Commun. Comput. Phys.
doi: 10.4208/cicp.210910.200611a

Vol. **12**, No. 1, pp. 65-84
July 2012

A Coupled Approach for Fluid Dynamic Problems Using the PDE Framework Peano

Philipp Neumann*, Hans-Joachim Bungartz, Miriam Mehl, Tobias Neckel and Tobias Weinzierl

Chair of Scientific Computing, Faculty of Informatics, Technische Universität München, Boltzmannstr. 3, 81545 Garching, Germany.

Received 21 September 2010; Accepted (in revised version) 20 June 2011

Available online 19 January 2012

Abstract. We couple different flow models, i.e. a finite element solver for the Navier-Stokes equations and a Lattice Boltzmann automaton, using the framework Peano as a common base. The new coupling strategy between the meso- and macroscopic solver is presented and validated in a 2D channel flow scenario. The results are in good agreement with theory and results obtained in similar works by Latt et al. In addition, the test scenarios show an improved stability of the coupled method compared to pure Lattice Boltzmann simulations.

AMS subject classifications: 35Q30, 35Q35, 76M10, 76-xx

Key words: Spatial coupling, Lattice Boltzmann, Navier-Stokes, Peano.

1 Introduction

Many flow systems – especially in micro- and nano-fluidics – are strongly influenced by physical processes that appear on different spatial and temporal scales. Flows through nanopores influenced by Brownian motion or flows in porous media are typical examples. Solving these kinds of complex systems by means of computational fluid dynamics (CFD) often requires highly adaptive concepts or different sorts of solvers depending on the current scale to be simulated.

With respect to Navier-Stokes and Lattice Boltzmann related solver techniques, detailed discussions and comparisons of their performance and qualitative behaviour can amongst others be found in [8, 13] and should only be touched here. Considering flow problems on mesoscopic scales, Lattice Boltzmann methods have been validated for a wide range of scenarios such as particulate flows [10] or fluctuating hydrodynamics

*Corresponding author. *Email address:* neumanph@in.tum.de (P. Neumann)

[7]. Furthermore, as discussed in our previous works [13], simulation tests by different groups in the finite Knudsen number regime revealed strong deviations of macroscopic Navier-Stokes and Burnett descriptions from the physical phenomena (cf. [17,18]) whereas mesoscopic approaches like Direct Simulation Monte Carlo and Lattice Boltzmann methods still yielded feasible results [9, 21]. On the other hand, Navier-Stokes systems might e.g. be superior in case of (laminar) convection dominated flows where classical Lattice Boltzmann automata often require even finer grid resolutions and small timesteps. Implicit solver strategies for the Navier-Stokes equations would allow for considerably bigger timesteps in these cases.

With implementations of a Navier-Stokes and a Lattice Boltzmann solver integrated within the Peano framework, we already deeply discussed and evaluated the single codes in [13]; it is in this context that the idea emerged to spatially couple both approaches and exploit the advantages of both schemes as also proposed by Latt et al. in [12]. There are different scenarios, where this idea might become favorable. Amongst others, considering fluid-structure interaction problems in the laminar flow regime, the weakly compressible Lattice Boltzmann method could be used to stabilize pressure induced perturbations in the surrounding of the respective structure, whereas the Navier-Stokes equations are solved further away from the structure. Besides, in the case of simulations of multiscale flow problems, like laminar flows near and through porous structures such as membranes, a spatial coupling of Navier-Stokes and Lattice Boltzmann solvers might allow to resolve the complex geometry of the porous medium and describe this part of the flow system by means of Lattice Boltzmann and – on a coarser grid – to compute the rest of the flow domain efficiently by (implicit) Navier-Stokes solver techniques.

There have already been several works on this field (see for example [1,12]). It is particularly the work presented in [12] where the coupling between a finite difference Navier-Stokes solver and a Lattice Boltzmann automaton is established and validated. Though parts of the respective algorithmic realisations carry over to our coupling approach, we want to focus on a new method for the macro-to-mesoscale coupling which is based on a minimisation procedure. We particularly point out the feasibility of coupling different types of solvers within the Peano framework.

This paper is structured as follows: The theoretical foundation for the Lattice Boltzmann method and our Navier-Stokes solver is briefly reviewed in Section 2. Besides, the Chapman-Enskog expansion connecting the meso- and macroscopic approach is carried out in order to obtain the conservation laws that need to be fulfilled at the interface between the two solvers. In Section 3, we give a short introduction to the framework Peano and discuss its basic properties at the example of our Lattice Boltzmann implementation. In Section 4, we discuss the underlying grid topology and afterwards focus on the methodology for coupling Navier-Stokes to Lattice Boltzmann and vice versa. A description of the underlying implementations within the Peano framework is given in Section 5. Results for the coupling applied to pure channel flow are provided in Section 6. Finally, we close the discussion and give a short conclusion and outlook on future work in Section 7.

2 Macro- and mesoscopic simulation methods

2.1 Navier-Stokes equations

The common macroscopic description of flow problems is given by the Navier-Stokes equations describing the conservation of mass and momentum for an incompressible fluid:

$$\nabla \cdot u = 0, \tag{2.1}$$

$$\partial_t u + (u \cdot \nabla) u = -\nabla p + \nu \Delta u, \tag{2.2}$$

where $u \in \mathbb{R}^D$ denotes the flow velocity, $p \in \mathbb{R}$ is the pressure and ν defines the kinematic viscosity. A finite element approach to solve this system of partial differential equations is implemented in Peano [14]. It uses d-linear elements in space and supports different timestepping schemes such as explicit/ implicit Euler or Crank-Nicholson. However, as Lattice Boltzmann is a pure explicit scheme, only the explicit Euler timestepping is to be used. The solving of the Navier-Stokes system is accomplished by a Chorin projection leading to a Poisson equation for the pressure that needs to be solved in every timestep dt, such that each timestep involves three steps:

- Compute the right hand side of the pressure Poisson equation from velocity values at time t.
- Solve the Poisson equation to obtain pressure values $p(t+dt)$.
- Update the velocity field using the old velocity values $u(t)$ and the new pressure values $p(t+dt)$.

The boundary conditions for the system (Dirichlet or Neumann conditions) can be set directly via the velocity values and the assembling method for the right hand side of the Poisson equation; see [14] for more details.

2.2 Lattice Boltzmann method

A mesoscopic approach to computational fluid dynamics is derived from the Boltzmann equation and is known as the Lattice Boltzmann method. Detailed descriptions of the method can amongst others be found in [16, 20]; a short introduction to its principles is provided in the following.

