

DISCRETE ARTIFICIAL BOUNDARY CONDITIONS FOR THE LATTICE BOLTZMANN METHOD IN 2D

DANIEL HEUBES¹, ANDREAS BARTEL¹ AND MATTHIAS EHRHARDT¹

Abstract. To confine a spatial domain to a smaller computational domain, one needs artificial boundaries. This work considers the lattice Boltzmann method and deals with boundary conditions for these open boundaries. Ideally, such a condition does not interact with the fluid at all. We present novel two-dimensional discrete artificial boundary conditions to pursue that goal and we discuss four different versions. This type of condition is formulated on the discrete lattice Boltzmann level and does not require a PDE formulation of the fluid. We set a special focus on the D2Q9 model. Our numerical results compare the novel discrete artificial boundary conditions to simulations using the existing non-reflecting characteristic boundary condition and an exit boundary condition.

1. INTRODUCTION

In the field of computational fluid dynamics, the lattice Boltzmann (LB) method is a widely used and a flexible tool. Not only its ease of implementation, but also its applicability to complex flows make the LB method attractive for real-world simulations. Applications are found in acoustics (e.g., [1]), blood flow (e.g., [2]) and fluid-structure interaction (e.g. [3]) (among many others).

To achieve an efficient numerical simulation, often the fluid domain is confined to a smaller computational domain. Thereby, some non-physical boundaries, so-called artificial boundaries, occur. Using standard boundary conditions (e.g., a pressure or velocity condition) at open boundaries, the boundaries behave in an unphysical manner: spurious waves are reflected. An ideal boundary condition at artificial boundaries does not create any spurious effects, which influence the simulation results. For the LB method often boundary conditions are derived from known macroscopic physical conditions. However, the problem of finding correct artificial boundary conditions (ABCs) holds also on the macroscopic scale, not only on the mesoscopic scale of the LB method.

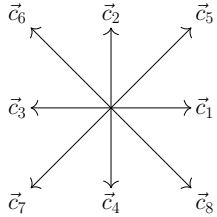
Several studies have been made for artificial boundaries. A review on absorbing boundary conditions for hyperbolic systems can be found in [4]. Hedstrom [5] and Thompson [6] developed *characteristic boundary conditions* (CBCs) for hyperbolic equations. In the LB method, one has to transfer any macroscopic formulation of an ABC to the mesoscopic level. Non-reflecting CBCs were adapted for the LB method [7–9] and create much smaller spurious effects than standard boundary conditions. To avoid a macroscopic formulation, we presented the first one-dimensional ABCs on the discretized LB formulation in earlier works [10, 11]. In the current work these discrete artificial boundary conditions (DABCs) are transferred and applied to two-dimensional LB simulations.

¹ Bergische Universität Wuppertal, Fachbereich Mathematik und Naturwissenschaften, Lehrstuhl für Angewandte Mathematik und Numerische Analysis, Gaußstraße 20, 42119 Wuppertal, Germany
e-mail: {heubes, bartel, ehrhardt}@math.uni-wuppertal.de

This article is structured as follows. In Section 2 we briefly explain the LB method, focusing on the D2Q9 model. The two-dimensional DABCs are constructed in Section 3. This paper ends with the presentation of numerical test results (Section 4) and with conclusions (Section 5).

2. THE LATTICE BOLTZMANN METHOD

We briefly summarize the lattice Boltzmann (LB) method in the two-dimensional space. Based on the chosen discretization one has q discrete velocities \vec{c}_i , $i = 0, \dots, q-1$ in the LB method. This yields the $D2Qq$ LB model (notation proposed by Qian et al. [12]). As one specific example, the popular $D2Q9$ model is given by the discrete velocities

$$\vec{c}_0 = \vec{0}, \quad \vec{c}_i = c \begin{pmatrix} \cos\left(\frac{\pi}{2}(i-1)\right) \\ \sin\left(\frac{\pi}{2}(i-1)\right) \end{pmatrix}, \quad \vec{c}_j = \sqrt{2}c \begin{pmatrix} \cos\left(\frac{\pi}{2}(j-\frac{1}{2})\right) \\ \sin\left(\frac{\pi}{2}(j-\frac{1}{2})\right) \end{pmatrix}, \quad (1)$$


with $i = 1, 2, 3, 4$ and $j = 5, 6, 7, 8$. Here, the parameter $c \neq 0$ scales the velocities. Given a regular lattice, the space points are denoted by \vec{x}_n and time points by t_s . The set of all considered space points is denoted by \mathcal{G}_x . In the LB method, the temporal evolution of so-called *populations* $f_i = f_i(\vec{x}_n, t_s)$ is computed for each lattice node \vec{x}_n and time t_s . That is, $f_i(\vec{x}_n, t_s)$ gives the number density (scaled by mass) of fictitious particles with velocity \vec{c}_i at each lattice node (\vec{x}_n, t_s) . The evolution of populations is described by the LB equation, which defines an update rule of the populations based on particles' collision and streaming (see also, e.g., [13–15] for more details):

$$f_i(\vec{x}_n + \vec{c}_i, t_{s+1}) = f_i(\vec{x}_n, t_s) + C_i(\vec{f}(\vec{x}_n, t_s)), \quad \text{for } i = 0, \dots, q-1. \quad (2)$$

Here the vector $\vec{f}(\vec{x}_n, t_s)$ gathers all populations at the lattice node (\vec{x}_n, t_s) . In the interior of \mathcal{G}_x , it is required to hold

$$\vec{x}_m := \vec{x}_n + \vec{c}_i \in \mathcal{G}_x, \quad \text{for all } i \in \{0, \dots, q-1\}, \quad (3)$$

such that particles move in one time step $t_s \rightarrow t_{s+1}$ exactly from one node to an adjacent node. The right hand side of (2) gives the populations after particle collisions, hence C_i models the change due to collision. A very popular choice for C_i is given by the BGK scheme [16], which is a single relaxation time (SRT-BGK) model. There are also models with more relaxation parameters, e.g., the multiple relaxation time (MRT) model [17] or the two-relaxation time model [18].

Using the SRT-BGK model, the LB equation (2) reads

$$f_i(\vec{x}_n + \vec{c}_i, t_{s+1}) = (1 - \omega)f_i(\vec{x}_n, t_s) + \omega f_i^{\text{eq}}(\vec{x}_n, t_s),$$

where $f_i^{\text{eq}}(\vec{x}_n, t_s)$ is a local equilibrium distribution and $\omega = 1/\tau$ a free relaxation parameter. For example in the D2Q9 model, the equilibrium reads

$$f_i^{\text{eq}}(\vec{x}_n, t_s) := E_i(\rho, \vec{u}) := w_i \rho \left[1 + \frac{3}{c^2} (\vec{c}_i \cdot \vec{u}) + \frac{9}{2c^4} (\vec{c}_i \cdot \vec{u})^2 - \frac{3}{2c^2} |\vec{u}|^2 \right] \quad (4)$$

with weights $w_0 = \frac{4}{9}$, $w_{1-4} = \frac{1}{9}$ and $w_{5-8} = \frac{1}{36}$. The fluid quantities

$$\rho(\vec{x}_n, t_s) = \sum_{i=0}^{q-1} f_i(\vec{x}_n, t_s), \quad \vec{u}(\vec{x}_n, t_s) = \begin{pmatrix} v(\vec{x}_n, t_s) \\ w(\vec{x}_n, t_s) \end{pmatrix} = \frac{1}{\rho(\vec{x}_n, t_s)} \sum_{i=1}^{q-1} \vec{c}_i f_i(\vec{x}_n, t_s) \quad (5)$$

are then approximations to the solution of the Navier-Stokes equations. This can be verified with help of a Chapman-Enskog expansion or other asymptotic analyses [19, 20]. Note that the arguments (\vec{x}_n, t_s) were suppressed in (4) for better readability.

The LB method requires the presence of all q populations in all grid points at any time. This requirement is ensured by the LB equation provided condition (3) is fulfilled. Indeed, condition (3) holds for all interior nodes, but is not given for those nodes near the boundary of the computational domain. In fact, we define a boundary node by the lack of some adjacent nodes. That is, a node $\vec{x}_b \in \mathcal{G}_x$ of the spatial discretization is said to be a boundary node if $\vec{x}_b + \vec{c}_i \notin \mathcal{G}_x$ for at least one discrete velocity \vec{c}_i . If \vec{c}_k is such a velocity (i.e., $\vec{x}_b + \vec{c}_k \notin \mathcal{G}_x$), then a boundary condition for $f_{\bar{k}}(\vec{x}_b, \cdot)$ is needed, where \bar{k} denotes the index defined by $\vec{c}_{\bar{k}} = -\vec{c}_k$. These unknown populations can be computed, e.g., by specifying a fluid pressure or velocity [21, 22]. The desired condition depends on the kind of the boundary. If the boundary is given physically as a wall, then often no-slip boundary conditions are applied. For open boundaries, which are not aligned with a physical boundary, sometimes periodic boundary conditions are reasonable. Applying a pressure or a velocity condition at these lattice sites will generate undesired, spurious reflections. In the following, we present a further approach, which aims at computing the unknown populations for open boundaries such that preferably no reflection in the fluid quantities (5) occurs. In this sense, such a boundary represents the physically correct behavior.

