Evaluating the Impact of Boundary Conditions on the MR-LBM

Marco A.Ferrari[1], Luiz A. Hegele Jr [2], Admilson T.Franco[1]

[1] *Research Center for Rheology and Non-Newtonian Fluids (CERNN), Postgraduate Program in Mechanical and Materials Engineering, Federal University of Technology – Paraná (UTFPR)*
Rua Deputado Heitor Alencar Furtado, 5000, 81280-340,Paraná , Brazil
marcoferrari@alunos.utfpr.edu.br, admilson@utfpr.edu.br
[2]*Department of Petroleum Engineering, Santa Catarina State University (UDESC)*
Av. Lourival Cesario Pereira, s/n, 88336-275, Balneário Camboriú - Santa Catarina, Brazil
luiz.hegele@udesc.br

Abstract. The fluid flow investigation in heterogeneous porous media has several applications, such as in the oil and gas industry. One of the problems is determining the permeability of the porous media. However, numerical studies often employed simple geometric forms (spheres, cubes, cylinders) to represent the porous media. Complex meshes are usually required when scans are employed to describe the porous media. The Lattice Boltzmann Method (LBM) can address this problem without complicated meshes. Recently, the moment representation of the LBM has gained interest due to its reduced memory requirements and increased speed. The reduced memory usage is achieved by storing moments from 0th to 2nd order instead of the mesoscopic populations. This change also reduces the bandwidth usage between the memory and the processor, which is a bottleneck for the LBM, thereby increasing performance. However, boundary lattices will require additional arithmetic operations when conditions other than a periodic need to be applied, leading to speed degradation. In simulations of a heterogeneous porous media, many lattices are subjected to boundary conditions, and the probability of having a neighbor lattice affected by it increases with a reduction in porosity. Additionally, cases with the same solid volume fraction can exhibit different performances. This discrepancy arises because the effect of the boundary condition in the processing is associated with the wet surface area. To investigate these parameters (solid volume fraction and wet surface area), we build heterogeneous porous media using a Gaussian blur over a randomly generated domain. We varied the porosity by tuning the threshold value while the standard deviation changed the surface area. The results demonstrate that as the porosity decreases, the computational performance also decreases. However, once the porosity reaches a certain threshold, the execution time decreases due to reduced total wet surface area, and the number of fluid nodes reduces. Additionally, there is a direct correlation between the computational speed and the standard deviation of the Gaussian blur for the wet surface area. With more lattices neighboring other solid lattices, there is a reduction of boundary conditions being applied as well as the number of collision-streaming steps being performed, resulting in improved performance.

Keywords: lattice Boltzmann method, moment representation, boundary condition, performance, porous media.

1 Introduction

The lattice Boltzmann method is a numerical method that can solve fluid flows differently from the finite or volume element methods. It solves the flow at the mesoscopic scale [1,2]. Over the years, various studies have been performed to increase computational efficiency, especially using GPUs. These improvements have either focused on different streaming patterns [3–6], memory layouts [5–9], or numerical precision [10]. In these studies, the performance was often assessed without boundary conditions, significantly increasing the processing time as additional arithmetic operations were required. This case is frequently encountered when modeling porous media. Different methods were proposed to reduce the memory footprint in sparse domains. Tomczak and Szafran [11] proposed tiling the domain and consequently eliminating the necessity of computing certain lattices when all the nodes in the tile are solid. They also proposed a memory layout in case only specific latices of the tile are solid. Namvar and Leclaire [12] applied array shifting to eliminate the tiles and only compute the fluid nodes, achieving reduced memory usage and improved performance as the volume fraction decreased. An alternative to reduce the memory footprint is the moment representation of the lattice Boltzmann method (MR-LBM) [9,13,14]. Unlike conventional LBM, it works with the moments from 0th to 2nd order instead of populations. This formulation's advantage is the possibility of reducing global memory usage and increasing processing speed [14]. Because it is a new method, there has been little investigation regarding how the boundary conditions affect the computing time. Some information on how the performance in the MR-LBM is affected by boundary conditions can be found in the work of Gounley et al.[9], where two real problems are compared: aortic vs. cerebral. However, their investigation focused on the impact of computational parameters instead of the number of lattices affected by boundary conditions. Therefore, this work aims to investigate the effects of the moment-based boundary conditions on MR-LBM using a representation of porous media as a test environment.

