}^{new}$ of the CDLB are given by

$$\vec{f}^{new} = \vec{f}^{old} + K\cdot\vec{k} \quad (6)$$

where $\vec{k}$ is the CDLB collision term and K is the orthogonal transformation matrix, that maps the moments into velocity space.

$$K = \begin{bmatrix} 1 & 0 & 0 & -4 & 0 & 0 & 0 & 0 & 4 \\ 1 & 1 & 0 & -1 & 1 & 0 & 0 & 2 & -2 \\ 1 & 0 & 1 & -1 & -1 & 0 & 2 & 0 & -2 \\ 1 & -1 & 0 & -1 & 1 & 0 & 0 & -2 & -2 \\ 1 & 0 & -1 & -1 & -1 & 0 & -2 & 0 & -2 \\ 1 & 1 & 1 & 2 & 0 & -1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 2 & 0 & 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 2 & 0 & -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 2 & 0 & 1 & 1 & -1 & 1 \end{bmatrix} \quad (7)$$

As the collision term of the CDLB is rather complex, we refer to the original paper [4].

Using vector notation

$$K = \left[\vec{K}_0, \ldots, \vec{K}_8\right], \quad (8)$$

$$\vec{f} = \begin{pmatrix} f_0 \\ \vdots \\ f_8 \end{pmatrix}, \quad (9)$$

the conserved moments are expressed as

DOI 10.3384/ECP140961221

$$\begin{aligned} \rho &= \vec{f} \cdot \vec{K}_0, \\ \rho v_x &= \vec{f} \cdot \vec{K}_1, \\ \rho v_y &= \vec{f} \cdot \vec{K}_2. \end{aligned} \tag{10}$$

3 Implementation

Lattice Boltzmann collision equations are usually written in terms of post-collision distributions f_i^c, i.e., $f_i(\vec{x}+\vec{c}_i, t+1) = f_i^c(\vec{x}, t)$.
In this contribution a formulation where the pre-collision distribution is described in terms of the post-collision distribution at the respective neighbor node, i.e., $f_i(\vec{x}, t) = \mathbf{pre}(f_i(\vec{x}, t)) \equiv f_i^c(\vec{x}-\vec{c}_i, t-1)$ is used. The operator **`pre`** of Modelica makes this formulation convenient [5].

3.1 Node element

All elements extend from a basic node element, the partial node model `PrtlNode_D2Q9`. There, all parameters and variables like the transformation matrix, weighting parameters etc. are defined. Furthermore the internal particle distribution variables are defined and initialization values are calculated. Collision and streaming of particles is based on an equidistant time step which is realized by a clock signal

```
clock := sample(dt, dt);
```

where `dt` is the width of one time step, taken equal to one (in lattice units). The f_i are time discrete quantities changing their values only at event instants which are triggered by the clock signal.
Each grid element has a rest particle and eight particles that are streamed to the first (horizontal and vertical) and second (diagonal) neighbors of the element. To link the elements, two kinds of connectors are implemented, a forward connector `f_fwd` and a backward connector `f_bwd`. Each consists of a Real input variable and a Real output variable to match its counterpart.

```
connector f_fwd
  input Real f_n;
  output Real f_p;
end f_fwd;
```

```
connector f_bwd
  output Real f_n;
  input Real f_p;
end f_bwd;
```