Space is discretised with cubic cells, and the velocity space is broken down to a minimal isotropic set of discrete velocities $c_i \in \mathbb{R}^D$, $i \in \{1, \cdots, Q\}$, allowing fluid molecules to either move to a neighbouring cell or to rest in the current one. The update rule to determine the probability $f_i(x,t)$ to find fluid molecules at position $x \in \mathbb{R}^D$ at time t moving with velocity c_i reads

$$f_i(x + c_i dt, t + dt) = f_i(x,t) + \Delta_i (f - f^{eq}), \tag{2.3}$$

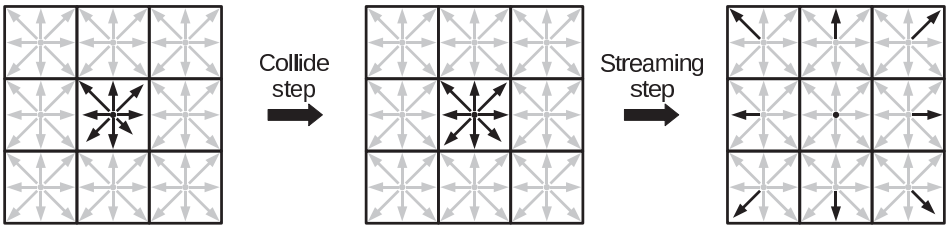

Figure 1: The Lattice Boltzmann algorithm for the D2Q9 model. In the collide step, the post-collision distributions $f_i^*(x,t)=f_i(x,t)+\Delta_i(f-f^{eq})$ are computed locally in each cell. In the streaming step, the distribution functions are moved along their respective lattice velocities c_i to their neighbours, i.e. $f_i(x+c_i dt,t+dt)=f_i^*(x,t)$.

where f^{eq} denotes the discrete representation of the equilibrium distribution,

$$f_i^{eq}(\rho,u)=w_i\rho\left(1+\frac{c_i\cdot u}{c_s^2}+\frac{(c_i\cdot u)^2}{2c_s^4}-\frac{u^2}{2c_s^2}\right) \tag{2.4}$$

with lattice weights w_i, the speed of sound $c_s=\frac{1}{\sqrt{3}}\frac{dx}{dt}$ on the lattice and fluid density $\rho(x,t)\in\mathbb{R}$ and velocity $u(x,t)\in\mathbb{R}^D$ computed from:

$$\begin{aligned}&\rho(x,t)=\sum f_i,\\&u_\alpha(x,t)=\frac{1}{\rho(x,t)}\sum f_i c_{i_\alpha}\quad\text{for all }\alpha\in\{1,\cdots,D\}.\end{aligned} \tag{2.5}$$

For the D2Q9 discretisation (two-dimensional lattice with nine velocities, see Fig. 1), the weights w_i are given by

$$w_i=\begin{cases}1/36, & \|c_i\|_2=\sqrt{2}\,dx,\\ 1/9, & \text{if}\quad \|c_i\|_2=dx,\\ 4/9, & \|c_i\|_2=0.\end{cases} \tag{2.6}$$

The operator $\Delta(f-f^{eq})$ is called collision operator and models intermolecular collision processes between the particle distribution functions f_i in each cell. For systems near equilibrium, collisions are taken into account via a linear relaxation of the particle populations towards the local equilibrium. For this purpose, the single relaxation time scheme which is also known as BGK model [4] can be applied:

$$\Delta_i(f-f^{eq})=-\frac{1}{\tau}\left(f_i-f_i^{eq}\right), \tag{2.7}$$

where τ is the relaxation time. For stability reasons, τ is restricted to the interval $(0.5,2)$.

Boundary conditions for the system are applied by reconstructing the missing distribution functions which would have entered the domain from outside. A typical example is given in form of the half-way bounce back rule modeling no-slip boundaries (see e.g. [16]).

2.3 Chapman-Enskog theory

In order to connect the meso- and macroscales, the Chapman-Enskog expansion [6] can be used to recover the Navier-Stokes equations from the Lattice Boltzmann equation in the continuum limit, that is in case that the Knudsen number

$$\epsilon := \frac{L}{L_H} \tag{2.8}$$

vanishes. In Eq. (2.8), L and L_H denote the length scales of the mesoscopic and the hydrodynamic (macroscopic) system. As the coupling of Navier-Stokes and Lattice Boltzmann systems strongly depends on the relation between macro- and mesoscopic quantities, the basic steps of the asymptotic analysis are carried out in the following.

The spatial coordinate x_H of the macroscopic description in hydrodynamics can be related linearly to its mesoscopic representation x which is used in the Lattice Boltzmann scheme:

$$x_H := \epsilon x. \tag{2.9}$$

In order to capture both convective and diffusive phenomena appearing on different temporal scales, the hydrodynamic times t_C and t_D are introduced and can be related to the Lattice Boltzmann time t as follows:

$$\begin{aligned} t_C &:= \epsilon t, \\ t_D &:= \epsilon^2 t. \end{aligned} \tag{2.10}$$

In a first step, the left hand side of Eq. (2.3) is expanded in a Taylor series around (x,t):

$$\begin{aligned} f_i(x+c_i dt, t+dt) = f_i(x,t) &+ \sum_\alpha c_{i_\alpha} dt \partial_{x_\alpha} f_i(x,t) + dt \partial_t f_i(x,t) + \frac{dt^2}{2} \sum_{\alpha,\beta} c_{i_\alpha} c_{i_\beta} \partial_{x_\alpha} \partial_{x_\beta} f_i(x,t) \\ &+ dt^2 \sum_\alpha \partial_{x_\alpha} \partial_t f_i(x,t) + \frac{dt^2}{2} \partial_t^2 f_i(x,t) + \mathcal{O}(dt^3). \end{aligned} \tag{2.11}$$

In addition, the distribution functions f_i are expanded in an asymptotic series near equilibrium:

$$f_i = f_i^{eq} + \epsilon f_i^{(1)} + \mathcal{O}(\epsilon^2) \tag{2.12}$$

with the non-equilibrium part $f_i^{neq} = \epsilon f_i^{(1)}$. As the density and momentum do not differ between the equilibrium and non-equilibrium state (see Eq. (2.5)), the non-equilibrium contributions vanish:

$$\sum_i f_i^{neq} = 0, \tag{2.13}$$

$$\sum_i f_i^{neq} c_{i_\alpha} = 0 \quad \text{for all } \alpha \in \{1, \cdots, D\}. \tag{2.14}$$

The right hand side of Eq. (2.3) is expanded analogously to Eq. (2.12):

$$\begin{aligned} &f_i+\Delta_i(f-f^{eq})\\ &=f_i^{eq}+\Delta_i^{(0)}(f-f^{eq})+\epsilon\left(f_i^{(1)}+\Delta_i^{(1)}(f-f^{eq})\right)+\epsilon^2\Delta_i^{(2)}(f-f^{eq})+\mathcal{O}(\epsilon^3). \end{aligned} \tag{2.15}$$

Introducing the macroscopic scaling for space and time from Eqs. (2.9) and (2.10) in Eq. (2.11), i.e.