2 Numerical method

In this work, we utilize the MR-LBM [9,13,14]. The algorithm begins by loading the moments from the global memory and reconstructing the populations using the following equation:

$$f_i(\mathbf{x},t) = \rho w_i \left(1 + a_s^2 u_\alpha c_{i\alpha} + \frac{1}{2} a_s^4 m_{\alpha\beta}^{(2)} \mathcal{H}_{\alpha\beta,i}^{(2)}\right), \tag{1}$$

where ρ, u, and $m^{(2)}$ represent the moments from 0th to 2nd order respectively. $\mathcal{H}_{\alpha\beta,i}^{(2)}$ denotes the Hermite polynomial of the second order, a_s is the scale factor, and is inversely proportional to the speed of the sound c_s, and w_i represents the quadrature weights, both of which depend on the velocity set used. In this study, we employed the D3Q19 velocity set, which consists of the following values:

$$\mathbf{c}_i = \begin{cases} (0,0,0) & ,i=0 \\ (\pm1,0,0)(0,\pm1,0)(0,0,\pm1) & ,i=1,\ \dots,\ 6 \\ (\pm1,\pm1,0)(\pm1,0,\pm1)(0,\pm1,\pm1), & i=7,\ \dots,\ 18 \end{cases},\ w_i = \begin{cases} 1/3 & ,i=0 \\ 1/18 & ,i=1,\ \dots,\ 6 \\ 1/36 & ,\ i=7,\ \dots,\ 18 \end{cases}, \tag{2}$$

The population reconstruction step is crucial as it enables the subsequent streaming step. These populations are stored in the processor cache, not increasing global memory usage. Following the streaming process completed, the post-streaming moments are calculated as:

$$\begin{aligned} \rho &= \sum_i f_i \\ \rho \bar{u}_\alpha &= \sum_i f_i c_{i\alpha} \\ \rho m_{\alpha\beta}^{(2)} &= \sum_i f_i \mathcal{H}_{\alpha\beta,i}^{(2)} \end{aligned} \quad . \tag{3}$$

If a boundary condition needs to be imposed, it overwrites the moments determined by Eq.(3). In this research, we employ the moment-based boundary conditions of Hegele et al. [15]. This method involves summing the known and unknown populations and computing the second-order moment:

$$\sum_{i\in I_s} f_i \mathcal{H}_{\alpha\beta,i}^{(2)} + \sum_{i\notin I_s} \hat{f}_i \mathcal{H}_{\alpha\beta,i}^{(2)} = \rho m_{\alpha\beta}^{(2)} \tag{4}$$

where I_s represents all the incoming populations in the lattice. For a Dirichlet boundary condition, this generates six equations for seven unknown variables (0th and 2nd order moments). The last equation is obtained by conserving the mass during the collision process:

$$\sum_{i\in I_s} f_i = (1-\omega)\sum_{i\in O_s} \hat{f}_i + \omega \sum_{i\in O_s} f_i^{(eq)} \tag{5}$$