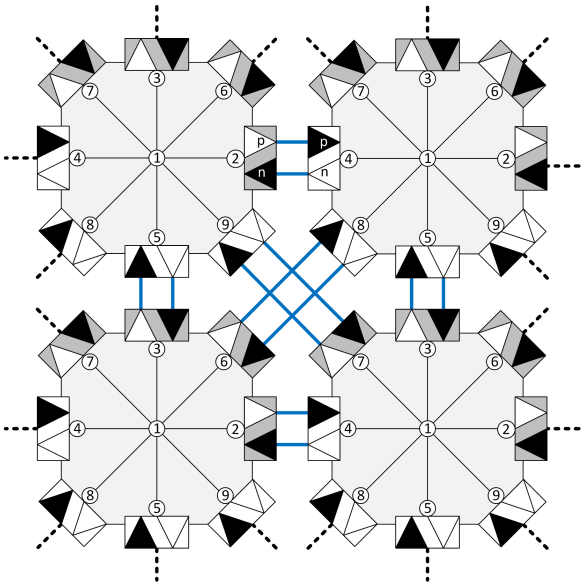

Figure 2: Schematic grid of four D2Q9 node elements including connections

Connectors are placed on the element models facing in all eight streaming directions. Connectors 2, 3, 6 and 7 are facing forwards and connectors 4, 5, 8 and 9 are facing backwards.
Various kinds of node elements extend from this partial model. They are explained in more detail in section 4.

3.2 Collision and streaming step

At every time step and at each grid element, the particle distributions are received, collided and propagated. The receiving and propagating step are generally known as streaming.
In a conventionally implemented LBM, streaming affects only ports of the same direction. This means, a particle distribution with velocity $\vec{c}_2$ exits port 2 and enters port 2 of the adjacent element. In Modelica, due to the connector concept, the case is slightly different. Port 2 of a grid element is connected to port 4 of the adjacent element, port 4 is connected to port 2 of the next element, and so on. To establish a correct streaming behavior the input, which holds the post-collision value at the last time step, must be mapped to the respective output, e.g., port 4 has to be mapped to port 2, port 2 has to be mapped to port 4. Then the particle distributions are collided. Then the post-collision values are written into the output variable of the respective connector. Propagation to the neighboring grid elements is done automatically by the connector, no additional commands are needed.

```
when clock then
  // mapping
  fold[1] := pre(f1);
  fold[2] := pre(f4.f_p);
  fold[3] := pre(f5.f_p);
  fold[4] := pre(f2.f_n);
  ...

  // collision
  fnew := ... fold;

  // output
  f1     := fnew[1];
  f2.f_p := fnew[2];
  f3.f_p := fnew[3];
  f4.f_n := fnew[4];
  ...
end when;
```

3.3 Mesh and connections – setting up the computational domain

In the example model, the 2D-flow model has to be described and the computational domain set up. Each model consists of sources, fluid nodes and boundary conditions. Two ways are possible to build up the model. The first is to build the model by dragging and dropping elements to the workspace and drawing connections by hand. Because the number of elements may be quite high and every element needs eight connects to its neighbours, the effort to set the model up like this is quite high. A more convenient method is proposed here. Providing the matrix `nodeType` that represents the LB discretized computational domain, all connections are generated automatically via nested loops.

The matrix can easily be set up in e.g. Microsoft Excel and then imported to the simulation example. A simple example of a two-dimensional duct model is shown below.

```
parameter Integer nodeType[:,:]=
          {{2,2,2,2,2,2,2,2,2},
           {3,1,1,1,1,1,1,1,4},
           {3,1,1,1,1,1,1,1,4},
           {3,1,1,1,1,1,1,1,4},
           {3,1,1,1,1,1,1,1,4},
           {2,2,2,2,2,2,2,2,2}};
...
CDLB.D2Q9 node[:,:](nodeType=nodeType,...);
```

The parameter `nodeType` is then propagated to the element `CDLB.D2Q9` which acts as generalized element representing all node types in conditional definition.

```
model D2Q9
  ...
  CDLB.FluidNode      fn if nodeType == 1;
  CDLB.BounceBackNode bn if nodeType == 2;
  CDLB.VelocityNode   vn if nodeType == 3;
  CDLB.DensityNode    dn if nodeType == 4;
  ...
end model;
```

The user only has to define the matrix, all connections are established automatically. They are defined in multiple loops to interconnect every element with its eight neighbors. As example, the connections for connector 4 are outlined here:

```
// connect row 2:end and col 2:end
for i in 2:1:nrow loop
  for j in 2:1:ncol loop
    connect(node[i,j].f4,node[i,j-1].f2);
  end for;
end for;

// connect row 1 and col 2:end
for j in 2:1:ncol loop
  connect(node[1,j].f4,node[1,j-1].f2);
end for;

// connect col 1 to col end
for i in 1:1:nrow loop
  connect(node[i,1].f4,node[i,end].f2);
end for;
```

These equations are repeated for all other connectors and are omitted here for sake of brevity.