3. DISCRETE ARTIFICIAL BOUNDARY CONDITION

Next, we determine the unknown populations of boundary nodes in a two-dimensional LB simulation for an open boundary by generalizing our one-dimensional approach [11].

3.1. Basic approach of discrete artificial boundary conditions

Let $\Gamma = \{\vec{x}_k \in \mathcal{G}_x \mid \vec{x}_k \text{ is a boundary node}\}$ be the set of all boundary nodes. Then at all $\vec{x}_b \in \Gamma$ there are some populations which have to be computed by a boundary condition. However, for the moment we restrict our explanation to those boundary nodes $\Gamma_E \subset \Gamma$ of a rectangular domain, for which a right adjacent node is missing, i.e., $\vec{x}_b^E \in \Gamma_E$ if and only if $\vec{x}_b^E + \vec{c}_k \notin \mathcal{G}_x$ and $c_{k,\alpha} > 0$ (where $\vec{c}_k = (c_{k,\alpha}, c_{k,\beta})^\top$). The situation is sketched in Fig. 1, where periodic boundary conditions are assumed to hold at the top and bottom of the computational domain. This assumption avoids having corners, which will be considered later. For the D2Q9 model (1), the task of the boundary condition in these nodes is to assign the populations $f_{3,6,7}(\vec{x}_b^E, t_s)$, $\vec{x}_b^E \in \Gamma_E$ (for any time level $t = t_s$). We return to a general formulation afterwards.

A novel discrete boundary condition for one-dimensional LB simulations was developed by the authors in [11]. There, the unknown populations at boundary nodes are derived by considering so-called LB subproblems. We follow the same idea to construct a boundary condition in two space dimensions. This means we solve two-dimensional LB subproblems to obtain the unknown populations. At each time level an individual subproblem is considered. Therefore the subproblems are labeled: the s -th subproblem is used to compute all unknown populations at time $t = t_s$.

We explain the procedure in detail for Γ_E , the D2Q9 case and by considering one subproblem. The s -th subproblem employs a usual LB simulation on a new set of nodes \mathcal{G}_x^s with equal lattice spacing (as in the original problem). The computational domain \mathcal{G}_x^s mainly consists of fictitious nodes \mathcal{F}_x^s , which are located in the exterior of \mathcal{G}_x , see Fig. 1. These fictitious nodes emulate the continuation of the lattice into the exterior. The number of \mathcal{F}_x^s nodes is influenced by a free parameter $H(s) \in \mathbb{N}$, which here represents the number of fictitious layers of nodes in x_1 -direction, see Fig. 1. Then the subproblem's domain \mathcal{G}_x^s is given by $\mathcal{G}_x^s := \Gamma_E \cup \mathcal{F}_x^s$ (visualized by the dashed rectangle in Fig. 1). Obviously, the original problem and the subproblem share nodes:

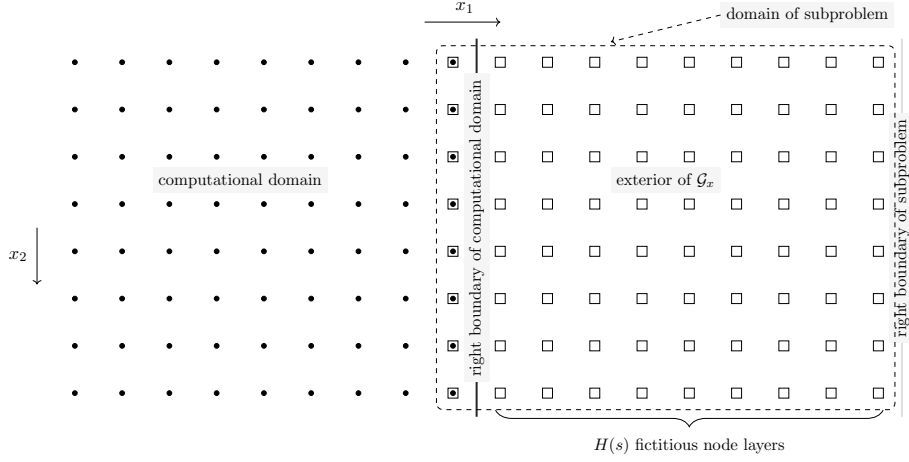


FIGURE 1. Time: $t = t_s$. Beyond the boundary of the computational domain fictitious nodes (\square) are added. They are used in the computation of the s -th subproblem.

the interface nodes Γ_E . On this interface (Γ_E), we have both the usual populations f_i and the subproblems' populations.

Now, let us denote the populations of the s -th LB subproblem by $h_i^s(\vec{x}_m, t_k)$, $\vec{x}_m \in \mathcal{G}_x^s$ and $t_k \in \mathcal{T}^s$. In \mathcal{G}_x^s , the h_i^s evolve according to the usual LB equation (2) (with f_i replaced by h_i^s). We demand to proceed $H(s)$ time levels in the subproblem, such that after $H(s)$ applications of the LB equation we reach the actual time level of the original problem, i.e., $t = t_s$. Thus, the time points for the s -th subproblem are $\mathcal{T}^s = \{t_{s-H(s)}, \dots, t_s\}$. For $t_k \in \{t_{s-H(s)}, \dots, t_{s-1}\}$ we use as boundary condition on Γ_E for the subproblem:

$$h_i^s(\vec{x}_b^E, t_k) := f_i(\vec{x}_b^E, t_k), \quad \vec{x}_b^E \in \Gamma_E.$$

These are known at time t_s . Analogously, after the $H(s)$ applications of the LB equation, we obtain the unknown populations $f_{3,6,7}$ by

$$f_{3,6,7}(\vec{x}_b^E, t_s) := h_{3,6,7}^s(\vec{x}_b^E, t_s), \quad \vec{x}_b^E \in \Gamma_E. \quad (6)$$

This equation eventually formulates the discrete artificial boundary condition (DABC). Figs. 2 and 3 provide flow charts of the procedure.

We remark, that the value of H is arbitrary and may vary with time t . We suggest to determine a maximal value $H_{\max} \in \mathbb{N}$ for the whole simulation ($H(s) \leq H_{\max}$, for all s) and to use $H(s) = \min\{s, H_{\max}\}$. If we consider the setting where the exterior domain is initially assumed in an equilibrium state, then the uncapped choice $H(s) = s$ (for all $s \in \mathbb{N}$) would result in an ideal boundary condition. However, the computational effort would be too high. Further we note that choosing $H(s)$ layers of fictitious nodes and simulating $H(s)$ time levels of the subproblem guarantees that no other boundary of the subproblem's domain affects the interface nodes (which is the boundary of the computational domain) at the time levels \mathcal{T}^s .

Although the basic approach of the DABC is already explained by (6), an application is not possible unless initialization and boundary conditions of the s -th subproblem have been discussed. This is addressed in following paragraph.

3.2. Well-definedness

Each subproblem, or to be more precisely the final mapping (6), is well-defined if there is an initialization rule for all its populations and if boundary conditions for certain appropriate populations are formulated. We

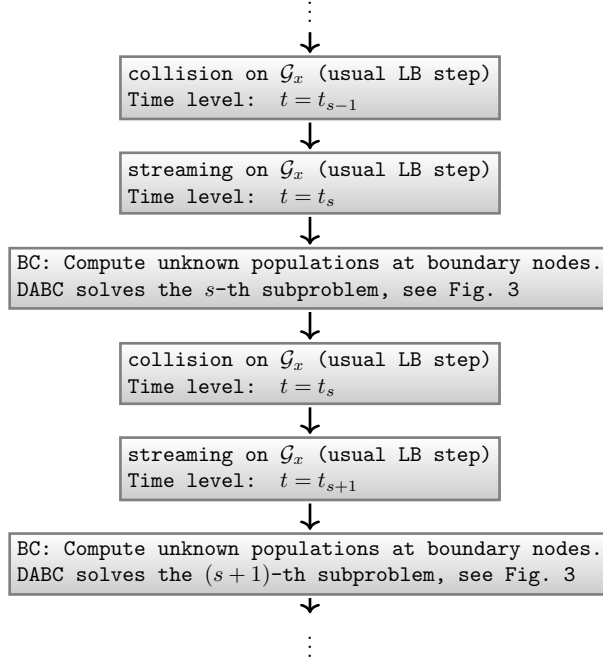


FIGURE 2. Flow chart explaining the application of the DABC.

remark that the initialization is the crucial part of the DABC, since all errors are caused here. We interpret an error $e_{f_i}(\vec{x}_b^E, t_s)$ as the deviation from an ideal population f_i^{ref} :

$$e_{f_i}(\vec{x}_b^E, t_s) := f_i(\vec{x}_b^E, t_s) - f_i^{\text{ref}}(\vec{x}_b^E, t_s).$$

For any choice of $H(s)$, there has to exist an initialization of the subproblem's populations, which leads to $e_{f_i} = 0$. However, there is no general approach for finding appropriate populations for the initialization of the subproblem. An ideal initialization strongly depends on all processes in the computational domain.