This set of equations must then be solved to determine the moment values, which will overwrite the previous values that were determined during the streaming step. Following that, the collision step is executed:

$$\rho^* u_\alpha^* = \sum_i f_i^*(\mathbf{r},t) c_{i\alpha} = (1-\omega)\rho u_\alpha + \omega \rho u_\alpha + \left(1-\frac{\omega}{2}\right) F_\alpha, \tag{6}$$

$$\rho^{*} m_{\alpha\beta}^{*(2)} = \left(1-\omega\right) \rho^{*} m_{\alpha\beta}^{(2)} + \omega\rho u_{\alpha} u_{\beta} + \left(1-\frac{\omega}{2}\right)\left(F_{\alpha} u_{\beta} + F_{\beta} u_{\alpha}\right). \tag{7}$$

where ω represents the relaxation frequency, which depends on the fluid viscosity and is defined $\omega = (\nu a_s^2+1/2)^{-1}$. The post-collision moments are then stored in the global memory, and the timestep is completed. An implementation challenge arises from the impracticality of solving the system formed by Eqs. (4) and (5) at each time step. Thus, it is preferable to compute all 2^8 possible boundary combinations and their respective solutions using a symbolic computation package in a pre-processing step.

The number 2^8 arises from the fact that each lattice can be subdivided into eight voxels, each representing a corner of the lattice. The combination of these voxels determines whether the boundary condition will be a wall, corner, or edge type, as illustrated in Figure 1. One or multiple voxels will be solid if the neighboring lattice in that direction represents a solid material. Hence, a lattice without boundary conditions can also be defined as a lattice with all eight voxels deactivated. Constructing the domain based on these voxels makes it possible to encompass all potential situations of boundary conditions that may arise when solving a porous media problem.

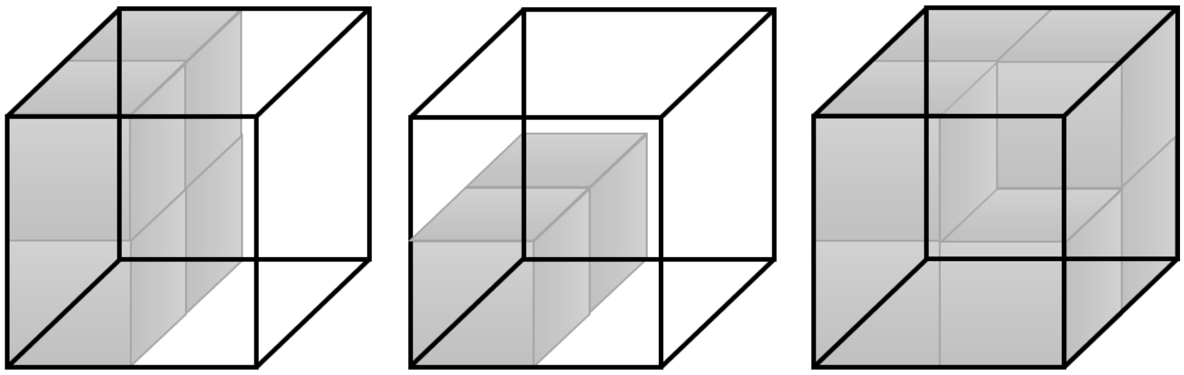


Figure 1-Representation of voxels structures inside one lattice to represent a wall, convex edge, and concave corner.

Finally, it is necessary to define how to create a porous structure with controlled porosity and the number of lattices affected by the boundary conditions. This study achieves this by applying a Gaussian blur to a randomly generated domain [16]. Initially, each lattice in the domain is assigned a random number between 0 and 1. We then compute the average value between the neighboring lattices by applying the Gaussian blur over the domain. If the resulting average value is less than a threshold value *x*, we assign the value of zero to the lattice. Otherwise, we assign the value of one [1]. Using a Gaussian blur increases the likelihood that neighboring lattices will have similar values. In this work, the size of the Gaussian blur is determined by 4σ+1 in each coordinate direction, where σ represents the standard deviation of the blur. For example, if $\sigma = 10$, the filter will be applied to 41^3 lattices.