$$\begin{aligned} &f_i(x,t)=\tilde{f}_i(x_H(x),t_C(t),t_D(t)),\\ &\partial_t=\epsilon\partial_{t_C}+\epsilon^2\partial_{t_D},\\ &\partial_{x_\alpha}=\epsilon\partial_{x_{H_\alpha}}\quad\text{for all } \alpha\in\{1,\cdots,D\}, \end{aligned} \tag{2.16}$$

one obtains:

$$\begin{aligned} f_i(x+c_idt,t+dt)=\tilde{f}_i^{eq}&+\epsilon\left(\tilde{f}_i^{(1)}+\sum_\alpha c_{i_\alpha}dt\partial_{x_{H_\alpha}}\tilde{f}_i^{eq}+dt\partial_{t_C}\tilde{f}_i^{eq}\right)\\ &+\epsilon^2\left(\sum_\alpha c_{i_\alpha}dt\partial_{x_{H_\alpha}}\tilde{f}_i^{(1)}+dt\partial_{t_D}\tilde{f}_i^{eq}+dt\partial_{t_C}\tilde{f}_i^{(1)}+\frac{dt^2}{2}\partial_{t_C}^2\tilde{f}_i^{eq}\right.\\ &\left.+\frac{dt^2}{2}\sum_{\alpha,\beta}c_{i_\alpha}c_{i_\beta}\partial_{x_{H_\alpha}}\partial_{x_{H_\beta}}\tilde{f}_i^{eq}+dt^2\sum_\alpha c_{i_\alpha}\partial_{x_{H_\alpha}}\partial_{t_C}\tilde{f}_i^{eq}\right)+\mathcal{O}(\epsilon^3). \end{aligned} \tag{2.17}$$

From the asymptotic theory, it follows that all terms in Eq. (2.17) and (2.15) of same order in ϵ have to be equal. For the terms of order ϵ^0, ϵ^1 and ϵ^2, this implies:

$$\Delta_i^{(0)}=0, \tag{2.18}$$

$$\left(\partial_{t_C}+\sum_\alpha c_{i_\alpha}\partial_{x_{H_\alpha}}\right)\tilde{f}_i^{eq}=\frac{1}{dt}\Delta_i^{(1)}(f-f^{eq}), \tag{2.19}$$

$$\partial_{t_D}\tilde{f}_i^{eq}+\frac{1}{2}\left(\partial_{t_C}+\sum_\alpha c_{i_\alpha}\partial_{x_{H_\alpha}}\right)\left(2\tilde{f}_i^{(1)}+\Delta_i^{(1)}\right)=\frac{1}{dt}\Delta_i^{(2)}(f-f^{eq}). \tag{2.20}$$

The moments of zeroth and first order of Eqs. (2.19) and (2.20) are determined from a multiplication of the equations with the factors 1 and c_{i_β}, $\beta=1,\cdots,D$, respectively, and integrating them over the velocity space, resembling a summation over i. The equations evolving are

$$\partial_{t_C}\tilde{\rho}+\sum_\alpha\partial_{x_{H_\alpha}}(\tilde{\rho}\tilde{u}_\alpha)=0, \tag{2.21}$$

$$\partial_{t_D}\tilde{\rho}=0 \tag{2.22}$$

for the moment of order zero and

$$\partial_{t_C}(\tilde{\rho}\tilde{u}_\beta)+\sum_\alpha \partial_{x_{H_\alpha}} \sum_i \left(\tilde{f}_i^{eq} c_{i_\alpha} c_{i_\beta}\right)=0, \tag{2.23}$$

$$\partial_{t_D}(\tilde{\rho}\tilde{u}_\beta)+\frac{1}{2}\sum_\alpha \partial_{x_{H_\alpha}} \sum_i \left(2\tilde{f}_i^{(1)}+\Delta_i^{(1)}\right) c_{i_\alpha} c_{i_\beta}=0 \tag{2.24}$$

for the moment of order one. The macroscopic Navier-Stokes equations for mass and momentum conservation are obtained from Eqs. (2.21), (2.22) and Eqs. (2.23), (2.24) by rescaling the equations by factors ϵ, ϵ^2, adding them up and resubstituting the original variables x and t for space and time:

$$\partial_t \rho+\sum_\alpha \partial_{x_\alpha}(\rho u_\alpha)=0, \tag{2.25}$$

$$\partial_t(\rho u_\beta)+\sum_\alpha \partial_{x_\alpha}(\rho u_\alpha u_\beta)=-\partial_{x_\beta} p-\frac{1}{2}\sum_\alpha \partial_{x_\alpha}\left(\sum_i (2f_i^{neq}+\Delta_i(f^{neq}))c_{i_\alpha}c_{i_\beta}\right). \tag{2.26}$$

Note that for the latter equation, the equalities $\sum_i f_i^{eq} c_{i_\alpha} c_{i_\beta}=p\delta_{\alpha\beta}+\rho u_\alpha u_\beta$ for the Eulerian stress and the equation of state

$$p=c_s^2\rho \tag{2.27}$$

for the pressure were used.

In order to make the non-equilibrium and collision contributions fit the viscous stress tensor from the Navier-Stokes system, that is

$$-\frac{1}{2}\sum_i (2f_i^{neq}+\Delta_i(f^{neq}))c_{i_\alpha}c_{i_\beta}=\nu\left(\partial_{x_\beta}u_\alpha+\partial_{x_\alpha}u_\beta\right)+\mathcal{O}(u^3), \tag{2.28}$$

the kinematic viscosity ν and the relaxation time τ need to fulfill

$$\nu=\rho c_s^2 dt\left(\tau-\frac{1}{2}\right). \tag{2.29}$$

3 Peano in a nutshell

While details on technical and application-specific aspects of the PDE framework Peano are available in [5, 14, 19], we shortly describe the basic ideas in this section.

The concept of Peano is to create adaptive Cartesian grids via a spacetree approach. This is similar to adaptive mesh refinement (AMR) (cf. [2, 3]) but provides a full grid hierarchy in the complete domain.

The grid is constructed starting from a hypercube root cell embedding the whole computational domain. The root cell is then subdivided in a recursive manner by splitting up each cell into three parts along each coordinate axis. Hence, adaptive grids for (complicated) geometries can be constructed easily and efficiently.

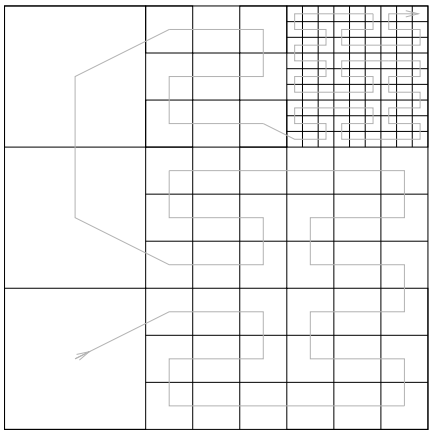

Figure 2: Example of a 2D discrete Peano curve on a corresponding adaptive Cartesian grid.

To serialise the underlying spacetree of cells for a traversal of the grid, a certain order is necessary. There are a variety of possibilities, but some of them have certain advantages for our concept.

The so called space-filling curves (cf. [15]) offer, besides good dynamic load balancing properties in parallel computations, additional features concerning an efficient cache usage in modern computer architectures as well as advantages constructing dynamically changing grids.

These self-similar curves are recursively defined and fit perfectly to our construction of the spacetree. Therefore, the Peano curve as one representant of space-filling curves, is used to iterate over the corresponding grid (and actually gave the name to our PDE framework). In Fig. 2, we exemplarily show a discrete iterative of the two-dimensional Peano curve yielding an adaptive Cartesian grid with three refinement levels. Using the Peano curve for the grid traversal implies a very high data locality. Combining this feature with a stack concept for the storage of the grid vertices (cf. [19]) allows for a very cache-efficient traversal mechanism.