To explain the initialization further, we consider the s -th subproblem of the problem from the previous paragraph. Now, if no better information is available, we propose two strategies for the initialization of the subproblems: a) *homogeneous equilibrium* for given ρ^s and \vec{u}^s

$$h_i^s(\vec{x}_m, t_0^s) = E_i(\rho^s, \vec{u}^s), \quad \forall \vec{x}_m \in \mathcal{F}_x^s, \quad (7)$$

or b) *constant extrapolation of initial boundary populations*

$$h_i^s(\vec{x}_m, t_0^s) = f_i(\vec{x}_b^E, t_0), \quad \forall \vec{x}_m \in \mathcal{F}_x^s, \quad \vec{x}_b^E \in \Gamma_E \text{ with } x_{m,\beta} = x_{b,\beta}^E, \quad (8)$$

where Greek indices denote spatial coordinates ($\vec{x} = (x_\alpha, x_\beta)^\top$). Also other initializations are conceivable, e.g., a convex combination of the above both possibilities. Independently of the initialization at nodes at \mathcal{F}_x^s , we always assign the populations at the interface nodes $\vec{x}_b^E \in \Gamma_E$ as follows (for all involved time levels)

$$h_i^s(\vec{x}_b^E, t_k) = f_i(\vec{x}_b^E, t_k), \quad i = 0, \dots, q-1, \quad t_k \in \{t_{s-H(s)}, \dots, t_{s-1}\}. \quad (9)$$

The assignment (9) is an initialization for $t_k = t_{s-H(s)}$ and a boundary condition for $t_k \in \{t_{s-H(s)+1}, \dots, t_{s-1}\}$. For the top and bottom boundary of the subproblem, we use here periodic boundary conditions, for simplicity.

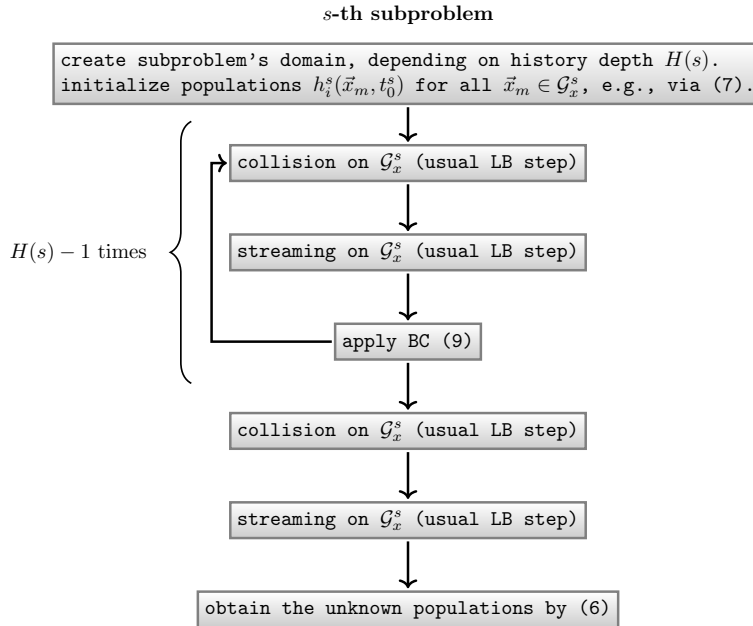


FIGURE 3. Flow chart explaining the procedure of an individual subproblem as referred to in Fig. 2.

In the subproblem, no boundary condition is required for the boundary nodes on the opposite site of Γ_E (that is the right boundary of \mathcal{G}_x^s). This is, because from this boundary only information from time $t = t_0^s = t_{s-H(s)}$ is affecting (6).

From (9) we see that the DABC takes past information of the original problem up to $H(s)$ time levels ago. For this reason the quantity $H(s)$ is called the *history depth of the s -th subproblem* and H_{\max} is denoted as the *maximal history depth*.

3.3. Generalization

So far, the basic approach was explained for the right boundary of a rectangular computational domain with periodic boundaries at the top and bottom. In the following we generalize the formulation of the DABC without any restrictions. Let Γ denote the set of boundary nodes, where the DABC should be used to compute the inward populations $f_k(\vec{x}_b, t_s)$ at time $t = t_s$. To this end, we employ the s -th subproblem, whose lattice \mathcal{G}_x^s consists of fictitious nodes \mathcal{F}_x^s (\square in figures) and joint interface nodes Γ (\blacksquare in the figures). The amount and positioning of the fictitious nodes depends on the original problem and the chosen history depth $H(s)$. We consider an example with different positionings of the fictitious nodes in the subsequent paragraph. The fictitious nodes have to be chosen such that after $H(s)$ applications of the LB equation all required quantities are given.

Let us consider the application of the DABC for two adjacent boundaries as depicted in Fig. 4 (for the top and right boundary). This small example demonstrates that the treatment of corners requires to consider one joint subproblem to achieve all unknown populations. It is not possible to consider two independent subproblems, one for the top and right boundary each. In the example, after 5 streaming steps the information from the filled square node enters a boundary node on the right side. Hence, it is important that the fictitious nodes which extend the computational domain vertically and horizontally build a connected set of LB nodes.

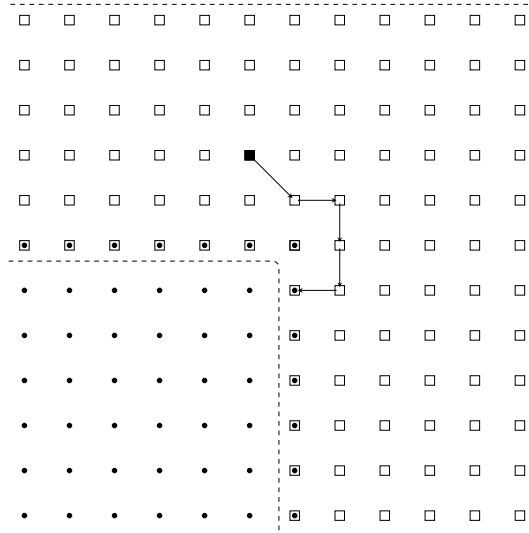


FIGURE 4. Application of the DABC ($H = 5$) for two adjacent boundaries. Nodes inside the dashed lines create the subproblem’s domain. Populations from the filled square node are tracked during the iteration of the subproblem (arrows).

Given the s -th subproblem, we perform $H(s)$ iterations of it. For the interface nodes $x_b \in \Gamma$, we set all populations according to

$$h_i^s(\vec{x}_b, t_k) = f_i(\vec{x}_b, t_k), \quad i = 0, \dots, q - 1, \quad t_k \in \{t_{s-H(s)}, \dots, t_{s-1}\}. \quad (10)$$

At time $t_0^s = t_{s-H(s)}$ we also need an initialization for the subproblem’s interior nodes, which is done, e.g., by adapting (7) or (8). After all $H(s)$ iterations are done, the unknown populations of the original problem are obtained:

$$f_k(\vec{x}_b, t_s) = h_k^s(\vec{x}_b, t_s), \quad \vec{x}_b \in \Gamma. \quad (11)$$

3.4. Discrete artificial boundary condition for a channel

Let us consider a channel flow (in x_1 -direction) with the aim to apply the DABC at the right boundary of the channel. This means, the inward populations at \blacksquare -nodes have to be computed, see Figs. 5 and 6. In Fig. 5 the situation is sketched for a theoretical choice of fictitious nodes, having more nodes in x_2 -direction than the actual computational domain. Contrary, in Fig. 6 the fictitious nodes have equal number in x_2 -direction, but also channel walls are incorporated in the subproblem’s domain (\blacktriangle and \blacksquare -nodes). Both cases represent a well defined situation (cf. Section 3.2), however only Fig. 6 seems to be physically reasonable.

This example demonstrates that there is no general rule for the subproblems. It is recommend to choose the subproblems the same way as a logical enlargement of the computational domain would be. The situation of Fig. 5 represents an outlet of a channel, whereas in Fig. 6 the boundary of the computational domain is within the channel.

3.5. Computational costs

The specific computational effort of the DABC depends on the LB model in use (SRT or MRT, with/without external forces, ...). Therefore we do not count arithmetic operations here. At each time level a LB subproblem

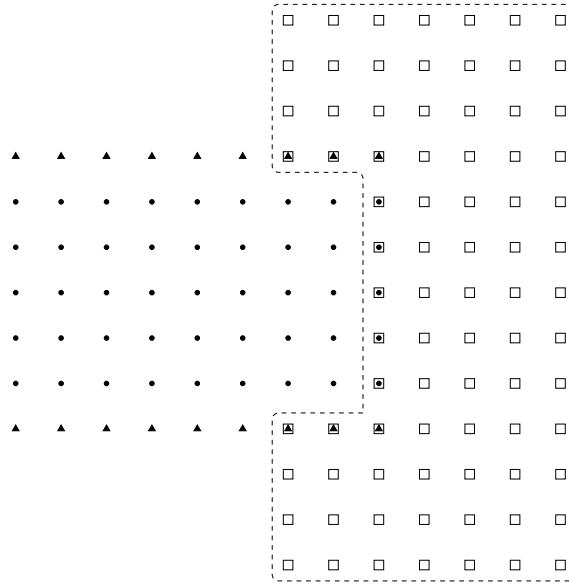


FIGURE 5. Application of DABC (here $H = 4$) at the right boundary of a channel. There are no channel walls in the subproblem. Nodes surrounded by the dashed line create the subproblem's domain.