4 Elements

4.1 FluidElement

The fluid element implements the collision, (6). To speedup symbolic pre-processing and compilation time, this is encapsulated in a function. To avoid reflections at the in- and outflow boundaries, sponge zones are implemented. There the relaxation factor τ is gradually increased to 1, thus driving the fluid towards its equilibrium state at the boundary.

4.2 BoundaryElement

To implement a solid, non-slip boundary condition a local bounce-back rule is applied on the solid node $\vec{x}_s$:

$$f^c_{i'}(\vec{x}_s,t) = f_i(\vec{x}_s,t) \tag{11}$$

where i' denotes the link with reversed velocity of i, i.e., $\vec{c}_{i'} = \vec{c}_i$, pointing into the fluid. Incoming distributions functions at a wall node are reflected back to the original fluid nodes, with the direction rotated by π. The "bounce-back on the node" is purely local, thus being implementable in the current concept, but it has

DOI 10.3384/ECP140961221

been proven to be only first-order accurate in time and space.

Shifting the solid wall half-way between the two nodes, leads to the "bounce-back on the link" which is of second order accuracy:

$$f_{i'}(\vec{x}_l, t+1) = f_i^c(\vec{x}_l, t) \tag{12}$$

where $\vec{x}_l$ is a fluid node next to the solid boundary and f^c is the post-collision value before propagation. Unfortunately it is not implementable in the current context, as each node element in Modelica cannot access its neighbors.

4.3 DirichletElement

Dirichlet boundary conditions can be set up based on the idea of bounce-back of the non-equilibrium part, as proposed in [6]. As an example, at a flow boundary having a normal vector into the fluid in positive x-direction, i.e., f_2, f_6, f_9 pointing into the fluid, these distribution functions are unknown after streaming. Equations (1) and (2) can be used to reconstruct the unknown distributions.

At a velocity boundary node, after streaming $f_1, f_3, f_4, f_5, f_7, f_8$ are known and v_x, v_y are specified. f_2, f_6, f_9 and ρ need to be determined. Equations (1) and (2) yield three equations. In order to close the system, it is assumed that it is admissible to bounce-back of the non-equilibrium part of the particle distribution normal to the boundary, i.e., $f_{i'}^{neq} = f_i^{neq} \equiv f_{i'} - f_{i'}^{eq} = f_i - f_i^{eq}$. For a Dirichlet element pointing in positive x-direction this gives

```
rho := 1/(1-vx) * ((fo[1]+fo[3]+fo[5]) +
             + 2 * (fo[4]+fo[7]+fo[8]));

fo[2] := fo[4] + 2/3*rho*vx;
fo[6] := fo[8] + 1/2*(fo[5] - fo[3])
               + 1/2*rho*vy + 1/6*rho*vx;
fo[9] := fo[7] + 1/2*(fo[3] - fo[5])
               - 1/2*rho*vy + 1/6*rho*vx;
```

For a known inlet velocity $v_{in,x}$, this system serves as velocity inlet, `CDLB.VelocityNode`, or can be rearranged in terms of known density, $\rho = \rho_{in}$, into a density inlet, `CDLB.DensityNode`.

4.4 Initial Conditions

At start of the simulation run, a flow at rest is assumed, setting the distribution functions to their equilibrium value.