In order to allow different types of solvers to use this functionality, the grid generation and data handling is hidden from the specific application. Each solver only requires to provide the structure of its degrees of freedom (i.e. defining the number of degrees of freedom and specify whether they are stored within a grid vertex or a grid cell) and the respective solver routines; the Peano grid traversal of all data is then carried out automatically.

With the traversal bound to the discrete representation of the Peano curve, the cell and vertex data are accessed in a specific order. In order to plug in solver specific functionalities such as local timestepping, spatial stencil evaluations etc., these solver routines need to be implemented by the use of adapters. Each adapter consists of several callback methods involving the access to cells and vertices. Traversing the grid, these methods are automatically triggered by the traversal mechanism of the framework. Examples for the

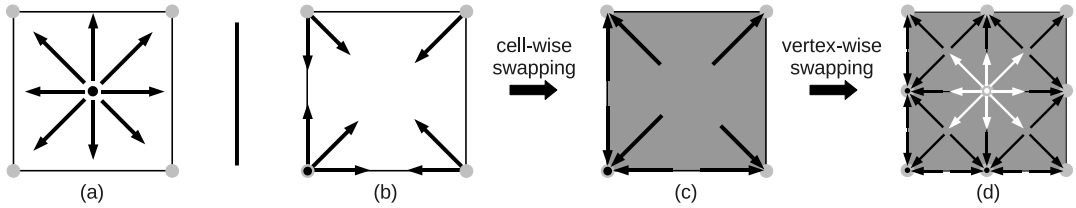

Figure 3: Storage scheme and streaming of the particle distribution functions. Fig. (a) shows the logic representation of the distributions (with respect to their associated lattice velocities), the Figs. (b)-(d) describe the algorithmic realisation of the streaming step. The distributions are initially stored on the vertices as shown in Fig. (b). In the enterElement(Cell, Vertex[2^D])-callback, the distributions are – after finishing the collision step – swapped within the frame of a single grid cell (Figs. (b)-(c)). The dark grey colour of the cell in Fig. (c) indicates that the enterElement-callback has returned. After visiting all adjacent cells of a vertex, the distributions stored on this vertex are again swapped locally so that the distribution functions for the next timestep are located at their new initial positions (Fig. (d)). The latter step is implemented within the touchVertexLastTime(Vertex)-callback.

callback functions include:

- touchVertexFirstTime(Vertex): This callback is triggered the very first time that a vertex is read from an input stack. As a consequence, this vertex has not been involved in any solver operations before.
- enterElement(Cell, Vertex[2^D]): This callback is triggered by the traversal mechanism each time when a grid cell is visited; as the cells are traversed along the Peano curve, this function is called exactly once for each grid cell during a single traversal.
- touchVertexLastTime(Vertex): This callback is triggered right before the vertex is written to an output stack. The vertex will not be involved in any other solver operations afterwards.

As an example, the storage and processing scheme for the particle distribution functions of the Lattice Boltzmann automaton within the framework [13] is reviewed in Fig. 3. Though evaluated within the center of a grid cell, the particle distribution functions f_i are stored on the grid vertices in a staggered manner (see Figs. 3 (a) and (b)). We already described the A-B pattern streaming approach in [13] involving two complete representations of the distribution functions and copy operations in the streaming step. As we briefly mentioned the A-A pattern [11] as a memory-efficient alternative before [13], we now consider its implementation within Peano using the adapter concept mentioned above. The scheme is depicted in Figs. 3 (b)-(d). After doing the local collision in the enterElement(Cell, Vertex[2^D])-callback, the distribution functions of opposite lattice velocities are swapped within the frame of a single cell (cell-wise swapping). When 2^D cells have finished the streaming, the vertex lying in the center between these cells already contains the new distributions for the next timestep. However, the distributions have to be swapped again as they are still located on a part of the vertex memory that is logically related to their former cell. This vertex-wise swapping is implemented within the touchVertexLastTime(Vertex)-callback.

4 Lattice Boltzmann–Navier-Stokes coupling

Having reviewed the Lattice Boltzmann and the Navier-Stokes approach as well as the basic principles of the Peano, we now discuss our coupling approach of the two methods within the framework.

The coupling of Lattice Boltzmann and Navier-Stokes systems requires the mapping of the respective unknowns

$$f_i, \rho^{LB}, u^{LB} \leftrightarrow p^{NS}, u^{NS}.$$

As the reconstruction of one set of these variables necessitates interpolation depending on the structural pattern of the underlying grids, the setup and positioning of the degrees of freedom for both fluid solvers is pointed out in Section 4.1. Subsequently, in Sections 4.2 and 4.3, the mappings $f_i, \rho^{LB}, u^{LB} \rightarrow p^{NS}, u^{NS}$ and $p^{NS}, u^{NS} \rightarrow f_i, \rho^{LB}, u^{LB}$ are described in detail.

4.1 Grid topology

In order to explain our coupling strategy, we restrict ourselves to a regular two-dimensional Cartesian grid. With the main feature of the Peano framework lying in adaptive Cartesian grids based on space-trees in arbitrary dimensions [5], note that our method can also be applied in an identical fashion to the adaptive counterparts of the described solvers for both 2D and 3D simulations.

The finite element implementation of the Navier-Stokes equation stores its velocity values u^{NS} and the pressure p^{NS} in a staggered manner on the grid [14] (see Fig. 4 on the left). The velocity vectors are computed on the grid vertices whereas the pressure is stored in the center of every grid cell. The degrees of freedom of the Lattice Boltzmann automaton, that is the particle distribution functions f_i, are also evaluated in the cell center (see Fig. 4 on the right).

As a part of the fluid domain shall be computed by Lattice Boltzmann and another part by Navier-Stokes, both solvers require valid boundary data for their simulations at the interface between the respective subdomains. Our solution to this issue is depicted in Fig. 5. For the Lattice Boltzmann method, the distribution functions that would have been streamed from the Navier-Stokes domain into the Lattice Boltzmann domain need to be constructed in a suitable manner. We solve this by surrounding the Lattice Boltzmann subdomain by an additional layer of cells; although being completely evaluated by

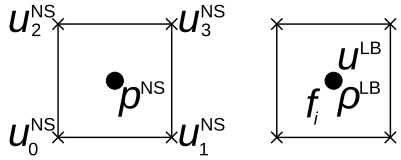

Figure 4: Storage pattern for both solvers. Left: Navier-Stokes solver. Right: Lattice Boltzmann solver.