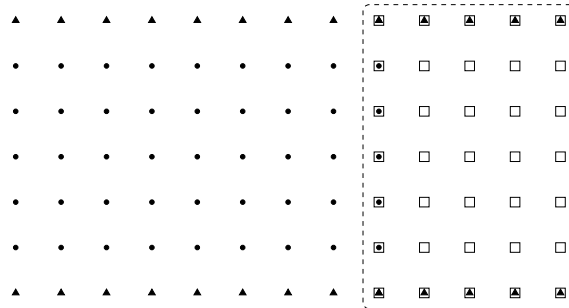


FIGURE 6. Application of DABC (here $H = 4$) at the right boundary of a channel. In the subproblem's domain there are also channel walls (squares filled with triangles). Nodes in the dashed rectangle are the subproblem's domain.

is solved. It is possible to align the collision and streaming steps with those of the original problem. A more detailed description of this type of enhancement is given in [11]. This means the DABC can be parallelized in the same way as the main LB simulation. In this implementation strategy, one has to treat at most H_{\max} subproblems simultaneously. If each subproblem consists of $J := \#\mathcal{G}_x^i$ nodes, the total costs of the DABC are equivalent to a lattice enlargement by $H_{\max} \cdot J$ nodes.

4. NUMERICAL RESULTS

Here we are able to describe the working principle of the DABC by a visual interpretation of the subproblems. Furthermore, we present results for two test scenarios using the D2Q9 model. To rate the performance and the errors of the (DABC), we compare our DABC results with results obtained from a constant pressure

condition [21], an exit boundary condition based on Grad's approximation [23], and with results from a non-reflecting characteristic boundary condition (CBC) [7–9]. In fact, for the CBC at the right boundary of a rectangular computational domain, we numerically solve the system

$$\frac{\partial \vec{U}}{\partial t} = - \begin{pmatrix} \frac{1}{2c_s^2} & 0 & \frac{1}{2c_s^2} \\ -\frac{1}{2\rho c_s} & 0 & \frac{1}{2\rho c_s} \\ 0 & 1 & 0 \end{pmatrix} \vec{\mathcal{L}}_x$$

at the boundary and transfer the outcome into a Dirichlet condition for the populations. Here $\vec{U}^\top = (\rho, v, w)$ is the vector of the fluid quantities (5) and $\vec{\mathcal{L}}_x$ denotes the wave amplitude variations:

$$\vec{\mathcal{L}}_x = \begin{pmatrix} \mathcal{L}_{x,1} \\ \mathcal{L}_{x,2} \\ \mathcal{L}_{x,3} \end{pmatrix} = \begin{pmatrix} \tilde{\lambda}_1 \vec{\ell}_1^\top \frac{\partial \vec{U}}{\partial x} \\ \tilde{\lambda}_2 \vec{\ell}_2^\top \frac{\partial \vec{U}}{\partial x} \\ \tilde{\lambda}_3 \vec{\ell}_3^\top \frac{\partial \vec{U}}{\partial x} \end{pmatrix}, \quad \text{with } \tilde{\lambda}_i = \begin{cases} \lambda_i & \text{outgoing} \\ 0 & \text{incoming,} \end{cases} \quad (\vec{\ell}_1, \vec{\ell}_2, \vec{\ell}_3) = \begin{pmatrix} c_s^2 & 0 & c_s^2 \\ -c_s \rho & 0 & c_s \rho \\ 0 & 1 & 0 \end{pmatrix}$$

and eigenvalues $\lambda_1 = v - c_s$, $\lambda_2 = v$, $\lambda_3 = v + c_s$. For more details and the treatment of other boundaries we refer to [9].

In the exit condition [23] the unknown populations are computed by evaluating Grad's approximation:

$$f_i^G(\rho, \vec{u}, P) = w_i \rho \left(1 + \frac{3}{c^2} (\vec{c}_i \cdot \vec{u}) + \frac{9}{2c^4} [P - c_s^2 \rho I] : [\vec{c}_i \vec{c}_i^\top - c_s^2 I] \right).$$

Here we use the Frobenius inner product $A : B = \text{tr}(AB^\top)$, the identity $I \in \mathbb{R}^{d \times d}$ and the pressure tensor P

$$P(\vec{x}_n, t_s) = \sum_{i=0}^{q-1} f_i(\vec{x}_n, t_s) \vec{c}_i \vec{c}_i^\top.$$

For the boundary condition Grad's approximation is evaluated with values for ρ , \vec{u} and P from the previous time level. See also [23] for more details on this boundary condition.

4.1. Concentric wave

For the first numerical test, we consider a fluid domain with a Gaussian pressure pulse:

$$p(x, y) = p_0 + (p_{\max} - p_0) \exp\left(\frac{-(x^2 + y^2)}{2\sigma^2}\right).$$

The pressure values are related to the density by

$$p(x, t) = c_s^2 \rho(x, t) = \frac{c^2}{3} \rho(x, t).$$

We set $\sigma = 0.1$ and the pressures according to $\rho_0 = 1$, $\rho_{\max} = 1.15$. The fluid velocity is homogeneously set to $\vec{u}(\cdot, t_0) = \vec{u}_0$ at initial time.

The rectangular lattice is chosen with 201×1001 nodes, representing the spatial domain $[-1, 1] \times [-5, 5]$. We apply periodic boundary conditions for the top and bottom boundary. A LB reference solution on a sufficiently larger domain is computed, such that errors

$$e_z(\vec{x}_n, t_s) := z(\vec{x}_n, t_s) - z^{\text{ref}}(\vec{x}_n, t_s), \quad \vec{x}_n \in \mathcal{G}_x, \quad s \in \mathbb{N}, \quad (12)$$

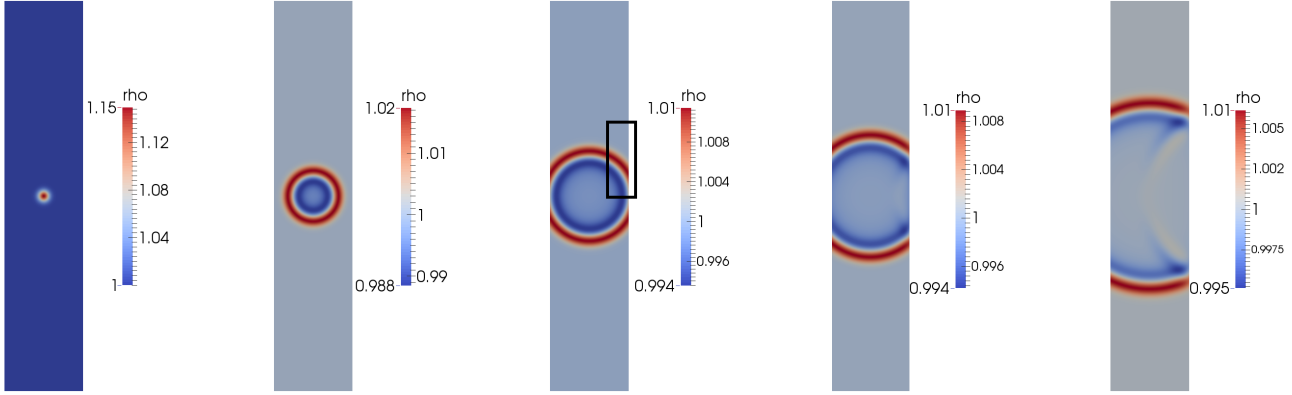


FIGURE 7. Temporal evolution of a Gaussian pressure pulse, here the density is illustrated. Snapshots correspond to $t \in \{t_0, t_{100}, t_{185}, t_{255}, t_{400}\}$. A reflecting wave can be observed traveling from the right boundary back into the interior.

can be computed for any available quantity z (e.g., $z \in \{\rho, \vec{u}, f_i\}$). With help of the reference populations an ideal boundary condition is applied at left boundary nodes \vec{x}_b^W :

$$f_j(\vec{x}_b^W, t_s) = f_j^{\text{ref}}(\vec{x}_b^W, t_s), \quad j \in \{1, 5, 8\}, \quad s \in \mathbb{N}.$$

On the right boundary we apply the DABC or any other condition to be tested.

4.1.1. Visual interpretation

In Fig. 7 the density profile is plotted for different time levels using the maximal history depth $H_{\max} = 40$ and the relaxation parameter $\omega = 1$. To give a visual interpretation of how the DABC is working, we consider the situation at time $t = t_{185}$. In fact, we zoom in the marked section of the third snapshot of Fig. 7. Therefore, in Fig. 8 the mass density ρ of the computational domain is shown at $t = t_{185}$. Moreover, the mass density in the 185-th subproblem is shown. On the different plots of Fig. 8 we see the mass density profile of the subproblem changing during the iterations of the subproblem. Note that the profile of the original problem is not proceeding forward in time, while the subproblem does. The first plot shows the subproblem at its initialization, which is done by (7) and (10) ($\rho^s = 1$, $\vec{u}^s = \vec{u}_0 = \vec{0}$). Only the last plot is relevant for the desired populations, which are given by (6) or (11), respectively. We see that after the final iteration of the subproblem both regions match together. Thus, the DABC constructs a suitable extension of the computational domain in fictitious nodes, which provides then all information for the unknown populations. This interpretation clarifies that the initialization of subproblems determines the accuracy of the DABC.