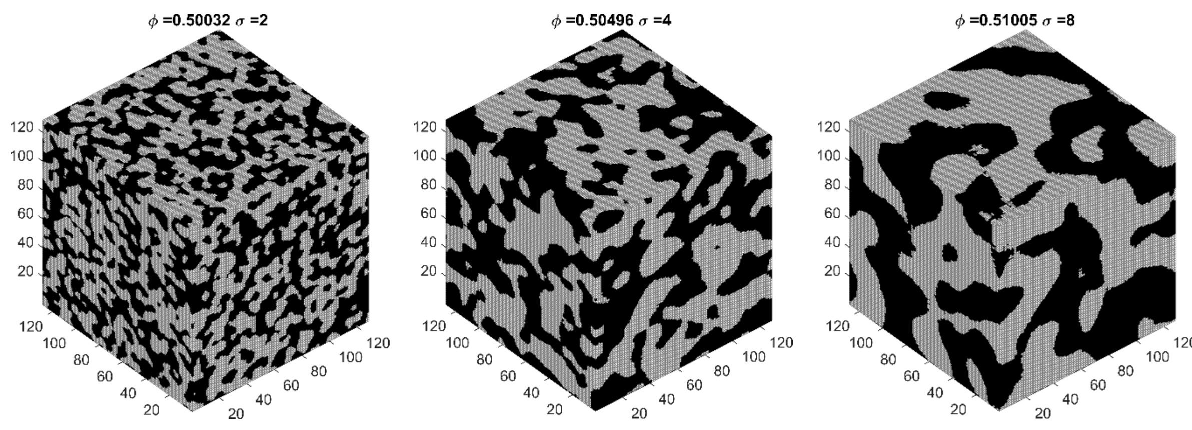
However, if we need a specific porosity value $\phi \neq 0.5$, the threshold value *x* must differ from ϕ. This is because increasing the value σ will cause the generated porosity ϕ to lean towards either 0 or 1, as the probability for the average value obtained through the Gaussian blur filter to be equal to the total average value of the domain reduces drastically. To address this issue, we conducted preliminary tests with different values of *x* and σ to determine the average porosity ϕ obtained. The results indicated that these values be approximated as follows:

$$\phi = x e^{\left\{-\frac{\sigma^2}{2}\left[\left(\frac{0.024}{1-2x}\right)+1.91\right]^{-2}\right\}}, \tag{8}$$

where *x* represents the threshold value, and σ is the standard deviation for the Gaussian filter to achieve the desired porosity ϕ. Equation (8) has to be solved with an iteratively for *x* in the interval (1, 0.5). If the desired porosity ϕ

[1] We differentiate from Ávilla et al. [16], which normalized to 0 and 1 on the last step, and moved the threshold value until the correct porosity was achieved. .

is lower than 0.5, it is necessary to use 1- ϕ instead and invert the values of 0 and 1 at the end. For ϕ = 0.5, the value of σ does not affect the porosity, as the domain is expected to be evenly balanced, and therefore x = *0.5*. Figure 2 showcases the domains created with different values of σ for the ϕ = 0.5. It illustrates that as σ increases, both the porous size and the variance in ϕ, resulting from the stochastic process, also increase.



ϕ =0.50032 σ =2
ϕ =0.50496 σ =4
ϕ =0.51005 σ =8

Figure 2 - Examples of domains (N=128) built with different values of σ (2, 4, and 8) for x = 0.5.

3 Results

3.1 Verification

We verify our numerical method by considering the cubic lid-driven cavity problem. In this problem, a cubic domain with a side length of N lattices is used, and Dirichlet boundary conditions are applied to all surfaces. The walls have a velocity equal to zero, except for one, where it has a tangential velocity in the x-direction equal to U_0. We compare the results for two Reynolds numbers: 100 and 1480. For Re = 100, we use the numerical results from Shu et al. [16]. For Re = 1480, we compare our results with the experimental findings of Liberzon et al. [17] and the numerical results of Zhang et al. [18]. The comparison is made based on the velocity profiles as a function of the mesh refinement along the (x, 0.5, 0.5) and (0.5, y, 0.5) axes, as shown in Figure 3.