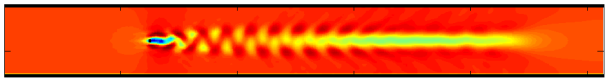

Figure 3: Profile of the magnitude of the flow velocity in lattice units in a fluid domain with one solid node at one quarter of the domain length and vertically centered. A periodic vortex occurs forming a von Karman vortex.

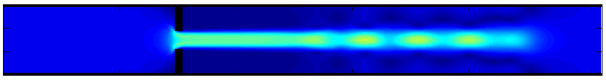

Figure 4: Profile of the magnitude of the flow velocity in lattice units in a fluid domain with an orifice at one quarter of the domain length. A jet is created which is amplified by aerodynamic effects.

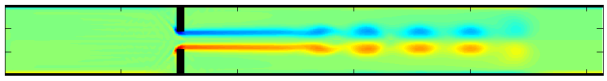

Figure 5: Profile of the vorticity in lattice units in a fluid domain with an orifice at one quarter of the domain length. A jet is created which is amplified by aerodynamic effects.

5 Test cases and results

5.1 Flow past a cylinder

The first test case deals with the flow around a cylinder in a fluid stream, which is formed by one solid node. It is positioned at approximately one quarter of the domain length and vertically centered in the computational domain. The nodes at the upper and lower boundaries are also fluid nodes. As all boundaries are interconnected with each other, this constitutes periodic flow boundary conditions. The computational domain consists of 258 x 30 grid nodes, leading to 570k equations, which is close to the maximum size manageable in Dymola.

Periodic vortex at the cylinder occurs forming a von Karman vortex street, see Figure 3 for a snapshot showing the velocity magnitude in lattice units. In order to obtain a high Reynolds number, the relaxation factor τ was set to $1/2$, yielding a very low kinematic viscosity, which is only determined by numerical precision of the solver.

5.2 Flow through an orifice

The second test case is a flow through an orifice where a jet is formed. The orifice is modeled as solid with

non-slip walls and is positioned in a duct with slip-walls at one quarter of the domain length. The computational domain consists of 258 x 30 grid nodes, leading to 570k equations.
Figures 4 and 5 show snapshots of the velocity magnitude and the vorticity in lattice units. Velocity perturbations at the flow inlet trigger initial perturbations in the jet shear layers which are further amplified by aerodynamic effects due to the Kelvin-Helmholtz instabilities until the jet breaks up into discrete vortices.

6 Conclusions

An approach for simulating fluid flow with the Cascaded Digital Lattice Boltzmann method is proposed. With this approach fluid flow problems can be addressed in a multi physical modeling language like Modelica. The method works for small scenarios but reaches the boundaries of efficient simulation quickly. Nevertheless, the approach works for 2D flows and can be extended to 3D flows easily.

References

[1] M. Bonvini, M.; Popovac, "Fluid flow modelling with Modelica," *Proceedings of the 7th International Conference on Mathematical Modelling, Vienna, Austria*, 2012.

[2] D. D'Humieres, "Generalized lattice-Boltzmann equations," *Rarefied Gas Dynamics: Theory and Simulations*, vol. 159, pp. 450–458, 1992.

[3] P. Lallemand and L.-S. Luo, "Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability," *Phys. Rev. E*, vol. 61, pp. 6546–6562, Jun 2000.

[4] M. Geier, J. G. Korvink, and A. Greiner, "Cascaded digital lattice Boltzmann automata for high Reynolds number flow," *PHYSICAL REVIEW E 73*, 2006.

[5] J. Brown, "Computational fluid dynamics in an equation-based, acausal modeling environment," Master's thesis, Atlanta, Ga. : Georgia Institute of Technology, 2010.

[6] Q. Zou and X. He, "On pressure and velocity boundary conditions for the lattice Boltzmann BGK model," *Physics of Fluids*, vol. 9, no. 6, pp. 1591–1598, Jun. 1997.

DOI 10.3384/ECP140961221