4.1.2. Simulation results

In addition to Fig. 7 the errors e_ρ , e_v and e_w , cf. (12), are depicted in Fig. 9 for time $t = t_{400}$. The error plots shown in the left column correspond to the subproblem initialized by (7), which is here equivalent to (8) due to the choice of ρ^s and \vec{u}^s . Errors in the right column correspond to a modified initialization of (8), where the time of evaluation is not fixed (t_0^s instead of t_0):

$$h_i^s(\vec{x}_m, t_0^s) = f_i(\vec{x}_b^E, t_0^s), \quad \forall \vec{x}_m \in \mathcal{F}_x^s, \quad \vec{x}_b^E \in \Gamma_E \text{ with } x_{m,\beta} = x_{b,\beta}^E. \quad (13)$$

For each error e_z we can see different shapes of the surfaces. It should be noted that the peaks are not located at the same points. Furthermore, we emphasize that an initialization according to (13) created instabilities in the one-dimensional case [11].

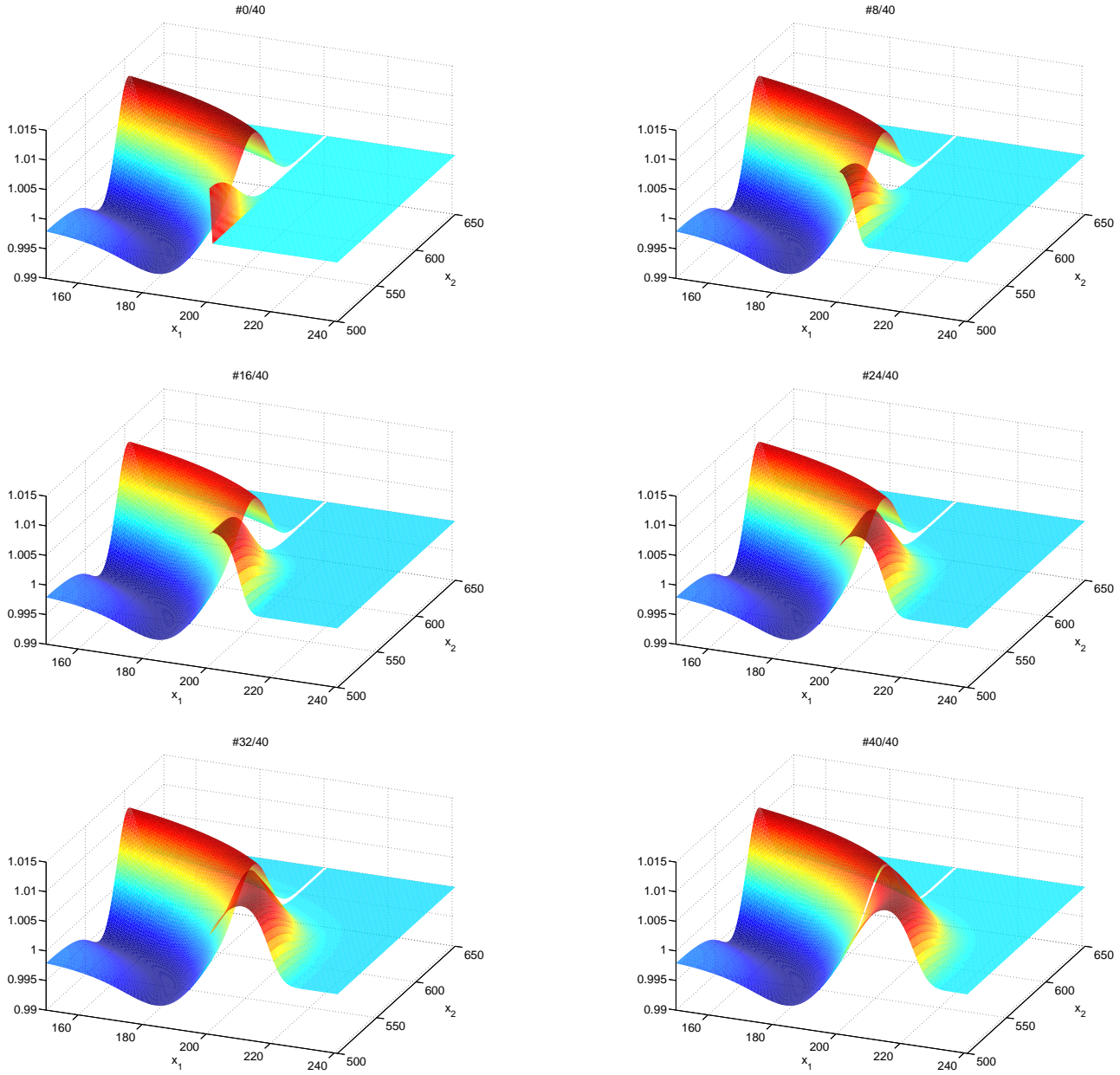


FIGURE 8. Temporal evolution of the 185-th subproblem ($x_1 \in [201, 241]$), which was initialized by (7)/(8). The plots show the section marked in Fig. 7. The profile of the original problem is shown for $x_1 \leq 200$.

In the sequel we consider normalized errors

$$N_z(t_s) := \|e_z(\cdot, t_s)\|_{\ell_2} = \sqrt{\sum_{\vec{x}_n \in \mathcal{G}_x} \left(z(\vec{x}_n, t_s) - z^{\text{ref}}(\vec{x}_n, t_s) \right)^2}, \quad (14)$$

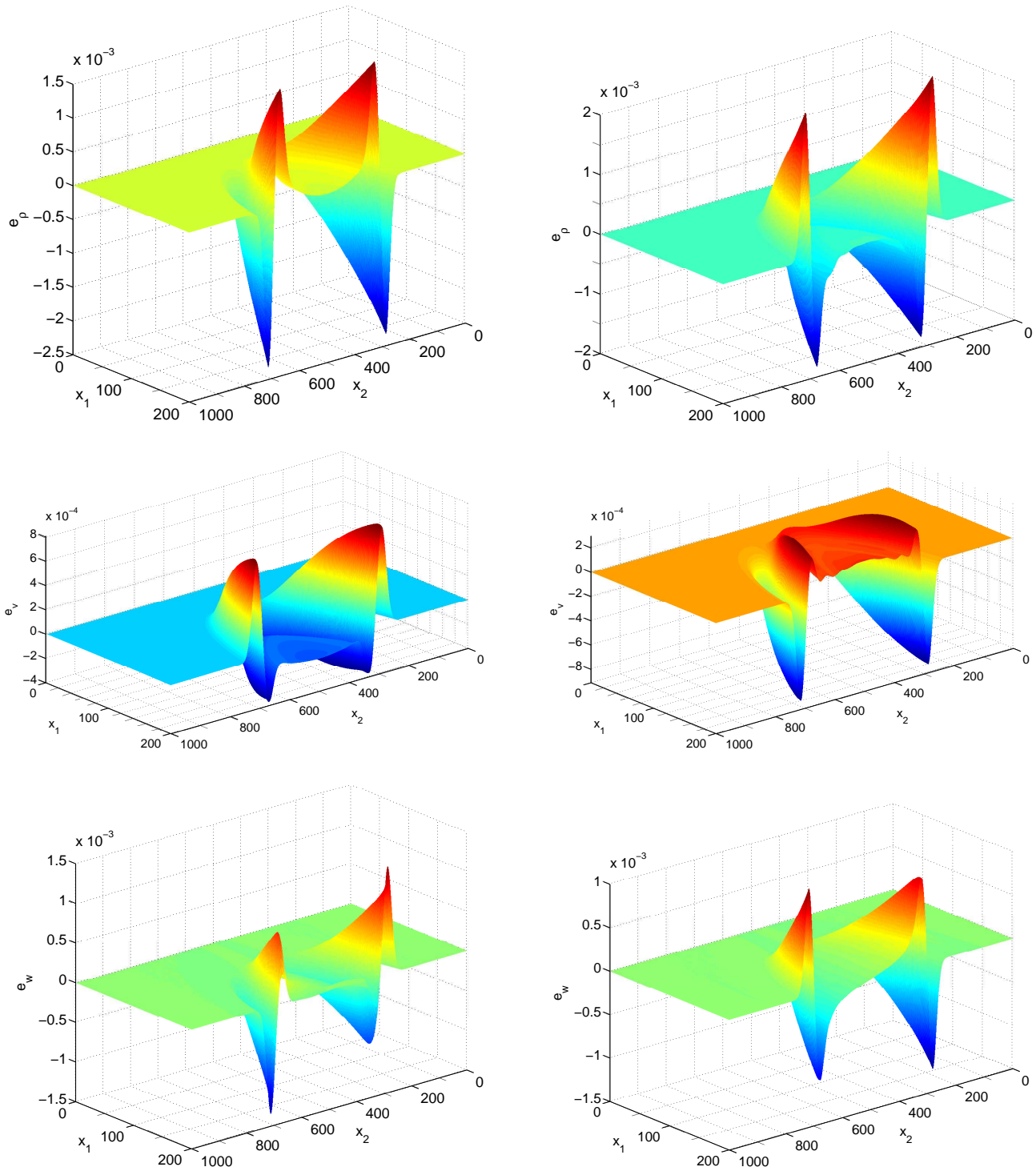


FIGURE 9. The left column shows the errors e_ρ , e_v and e_w (cf. (12)) when using the DABC with (7)/(8) for the first test case (concentric wave). Similarly, the right column shows errors when initializing with (13).