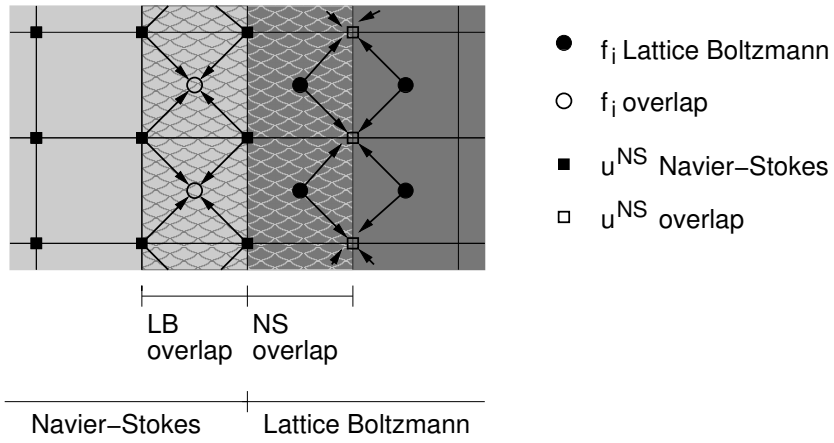

Figure 5: Overlap grid for coupled simulations. The velocities u^{NS} need to be computed from the neighbouring Lattice Boltzmann cells, the particle distribution functions f_i have to be constructed from the velocity and pressure values of the corresponding Navier-Stokes cells.

the Navier-Stokes solver, these cells are used as an overlap region in which the Navier-Stokes input is used to set up boundary values for the Lattice Boltzmann domain; the respective boundary condition is to be pointed out in Section 4.3. On the other hand, consider the vertices between the regular Lattice Boltzmann cells and this new overlap region: The velocities stored on these vertices shall be computed by Navier-Stokes. For this purpose, an additional artificial boundary layer of vertices is needed to evaluate the weak derivatives at these vertices. This is established by marking the first layer of regular Lattice Boltzmann cells as overlap region for the Navier-Stokes solver and the vertices between the created Navier-Stokes overlap and the rest of the Lattice Boltzmann domain as respective boundary vertices for the Navier-Stokes subdomain.

Having introduced our grid setup, we discuss the numerics of the coupling of the degrees of freedom in the following section.

4.2 Lattice Boltzmann–to–Navier-Stokes coupling

As described above, boundary conditions for the Navier-Stokes solver need to be specified at vertices between the pure Lattice Boltzmann domain and the Navier-Stokes overlap utilising the degrees of freedom of the Lattice Boltzmann solver. As the pressure p^{NS} drops out of the continuity equation as described in Section 2.1, it is only the fluid velocities u^{NS} on the artificial boundary nodes at the Navier-Stokes overlap domain that need to be constructed from the particle distribution functions. This is achieved by taking the Lattice Boltzmann velocities u^{LB} computed according to Eq. (2.5) from all neighbouring fluid cells nb of a respective boundary vertex and applying a d-linear interpolation to obtain an approximate solution on this vertex (see Fig. 5). We further need to scale our velocities since standard Lattice Boltzmann implementations refer to a dimensionless grid

with unit meshsize and unit timesteps:

$$u^{NS} = \frac{1}{2^D} \sum_{cells\ nb} \left(u^{LB}_{nb} \cdot \frac{dx}{dt} \right). \tag{4.1}$$

4.3 Navier-Stokes–to–Lattice Boltzmann coupling

While scaled interpolation is sufficient to provide boundary data for the Navier-Stokes domain, this is not the case when moving from the Navier-Stokes to the Lattice Boltzmann system. The reason for this is that the information provided on the coarser macroscale only represents a subset of the information which is present on the mesoscale. In general, the number of particle distribution functions Q (e.g. $Q=9$ for the D2Q9 model) exceeds the number of unknowns $D+1$ of the Navier-Stokes system. However, the equilibrium states $f_i^{eq}(\rho,u)$ of the system can be uniquely described by the macroscopic quantities u and ρ. Hence, we first split the distribution functions in an equilibrium and non-equilibrium part,

$$f_i = f_i^{eq}(\rho^{LB},u^{LB}) + f_i^{neq}, \quad i=1,\cdots,Q. \tag{4.2}$$

The quantities ρ^{LB} and u^{LB} can be determined by direct injection and d-linear interpolation (see Fig. 5) from the Navier-Stokes variables p^{NS} and u^{NS}, including a pressure-to-density conversion according to Eq. (2.27) and a respective rescaling as mentioned in the previous section. One further problem is caused by the macroscopic pressure p^{NS} which is only known up to a certain constant; the constant itself may vary between different simulation setups and even between different timesteps. In order to uniquely determine a valid density expression from the pressure, we follow the method from [12] where the constant is chosen to be the average pressure over the interface in every timestep. In our case this implies computing the mean pressure $\bar{p}^{NS}$ over all cells in the Lattice Boltzmann overlap region and setting

$$\rho^{LB} = \rho_0^{LB} \cdot \left(\frac{p^{NS} - \bar{p}^{NS}}{c_s^2} + 1 \right) \tag{4.3}$$

with the dimensionless reference density $\rho_0^{LB}=1$.

Still, the non-equilibrium parts in Eq. (4.2) need to be determined. Latt et al. [12] introduce further approximations to the terms f_i^{neq} in the Chapman-Enskog expansion and derive an explicit relation between the non-equilibrium parts and the velocity gradients. These approximations comprise a neglection of spatial and temporal derivatives of the non-equilibrium parts and a restriction to a first-order velocity approximation of the equilibrium functions f_i^{eq}. However, especially in case of non-stationary flows or in the first timesteps of a Lattice Boltzmann simulation, where the system initially needs to reach a stable state, spatial and temporal variations in the non-equilibrium parts necessarily need to occur. Besides, a second-order accurate solution with respect to the flow velocity u might be preferable.

Therefore, we follow a different approach for the construction of the non-equilibrium distributions f_i^{neq}. As mass, momentum (see Eq. (2.13) and (2.14)) and viscous stresses have to be conserved at the interface between the two solver regions, we get the following constraints for the non-equilibrium parts:

$$\begin{aligned}
&\sum_i f_i^{neq} = 0, \\
&\sum_i f_i^{neq} c_{i_\alpha} = 0, && \text{for all } \alpha \in \{1,\cdots,D\}, \\
&\sum_i f_i^{neq} c_{i_\alpha} c_{i_\beta} = -\frac{2\nu}{2-\frac{1}{\tau}} \left(\partial_{x_\beta} u_\alpha + \partial_{x_\alpha} u_\beta \right), && \text{for all } \{\alpha,\beta \in \{1,\cdots,D\} : \alpha \leq \beta\},
\end{aligned} \tag{4.4}$$

where the latter equation results from the insertion of the BGK collision operator into Eq. (2.28). Note that this particular choice of the collision operator is only made for simplicity. Other collision operators might as well be used at this point; however, beside changes in the Chapman-Enskog expansion, the properties of the linear system of side constraints might change affecting the following reconstruction procedure of the non-equilibrium parts.