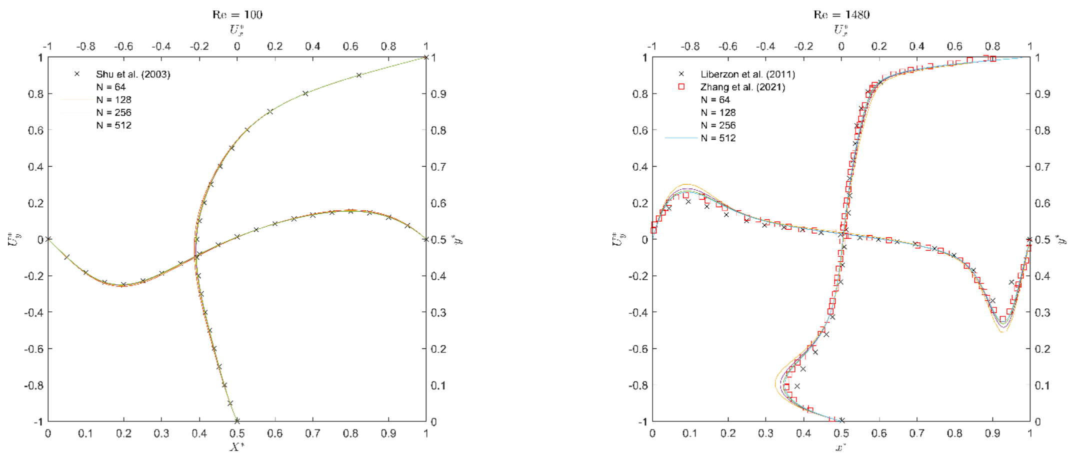


Figure 3 – Velocity profiles along the axis (x,0.5,0.5) and (0.5,x,0.5) for the cubic lid-driven cavity problem and comparison with literature data.

The results demonstrate the satisfactory accuracy achieved by the moment-based boundary conditions. As the mesh is refined, the velocity profiles converge toward the previously obtained results obtained in the literature. The most considerable discrepancies occur in the edges of the central vortex as the Reynolds number increases. However, these differences are promptly eliminated with mesh refinement.

3.2 Performance results in porous media

We used an NVIDIA RTX 3060 12GB as a test device to obtain the results. Previous work [14] shows that this device has a good balance between computing power and bandwidth, allowing it to reach 99% of the GPU and memory controller load capacity when using single-point precision. Therefore, any performance loss due to boundary conditions can be easily identified. We evaluated three mesh sizes (64, 128, and 256) for five porosities and four values of σ in Eq. (8). The results, shown in Figure 4, were obtained by determining the mean MLUPS (million lattice updates per second) of 3 runs with 10,000-time steps each. Additionally, we compared the results with those of the cubic lid-driven cavity.