	Error $N_\rho(t)$ at			
	$t = t_{175}$	$t = t_{250}$	$t = t_{325}$	$t = t_{400}$
Zou/He pressure	0.472981	0.678311	0.722984	0.723998
Grad's	0.128035	0.162139	0.155232	0.148781
CBC	0.016336	0.066936	0.108164	0.135353
DABC H=4, (7)/(8)	0.016035	0.077023	0.119626	0.146777
DABC H=4, (13)	0.030312	0.059209	0.091554	0.116822
DABC H=10, (7)/(8)	0.011039	0.065418	0.107186	0.134847
DABC H=10, (13)	0.020151	0.060705	0.097759	0.124270
DABC H=20, (7)/(8)	0.007960	0.061694	0.104964	0.133531
DABC H=20, (13)	0.015407	0.062910	0.102058	0.129025
DABC H=40, (7)/(8)	0.001986	0.055666	0.104786	0.136482
DABC H=40, (13)	0.007918	0.070754	0.114453	0.141661
DABC H=80, (7)/(8)	0.000002	0.027226	0.085173	0.121441
DABC H=80, (13)	0.000033	0.075978	0.145701	0.173462

TABLE 1. Errors N_ρ for different boundary conditions and time levels. First test case (Gaussian pressure pulse), $\omega = 1$ and $\vec{u}_0 = 0$.

where z can be replaced by any available quantity.

In the one-dimensional case the error could be decreased by taking a larger history depth, see [11]. First we investigate if a similar behavior can be detected also in two space dimensions. Therefore, we consider the above test case repeatedly with different maximal history depths in the range from 4 to 80 and compute the corresponding errors N_z . The parameters chosen for the simulation are $\omega = 1$ and $\vec{u}_0 = 0$. Results are presented in Tables 1–3, which also contain reference values. That is, the errors obtained when using a Zou/He pressure boundary condition ($\rho = 1$ and $u_{\parallel} = 0$) [21], an exit BC based on Grad's approximation [23], as well as a CBC (LODI) from [9] as described at the beginning of the current section. All in all, the smallest errors are obtained when using the novel DABC with an initialization due to (7) respectively (8). We see that the CBC and the DABC behave equally in N_ρ and N_v , whereas for N_w the DABC is superior. The Grads' based boundary condition performs best compared to the artificial boundary conditions for the parallel velocity component. However, unlike the one-dimensional case, a significant decreasing influence of a larger history depth is not visible. The test case is challenging, because the wave is interacting with the boundary all the time, which is different to the one-dimensional test cases in [11]. In our opinion, this difference causes that only a decreased influence of the history depth can be detected.

For the presentation of the next results we do not change the parameters. But, we fix the maximal history depth by $H_{\max} = 20$ and vary the relaxation time $\tau = \frac{1}{\omega}$ from 0.6 to 1.5. The errors are computed at time $t = t_{400}$ and presented in Table 4. Note that the viscosity ν is related to the relaxation time as

$$\nu = \frac{2\tau - \Delta t}{6} c^2.$$

Hence, the concentric wave is decaying faster for larger values of τ , resulting in smaller errors, since already the wave interacting with the boundary is smaller. The DABC gives satisfying results for all relaxation times.

The final result presented for the current test case fixes all parameters as before except for the initial fluid velocity \vec{u}_0 . The velocity component tangential to the boundary ($u_{0,\beta}$) is zero, whereas the component

	Error $N_v(t)$ at			
	$t = t_{175}$	$t = t_{250}$	$t = t_{325}$	$t = t_{400}$
Zou/He pressure	0.277485	0.359907	0.358679	0.345152
Grad's	0.074973	0.087297	0.080632	0.076281
CBC	0.009046	0.036197	0.048954	0.053523
DABC H=4, (7)/(8)	0.009193	0.041705	0.055181	0.059633
DABC H=4, (13)	0.018250	0.033751	0.042891	0.047084
DABC H=10, (7)/(8)	0.006549	0.036287	0.049870	0.054805
DABC H=10, (13)	0.012309	0.034344	0.045440	0.049971
DABC H=20, (7)/(8)	0.004842	0.034892	0.049417	0.054755
DABC H=20, (13)	0.009487	0.035349	0.047364	0.052022
DABC H=40, (7)/(8)	0.001232	0.032889	0.051021	0.057585
DABC H=40, (13)	0.004922	0.039249	0.052736	0.057175
DABC H=80, (7)/(8)	0.000001	0.016820	0.046855	0.056847
DABC H=80, (13)	0.000021	0.046725	0.071642	0.075620

TABLE 2. Errors N_v for different boundary conditions and time levels. First test case (Gaussian pressure pulse), $\omega = 1$ and $\vec{u}_0 = 0$.

	Error $N_w(t)$ at			
	$t = t_{175}$	$t = t_{250}$	$t = t_{325}$	$t = t_{400}$
Zou/He pressure	0.078614	0.183191	0.225385	0.242753
Grad's	0.016630	0.045819	0.049365	0.050131
CBC	0.032066	0.050150	0.063324	0.074517
DABC H=4, (7)/(8)	0.014582	0.036721	0.056772	0.072032
DABC H=4, (13)	0.005688	0.028873	0.047478	0.061502
DABC H=10, (7)/(8)	0.007677	0.029861	0.050469	0.066239
DABC H=10, (13)	0.004140	0.028261	0.049035	0.064021
DABC H=20, (7)/(8)	0.002870	0.024961	0.047393	0.064187
DABC H=20, (13)	0.003158	0.027737	0.049795	0.065342
DABC H=40, (7)/(8)	0.000325	0.018999	0.044452	0.063156
DABC H=40, (13)	0.001496	0.028169	0.052968	0.069439
DABC H=80, (7)/(8)	0.000000	0.005729	0.032076	0.052336
DABC H=80, (13)	0.000006	0.022848	0.056737	0.075456

TABLE 3. Errors N_w for different boundary conditions and time levels. First test case (Gaussian pressure pulse), $\omega = 1$ and $\vec{u}_0 = 0$.

		$N_\rho(t_{400})$	$N_v(t_{400})$	$N_w(t_{400})$
$\tau = 0.60$	Zou/He pressure	0.974653	0.464104	0.324983
	Grad's	0.195302	0.101006	0.059928
	CBC	0.172067	0.065029	0.092413
	DABC H=20, (7)/(8)	0.183732	0.071039	0.086858
	DABC H=20, (13)	0.178314	0.067579	0.086631
$\tau = 0.80$	Zou/He pressure	0.822321	0.391664	0.275049
	Grad's	0.167058	0.085893	0.054233
	CBC	0.150659	0.058254	0.082080
	DABC H=20, (7)/(8)	0.152543	0.060856	0.072713
	DABC H=20, (13)	0.148096	0.057966	0.073507
$\tau = 1.25$	Zou/He pressure	0.639278	0.305219	0.214941
	Grad's	0.133036	0.068092	0.046461
	CBC	0.121525	0.049330	0.067567
	DABC H=20, (7)/(8)	0.117714	0.049754	0.057141
	DABC H=20, (13)	0.112726	0.047065	0.058467
$\tau = 1.50$	Zou/He pressure	0.578431	0.276624	0.195010
	Grad's	0.121745	0.062277	0.043783
	CBC	0.111281	0.046274	0.062339
	DABC H=20, (7)/(8)	0.106612	0.046287	0.052214
	DABC H=20, (13)	0.101099	0.043626	0.053633

TABLE 4. Errors N_ρ , N_v and N_w for different boundary conditions and relaxation times. First test case (Gaussian pressure pulse), with $\vec{u}_0 = 0$.

perpendicular to the boundary ($u_{0,\alpha}$) is varied. As before, errors are computed at $t = t_{400}$. When considering the errors, shown in Table 5, we see that the DABC, initialized with respect to (13), has clearly smaller errors for positive velocities. However the errors are larger for negative velocities. With initialization according to (7)/(8) the DABC behaves similar to the CBC. For all test cases the fixed pressure boundary condition results in largest errors.