Having $1+D+D(D+1)/2 = (D+1)(D+2)/2$ independent conditions for the non-equilibrium parts (for 2D: 6 conditions), we still have an under-determined system for f_i^{neq}. As deviations from the equilibrium state are typically very small in a fluid at continuum scale, we choose to minimise the non-equilibrium parts in a certain sense to obtain a fully determined system of non-equilibrium deviations f_i^{neq}. Therefore, let

$$g(f^{neq}) = \sum_{i=1}^{Q} \sum_{j=i}^{Q} g_{ij} f_i^{neq} f_j^{neq} + \sum_{i=1}^{Q} g_i f_i^{neq} + g_0 \tag{4.5}$$

be a second order polynomial function in f^{neq}. Then, we use those variables f_i^{neq} which minimise g such that the side constraints from Eq. (4.4) are fulfilled. Thus, for a given function $g(f^{neq})$, we solve

$$\begin{aligned}
&\min_{f^{neq} \in \mathbb{R}^Q} g(f^{neq}) \text{ such that} \\
&\quad \sum_i f_i^{neq} = 0, \\
&\quad \sum_i f_i^{neq} c_{i_\alpha} = 0, && \text{for all } \alpha \in \{1,\cdots,D\}, \\
&\quad \sum_i f_i^{neq} c_{i_\alpha} c_{i_\beta} = -\frac{2\nu}{2-\frac{1}{\tau}} \left(\partial_{x_\beta} u_\alpha + \partial_{x_\alpha} u_\beta \right), && \text{for all } \{\alpha,\beta \in \{1,\cdots,D\} : \alpha \leq \beta\}.
\end{aligned} \tag{4.6}$$

Possible choices for $g(f^{neq})$ are the squared L^2-norm

$$g(f^{neq}) := \|f^{neq}\|_2^2 = \sum_i f_i^{neq^2}, \tag{4.7}$$

an expression representing an approximation for the order of the local Knudsen number (we will refer to this as the "squared Knudsen-norm")

$$g(f^{neq}) := \sum_i \left(\frac{f_i^{neq}}{f_i^{eq}(\rho^{LB}, u^{LB})} \right)^2, \tag{4.8}$$

or an "approximate squared Knudsen-norm" approximating the equilibrium distribution in the const-density low-velocity limit $f_i^{eq} \approx f_i^{eq}(\rho_0^{LB}, \vec{0}) = w_i$:

$$g(f^{neq}) := \sum_i \left(\frac{f_i^{neq}}{w_i} \right)^2. \tag{4.9}$$

The restriction to functions lying in the space of second order polynomials is chosen for the sake of simplicity; for second order polynomials, the minimisation problem described above leads to a linear system of equations for the corresponding Lagrange multipliers and can uniquely be solved if this system is non-degenerated, i.e. its matrix $E \in \mathbb{R}^{(D+1)(D+2)/2 \times (D+1)(D+2)/2}$ has full rank. For positive coefficients $g_{ii} > 0$ of the minimisation polynomial, the matrix E becomes symmetric positive definite and the uniqueness of the solution is proven. For the proof of this statement, see the appendix. The cost functions defined by Eqs. (4.7) and (4.9) have the advantage that the matrix E is known a priori and can easily be inverted. As a consequence, the solving of the small linear systems in all Lattice Boltzmann overlap cells can be omitted and changed into a cheap matrix-vector multiplication.

4.4 Algorithm

Having described the coupling method, the overall algorithm for coupled Lattice Boltzmann–Navier-Stokes simulations shall be reviewed:

```
// choose and set minimisation function g(f^neq)
setMinimisationFunction();

// flag cells and vertices belonging to LB or NS domain
// or belonging to LB or NS overlap domain
setupDomainFlagging();

t=0;
while (t+dt < t_end)
  // construct f_i=f_i^eq + f_i^neq in LB overlap region
  constructPdfsInLatticeBoltzmannOverlap();
  // solve one LB timestep in LB region
  solveLatticeBoltzmannDomain();
```

```
    // compute velocities on vertices of artificial NS boundary
    constructVelocitiesInNavierStokesOverlap();
    // solve one NS timestep
    solveNavierStokesDomain();
    t=t+dt
  end
```

5 Implementation

As the Peano framework provides both implementations of the finite element Navier-Stokes solver and the Lattice Boltzmann method, a coupling according to the descriptions in Section 4 could easily be established. In this section, we present some details of the resulting code structure.

First, in order to provide both particle distribution functions and macroscopic quantities on the grid, the data structures for the vertices and cells were prepared such that they hold both Navier-Stokes and Lattice Boltzmann degrees of freedom at the same time. By this, a dynamic change of the Lattice Boltzmann–Navier-Stokes domain partitioning becomes possible which might turn out as an advantage in case of (dynamically) adaptive simulation setups where one is interested in understanding flow phenomena on different scales in different regions of the simulation domain. In order to combine both solvers without introducing invasive dependencies between them, a new component "solver-coupling" was included in the Peano framework. This component extends existing adapters and allows for modifications of their callbacks during a grid traversal. As an example, instead of executing the Lattice Boltzmann algorithm on the whole grid, a simple extension was plugged in via the solver-coupling component delegating calls to the Lattice Boltzmann solver only for the case that cells in the Lattice Boltzmann region are visited; otherwise, the callbacks to the solver are suppressed. The original callbacks of the solver adapter do not need to be touched for this procedure; it is only the "coupling rule" (in this case: only touch Lattice Boltzmann cells) that needs to be implemented.

More complex rules for the original adapters can be plugged in the same way. Referring to the algorithm in Sect. 4.4, the step `constructPdfsInLatticeBoltzmannOverlap()` can for example be executed during the solving of the Lattice Boltzmann domain, that is during the step `solveLatticeBoltzmannDomain()`. In the same fashion, the construction of the velocities at the artificial Navier-Stokes boundary can be integrated into one of the grid traversals that are carried out by the Navier-Stokes solver. Compared to the stand-alone Navier-Stokes solver, the coupled algorithm only requires one additional grid traversal in the setup phase for the domain flagging and one additional traversal per timestep for solving the Lattice Boltzmann domain. As the number of grid traversals is one of the most important factors with respect to performance within Peano, our solver-coupling approach thus only yields a minimal increase in grid traversals.

Beside the extension and modification of existing adapters, the steering of the coupled simulation needs to be established requiring knowledge of both solver algorithms. This issue is solved by using the concept of multiple inheritance. With each, the Navier-Stokes and the Lattice Boltzmann solver, containing a simulation class for steering and executing the algorithmic steps of the respective solver, we let the coupled simulation class inherit from both steering classes. Having all the single algorithmic steps available in our new class, we can reorder and adapt them according to our algorithm from Section 4.4.

6 Results

The coupling algorithm was tested in a two-dimensional channel flow. In all setups, the flow field was initialised with zero-velocity and constant pressure. For the coupling in the Lattice Boltzmann overlap region, the "squared Knudsen-norm" was applied as a cost function; as the other cost functions showed similar behaviour through all tests, we stick to this minimisation function from now on.