4.2. Flow past an obstacle in channel

In the second test example we simulate a flow past a square obstacle in a channel at $\text{Re} = 100$. The square obstacle has a dimension of $L = 15$ lattice nodes. The width and total length of the channel are $5L$ and $11L$, respectively. It is displaced vertically by $\frac{L}{5}$ lattice nodes from the center of the channel, to break symmetry and allow a vortex street to develop. We place the obstacle $3L$ nodes from the inlet (left boundary), such that there are $7L$ lattice nodes behind the obstacle in direction of the flow. See also Fig. 10 for a visualization of the setting. Normally in a simulation, the right boundary would have a larger distance to the obstacle, but in the scope of testing a boundary condition the choice seems reasonable. At the inlet we impose a parabolic velocity profile with its maximal velocity $u_{\max} = \frac{1}{9}$ in the center, whereas the velocity component in y -direction is set to zero [21]. For the right boundary of the computational domain we test several DABCs and compare the errors (14) with those when using a CBC or exit condition of [23]. We initialize the fluid in the interior of the computational domain with the same parabolic velocity profile and a decreasing density from inlet to outlet. The decay is chosen such that the resulting pressure gradient fits to the stationary solution of the Poiseuille flow when there would not be an obstacle [24].

		$N_\rho(t_{400})$	$N_v(t_{400})$	$N_w(t_{400})$
$u_{0,\alpha} = -0.20$	Zou/He pressure	0.761080	0.360090	0.294364
	Grad's	0.126869	0.060478	0.051424
	CBC	0.082773	0.039248	0.079130
	DABC H=20, (7)/(8)	0.076132	0.037925	0.067832
	DABC H=20, (13)	0.159738	0.081832	0.086171
$u_{0,\alpha} = -0.10$	Zou/He pressure	0.729659	0.341081	0.259046
	Grad's	0.136463	0.067446	0.049855
	CBC	0.107232	0.044476	0.071911
	DABC H=20, (7)/(8)	0.102220	0.044317	0.060727
	DABC H=20, (13)	0.150410	0.063511	0.081461
$u_{0,\alpha} = -0.05$	Zou/He pressure	0.695514	0.328143	0.239728
	Grad's	0.137487	0.069343	0.048435
	CBC	0.116508	0.046956	0.068698
	DABC H=20, (7)/(8)	0.112323	0.047139	0.057872
	DABC H=20, (13)	0.134400	0.055193	0.071381
$u_{0,\alpha} = 0.05$	Zou/He pressure	0.606929	0.293510	0.198921
	Grad's	0.131049	0.067892	0.045595
	CBC	0.120947	0.051021	0.062830
	DABC H=20, (7)/(8)	0.126173	0.054359	0.059037
	DABC H=20, (13)	0.096058	0.041342	0.047969
$u_{0,\alpha} = 0.10$	Zou/He pressure	0.556697	0.273143	0.177444
	Grad's	0.123929	0.064791	0.043658
	CBC	0.125892	0.055574	0.059419
	DABC H=20, (7)/(8)	0.129904	0.058062	0.057923
	DABC H=20, (13)	0.077947	0.035392	0.037626
$u_{0,\alpha} = 0.20$	Zou/He pressure	0.453472	0.230098	0.134056
	Grad's	0.107946	0.056787	0.039634
	CBC	0.136904	0.065509	0.054637
	DABC H=20, (7)/(8)	0.133179	0.064572	0.051532
	DABC H=20, (13)	0.049582	0.025437	0.022306

TABLE 5. Errors N_ρ , N_v and N_w for different boundary conditions and initial fluid velocities \vec{u}_0 . First test case (Gaussian pressure pulse).

We tested several versions of the DABC. They differ in the choice of the subproblem's initialization and the maximal history depth. In the following we refer to the different initialization strategies by DABC- n , $n \in \{1, 2, 3, 4\}$. Hereby, DABC-1 uses the strategy (7) with $\rho^s \equiv \rho(\vec{x}_b, t_0)$ and $\vec{u}^s = \vec{0}$. The second initialization type DABC-2 is given by (8), and DABC-3 is using the modification (13). Finally, DABC-4 can be written shortest in terms of the reference solution

$$h_i^s(\vec{x}_m, t_0^s) = f^{\text{ref}}(\vec{x}_m, t_0),$$

which, thereby, is very similar to DABC-2. The sole exception is that the density is decreasing in DABC-4, whereas it has a constant level in DABC-2. Note, that the choice DABC-4 is not requiring a reference solution,

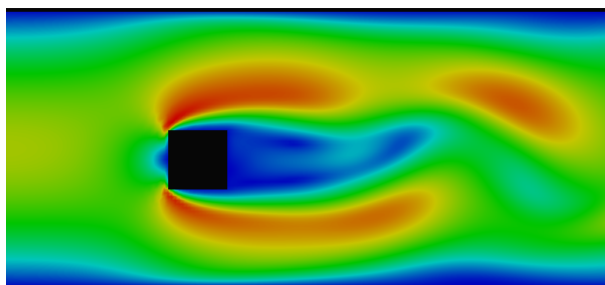


FIGURE 10. Simulation setting of the second test case. The snapshot shows the modulus of the velocity and displays a vortex.

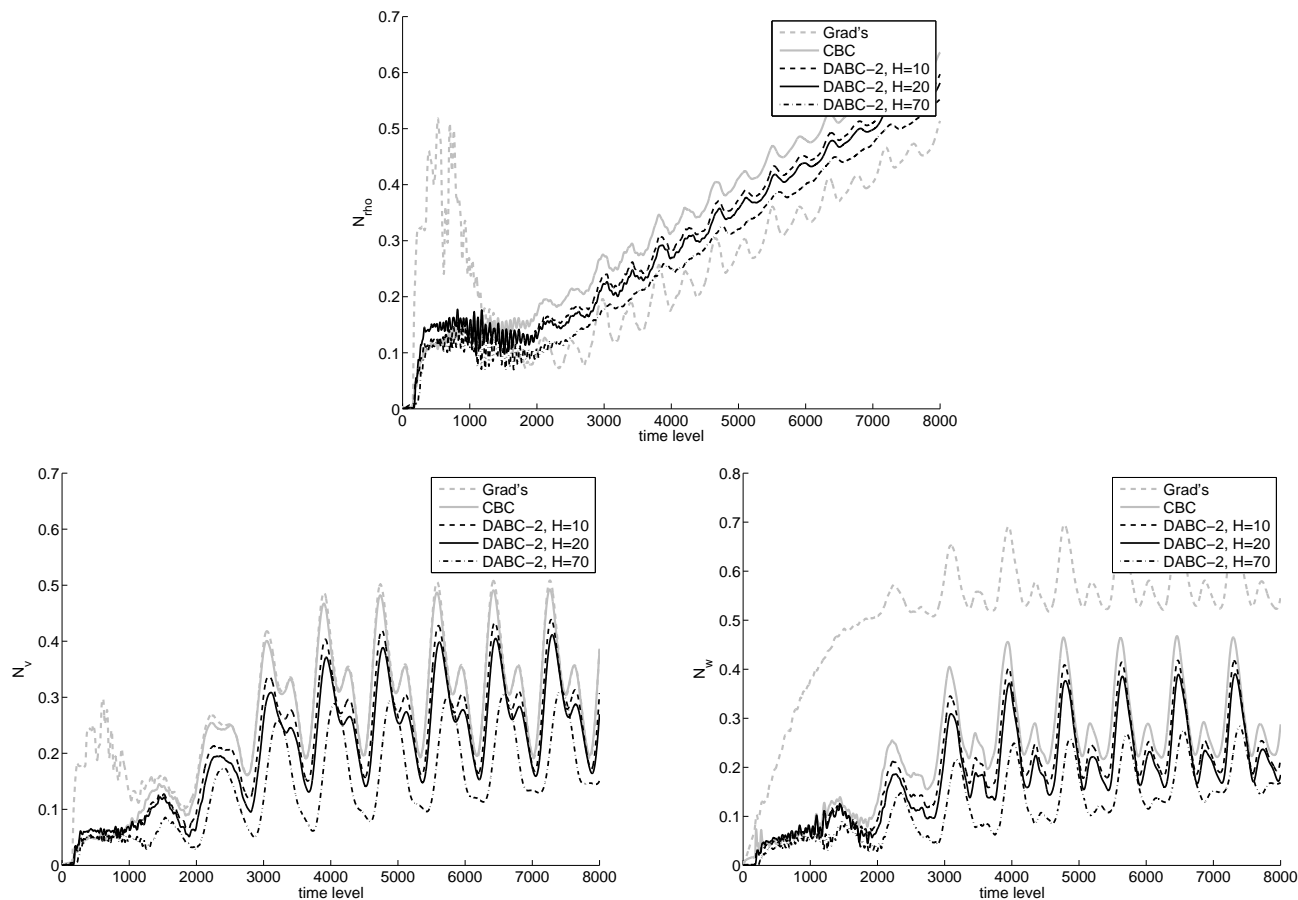


FIGURE 11. Second test case: Normalized errors (14) (N_ρ , N_v and N_w) for DABCs with different maximal history depth. Initialization due to strategy DABC-2. Three different maximal history depths H_{\max} were tested (10, 20 and 70). Also errors of a CBC (LODI) and Grad's based exit condition are shown.

it is only a logical extension of the interior initialization. Also note, that DABC-1 and DABC-2 coincided in the previous test case, but differ in the current test.

The plots in Fig. 11 show the errors (14) of DABC-2 using different maximal history depths. We clearly

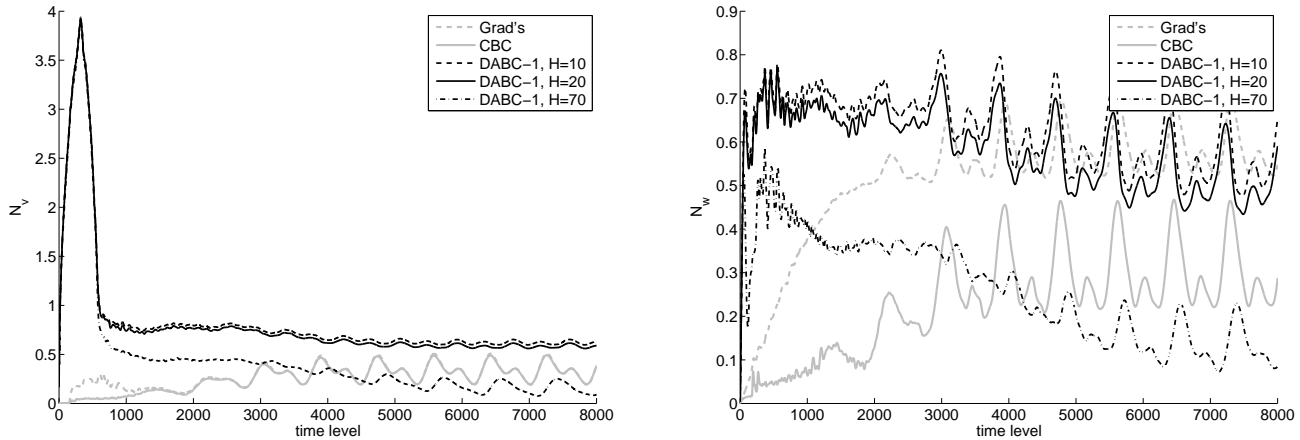


FIGURE 12. Second test case: Normalized errors (14) (N_v and N_w) for DABC-1 and three maximal history depths H_{\max} (10, 20 and 70). Plots also contain errors of a CBC (LODI) and Grad's based exit condition.

observe, that higher history depths result in smaller errors. They all are smaller than the reference error of a non-reflecting CBC. Although a difference between errors of DABC-2 and DABC-4 could not be detected visually, the errors of DABC-2 are a minimal smaller. The error plots of DABC-4 are not shown, since they look equally to DABC-2. As in the one-dimensional case, DABC-3 shows instabilities and thus corresponding error plots are omitted.

Already at the beginning of the simulation, using the DABC-1 there is a pressure wave generated at the right boundary traveling into the domain. By this, the whole density level is increased and remains on a higher niveau. This makes a consideration of N_ρ worthless. The pressure wave also affects the velocity profile for v , as one can see in the left plot of Fig. 12. As before, the same influence of the maximal history depth is also visible for DABC-1. The errors are significantly higher compared to DABC-2 and DABC-4.

All in all, in both test cases the initialization strategy (8) (DABC-2 in the second test case) produces good results.

5. CONCLUSIONS

In this work, the one-dimensional discrete artificial boundary condition (DABC) for the lattice Boltzmann (LB) method [11] was successfully transferred to two space dimensions. Our formulation of the DABC was done in a general way, but we set a special focus on the D2Q9 LB model. For the implementation of the DABC, the LB method was equipped with a so-called subproblem in each time step, which offers a free parameter, called the history depth. This tuning parameter determines the number of past time levels, which are taken into account in the subproblems. In addition, this parameter fixes the size of the subproblems, and thus it determines both, the accuracy and the computational effort of the DABC. Comparing the DABC with an ideal transparent boundary condition, any error is caused already in the initialization phase of these subproblems.

In the one-dimensional case, the history depth could be tuned to control the error. We presented two test cases, a two-dimensional Gaussian pressure pulse and a flow past an obstacle in a channel. The results of the second test case demonstrated that the history depth controls the error also for the two-dimensional DABC. By the first test, we could explain the working principle of the DABC: the solutions of the subproblems are shaped in such a way that they virtually extend the computed solution of the (actual) computational domain. Then, for the boundary of the computational domain, all missing information are taken from the subproblems.

The numerical tests showed that our proposed initialization strategy (8) leads to errors smaller than those obtained by characteristic boundary conditions.

REFERENCES

- [1] D. Haydock, J. M. Yeomans, *Lattice Boltzmann simulations of acoustic streaming*, Journal of Physics A: Mathematical and General 34 (25) (2001) 5201–5213.
- [2] C. Sun, L. L. Munn, *Lattice-Boltzmann simulation of blood flow in digitized vessel networks*, Computers & Mathematics with Applications 55 (7) (2008) 1594–1600.
- [3] S. Geller, J. Tölke, M. Krafczyk, *Fluid-Structure Interaction: Modelling, Simulation, Optimisation*, Vol. 53 of Lecture Notes in Computational Science and Engineering, Springer Verlag, Berlin, Germany, 2006, Ch. Lattice-Boltzmann Method on Quadtree-Type Grids for Fluid-Structure Interaction, pp. 270–293.
- [4] M. Ehrhardt, *Absorbing boundary conditions for hyperbolic systems*, Numerical Mathematics: Theory, Methods and Applications 3 (3) (2010) 295–337.
- [5] G. W. Hedstrom, *Nonreflecting boundary conditions for nonlinear hyperbolic systems*, Journal of Computational Physics 30 (2) (1979) 222–237.
- [6] K. W. Thompson, *Time dependent boundary conditions for hyperbolic systems*, Journal of Computational Physics 68 (1) (1987) 1–24.
- [7] S. Izquierdo, N. Fueyo, *Characteristic nonreflecting boundary conditions for open boundaries in lattice Boltzmann methods*, Physical Review E 78 (2008) 046707.
- [8] D. Kim, H. M. Kim, M. S. Jhon, S. J. Vinay III, J. Buchanan, *A characteristic non-reflecting boundary treatment in lattice Boltzmann method*, Chinese Physics Letters 25 (6) (2008) 1964.
- [9] D. Heubes, A. Bartel, M. Ehrhardt, *Characteristic boundary conditions in the lattice Boltzmann method for fluid and gas dynamics*, Journal of Computational and Applied Mathematics 262 (2013) 51–61.
- [10] D. Heubes, A. Bartel, M. Ehrhardt, *Exact artificial boundary conditions for a lattice Boltzmann method*, Computers & Mathematics with Applications 67 (11) (2014) 2041–2054.
- [11] D. Heubes, A. Bartel, M. Ehrhardt, *Concept for a one-dimensional discrete artificial boundary condition for the lattice Boltzmann method*, Computers & Mathematics with Applications (2014), submitted.
- [12] Y. Qian, D. d’Humières, P. Lallemand, *Lattice BGK Models for Navier-Stokes Equation*, Europhysics Letters 17 (6) (1992) 479–484.
- [13] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Oxford University Press, Oxford, UK, 2001.
- [14] D. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models*, Springer Verlag, Berlin, Germany, 2000.
- [15] D. Heubes, A. Bartel, M. Ehrhardt, *An Introduction to the Lattice Boltzmann Method for Coupled Problems*, pp. 3–30 in [25].
- [16] P. L. Bhatnagar, E. P. Gross, M. Krook, *A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems*, Physical Review 94 (3) (1954) 511–525.
- [17] D. d’Humières, *Generalized lattice-Boltzmann equations*, Rarefied gas dynamics – Theory and simulations (1994) 450–458.
- [18] I. Ginzburg, F. Verhaeghe, D. d’Humières, *Two-relaxation-time lattice Boltzmann scheme: About parametrization, velocity, pressure and mixed boundary conditions*, Communications in Computational Physics 3 (2008) 427–478.
- [19] S. Chapman, T. Cowling, *The Mathematical Theory of Non-Uniform Gases*, Cambridge University Press, London, UK, 1970.
- [20] M. Junk, A. Klar, L.-S. Luo, *Asymptotic analysis of the lattice Boltzmann equation*, Journal of Computational Physics 210 (2) (2005) 676–704.
- [21] Q. Zou, X. He, *On pressure and velocity boundary conditions for the lattice Boltzmann BGK model*, Physics of Fluids 9 (6) (1997) 1591–1598.
- [22] T. Inamuro, M. Yoshino, F. Ogino, *A non-slip boundary condition for lattice Boltzmann simulations*, Physics of Fluids 7 (1995) 2928–2930.
- [23] S.S. Chikatamarla, S. Ansumali, I.V. Karlin, *Grad’s approximation for missing data in lattice Boltzmann simulations*, Europhysics Letters 74 (2) (2006) 215–221.
- [24] G. K. Batchelor, *An introduction to fluid dynamics*, Cambridge university press, 2000.
- [25] M. Ehrhardt (Ed.), *Novel Trends in Lattice Boltzmann Methods*, Progress in Computational Physics, Volume 3, Bentham Science Publishers Ltd., 2013.