In a first run, a grid of 40×40 cells was set up with a Lattice Boltzmann region of 16×16 cells placed in the middle of the domain; the rest of the channel was left to the Navier-Stokes solver. The simulation was carried out at a Reynolds number $Re=1$ using a timestep $dt=0.01$ and a relaxation time $\tau=0.56$. The parabolic profiles obtained from a pure Navier-Stokes and the coupled simulation in the middle of the channel are presented in Fig. 6. Both profiles are in perfect agreement pointing out the influence of incompressibility of the Navier-Stokes solver on the Lattice Boltzmann domain and the reduction of the equilibrium deviations in the Navier-Stokes–to–Lattice Boltz-

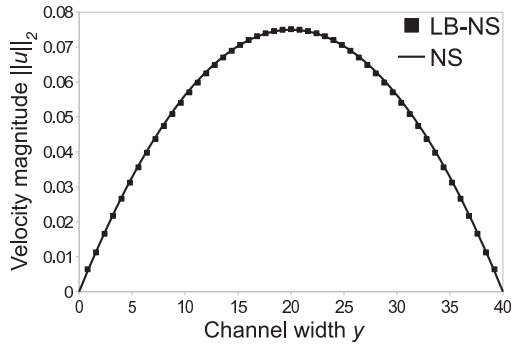

Figure 6: Parabolic profile measured in the middle of a 40×40 channel. The continuous line corresponds to the Navier-Stokes solution whereas the black squares represent the hybrid Lattice Boltzmann–Navier-Stokes solution. Another parabolic curve was obtained from a pure Lattice Boltzmann simulation, which is close to identical to both the Navier-Stokes and the Lattice Boltzmann–Navier-Stokes profile and therefore has been left out in this figure.

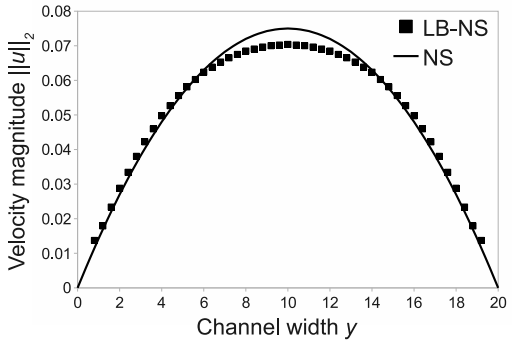

Figure 7: Parabolic profile measured in the middle of a 20×20 channel. The continuous line corresponds to the Navier-Stokes solution whereas the black squares represent the hybrid Lattice Boltzmann–Navier-Stokes solution.

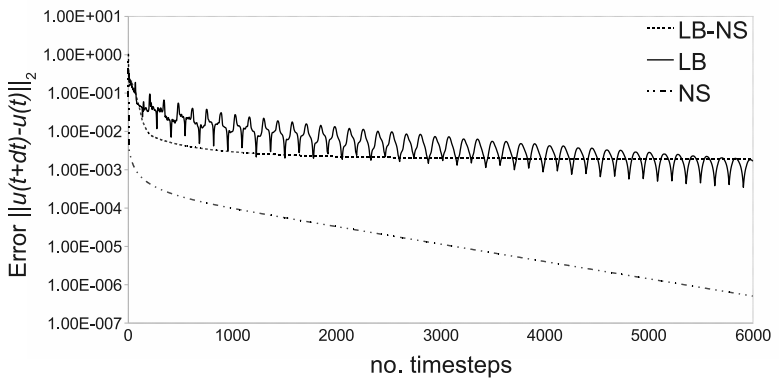

Figure 8: Error $\|u(t+dt)-u(t)\|_2$ measured over time for the channel scenario. The continuous line represents the pure Lattice Boltzmann solution, the dash-dotted line describes the error decay for the Navier-Stokes solver. The dashed curve corresponds to the coupled Lattice Boltzmann–Navier-Stokes solution.

mann coupling. Fig. 8 shows the convergence to the steady state for the same channel cut in a Navier-Stokes, a Lattice Boltzmann and a coupled simulation run using $\|u(t+dt)-u(t)\|_2$ as error guess. In case of the pure Lattice Boltzmann setup, the non-equilibrium bounce back schemes developed in [22] were applied at the in- and outlet boundary and the half-way bounce back rule was used at the no-slip walls of the channel. Especially in the first simulation steps, the order of the error reduction of the coupled system is superior to the one of the Lattice Boltzmann simulation. However, the final error obtained after $\sim$3000 timesteps is approximately of the same order for the coupled and the Lattice Boltzmann simulation; for further explanations on the stalling of the error curves for the two methods, see [12].

An important feature of the coupled version is the smoothing property resulting in a flattened error curve whereas the Lattice Boltzmann simulation shows the typical decaying oscillations. In order to further investigate the influence of the Navier-Stokes solver on the Lattice Boltzmann domain, a second coupled simulation was performed on a 20×20 grid at unit Reynolds number applying the Lattice Boltzmann method in a 8×8 box in the channel center. The simulation could still approximate the parabolic profile (see Fig. 7) whereas a pure Lattice Boltzmann simulation on the same grid became unstable resulting in negative distribution functions due to the small relaxation time $\tau=0.51$ and the very simple initial and boundary conditions.

7 Conclusion

We presented a new two-way coupling method between a finite element Navier-Stokes solver and a Lattice Boltzmann automaton. The coupling was carried out within the Peano framework; hereby, a new component for solver-coupling within the framework turned out to be of essential help, especially with respect to keeping the level of interfer-

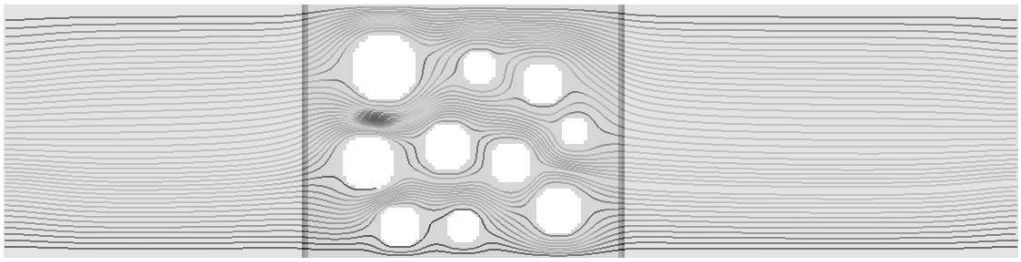

Figure 9: Application of the hybrid Navier-Stokes–Lattice Boltzmann scheme to a two-dimensional directed flow through a simplified porous medium represented by randomly placed spheres. The two vertical dark-grey lines represent the boundaries between the two outer Navier-Stokes domains and the Lattice Boltzmann domain which contains the porous medium.

ence of both solver implementations minimal and reducing the work load for the software developer during coupling. The hybrid Lattice Boltzmann–Navier-Stokes method was tested in a 2D channel flow and showed qualitatively and quantitatively the correct behaviour. Although the convergence for the Lattice Boltzmann method could not be improved by the coupling method, a stabilisation of the BGK relaxation time scheme and a smoothing of the error in the velocity profile could be observed. Further tests will be carried out to completely investigate the potential and capabilities of the introduced coupling and its quality in more complicated simulation setups. This particularly comprises time-dependent setups and setups in non-trivial geometries (such as porous media, see Fig. 9), three-dimensional scenarios and adaptive flow simulations. For adaptive simulations, as mentioned in Section 1, the Navier-Stokes solver might be used on the coarse grid cells whereas the Lattice Boltzmann automaton is applied to the fine cells incorporating additional physics that are only relevant on the small scale levels such as Brownian fluctuations. The mentioned scenarios and setups are subject to current work.

Appendix: On the solution of the minimisation problem

Theorem A.1. *Consider the minimisation problem from Eq.* (4.6),

$$\min_{f^{neq}\in\mathbb{R}^Q} g(f^{neq}) \quad \textit{such that} \quad A\cdot f^{neq}=b$$

with matrix $A\in\mathbb{R}^{K\times Q}$, $\mathrm{rank}(A)=K$, *and vector* $b\in\mathbb{R}^K$ *defining the side constraints for mass, momentum and viscous stress conservation and* $K=(D+1)(D+2)/2<Q$. *If all coefficients* g_{ii} *of* $g(f^{neq})$ *are bigger than zero, then Eq.* (4.6) *has a unique minimum. In particular, the matrix* $E:=ADA^\top$ *with* $D:=\mathrm{diag}(1/(2g_{ii}))\in\mathbb{R}^{Q\times Q}$ *is symmetric positive definite.*

Proof. We define the Lagrange function

$$h(f^{neq}):=g(f^{neq})+\lambda\cdot(b-Af^{neq}) \tag{A.1}$$

with Lagrange multipliers $\lambda \in \mathbb{R}^K$. Differentiating $h(f^{neq})$ with respect to f_i^{neq} yields:

$$\partial_{f_i^{neq}} h(f^{neq}) = 2g_{ii} f_i^{neq} + \left(g_i + \sum_{j=1}^{i-1} g_{ji} + \sum_{j=i+1}^{Q} g_{ij} \right) - \sum_k \lambda_k A_{ki}, \quad i \in \{1, \cdots, Q\}. \tag{A.2}$$

Requiring a vanishing gradient for the Lagrange function leads to:

$$\frac{1}{2g_{ii}} \sum_k \lambda_k A_{ki} = f_i^{neq} + \frac{1}{2g_{ii}} \left(g_i + \sum_{j=1}^{i-1} g_{ji} + \sum_{j=i+1}^{Q} g_{ij} \right), \quad i \in \{1, \cdots, Q\}. \tag{A.3}$$

Multiplying the equation system from Eq. (A.3) with the matrix A results in $E \cdot \lambda = r$ with E defined as above and the right hand side $r \in \mathbb{R}^K$ given by

$$r_k := b_k + \sum_{i=1}^{Q} \frac{A_{ki}}{2g_{ii}} \left(g_i + \sum_{j=1}^{i-1} g_{ji} + \sum_{j=i+1}^{Q} g_{ij} \right), \quad k \in \{1, \cdots, K\}. \tag{A.4}$$

The symmetry of $E = ADA^\top$ is trivial. For the positive definiteness, consider

$$x^\top E x = \sum_{k=1}^{K} x_k \sum_{i=1}^{Q} A_{ki} \frac{1}{2g_{ii}} \sum_{j=1}^{K} A_{ji} x_j = \sum_{i=1}^{Q} \left(\sum_{k=1}^{K} A_{ki} x_k \right)^2 \tag{A.5}$$

for any non-zero vector $x \in \mathbb{R}^K$. As $\mathrm{rank}(A) = K$, there is at least one term $(\sum_{k=1}^{K} A_{ki} x_k)^2 > 0$; thus, it is

$$x^\top E x > 0 \quad \forall x \in \mathbb{R}^K \setminus \{\vec{0}\}. \tag{A.6}$$

This completes the proof. □

References

[1] P. Albuquerque, D. Alemani, B. Chopard, and P. Leone. A Hybrid Lattice Boltzmann Finite Difference Scheme for the Diffusion Equation. Int. J. Mult. Comp. Eng., 4(2):209–219, 2006.

[2] M. J. Berger and P. Collela. Local adaptive mesh refinement for shock hydrodynamics. J. Comput. Phys., 82:64–84, 1989.

[3] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. J. Comput. Phys., 53:484–512, 1984.

[4] P. L. Bhatnagar, E. P. Gross, and M. Krook. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. Phys. Rev., 94(3):511–525, 1954.

[5] H.-J. Bungartz, M. Mehl, T. Neckel, and T. Weinzierl. The PDE framework Peano applied to fluid dynamics: An efficient implementation of a parallel multiscale fluid dynamics solver on octree-like adaptive Cartesian grids. Comput. Mech., 46(1):103–114, June 2010. Published online.

[6] S. Chapman and T. G. Cowling. The Mathematical Theory of Nonuniform Gases. Cambridge University Press, London, 1960.

[7] B. Dünweg, U. D. Schiller, and A. J. Ladd. Statistical Mechanics of the Fluctuating Lattice Boltzmann Equation. Phys. Rev. E, 76(036704), 2007.
[8] S. Geller, M. Krafczyk, J. Tölke, S. Turek, and J. Hron. Benchmark computations based on Lattice-Boltzmann, finite element and finite volume methods for laminar flows. Comput. Fluids, 35(8–9):888–897, 2006. Proceedings of the First International Conference for Mesoscopic Methods in Engineering and Science.
[9] G. Karniadakis, A. Beskok, and N. Aluru. Microflows and Nanoflows. Fundamentals and Simulation. Springer, New York, 2005.
[10] A. J. C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. J. Fluid Mech., 271:285–309, 1994.
[11] J. Latt. Technical report: How to implement your DdQq dynamics with only q variables per node (instead of 2q). 2007. Technical report.
[12] J. Latt, B. Chopard, and P. Albuquerque. Spatial Coupling of a Lattice Boltzmann fluid model with a Finite Difference Navier-Stokes solver, 2005. Published online, `http://arxiv.org/pdf/physics/0511243v1`.
[13] M. Mehl, T. Neckel, and P. Neumann. Navier-Stokes and Lattice-Boltzmann on octree-like grids in the Peano framework. Int. J. Numer. Meth. Fluids, 65:67–86, 2011.
[14] T. Neckel. The PDE Framework Peano: An Environment for Efficient Flow Simulations. Verlag Dr. Hut, 2009.
[15] H. Sagan. Space-filling Curves. Springer-Verlag, New York, 1994.
[16] S. Succi. The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Oxford University Press, New York, 2001.
[17] M. Tij, M. Sabbane, and A. Santos. Nonlinear Poiseuille flow in a gas. Phys. Fluids, 10:1021–1027, 1998.
[18] F. J. Uribe and A. L. Garcia. Burnett description for plane Poiseuille flow. Phys. Rev. E, 60(4):4063–4078, Oct 1999.
[19] T. Weinzierl. A Framework for Parallel PDE Solvers on Multiscale Adaptive Cartesian Grids. Verlag Dr. Hut, 2009.
[20] D. A. Wolf-Gladrow. Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction. Springer, 2000.
[21] Y. Zheng, A. L. Garcia, and B. J. Alder. Comparison of kinetic theory and hydrodynamics for Poiseuille flow. RAREFIED GAS DYNAMICS: 23rd International Symposium, 663(1):149–156, 2003.
[22] Q. Zou. and X. He. On pressure and velocity flow boundary conditions and bounceback for the Lattice Boltzmann BGK model. Phys. Fluids, 9(6):1591–1598, 1997.