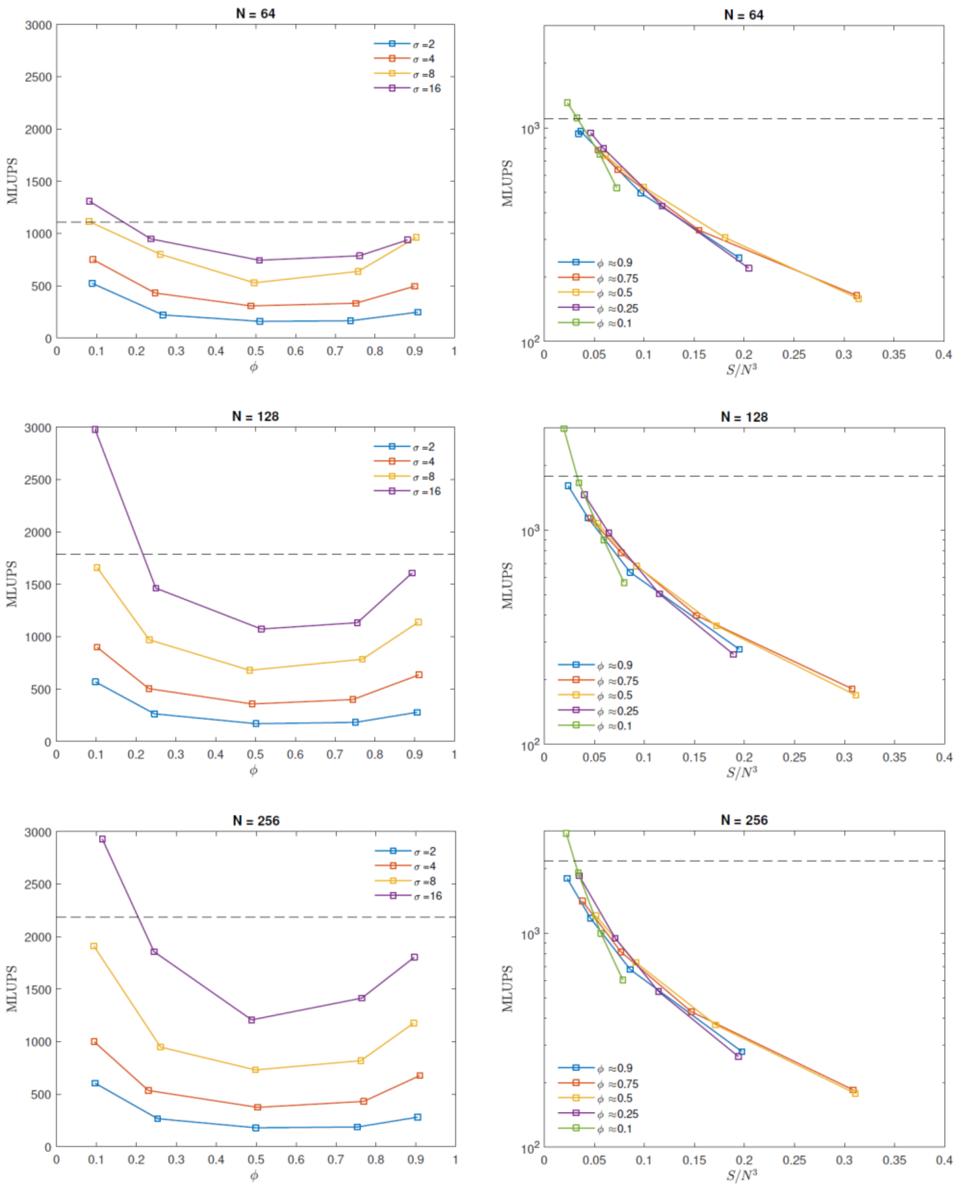


Figure 4 – Performance as a function of porosity, mesh size for different values of σ, and ratio of boundary nodes (S) with total nodes (N^3). The dashed line represents the result of a cubic lid-driven cavity problem.

The results indicate that as the porosity decreases, the computing speed also decreases, reaching its minimum at $\phi = 0.5$. This point also coincides with the highest number of nodes affected by boundary conditions.

However, as the porosity decreases further, the performance increases due to the increased number of solid nodes that do not require computation. In some instances, it even surpassed the lid-driven cavity, where only the lattices on the surface of the domain have boundary conditions.

By increasing the value of σ, there is also an increase in the processing speed because σ directly increases the probability of neighboring lattices having the same definition (solid or liquid). As a result, the ratio of lattices affected by boundary conditions is significantly reduced. To illustrate, consider a solid lattice surrounded by fluid nodes. In this case, 26 lattices would have boundary conditions. However, if this solid lattice is part of a plane wall, it would only cause, on average, one lattice to have boundary conditions, reducing the total number of arithmetic operations. Another consequence is the reduction in warp branching, where fluid lattices without boundary conditions in the same thread with ones that have boundary conditions still should perform the boundary conditions calculations that will no longer be needed. This effect is more evident on the right side of Figure 4, where the x-axis represents the percentage of lattices with boundary conditions out of the total number of lattices. There is a clear indication of a performance curve that strongly depends on this ratio. For lower ratio values, the computing speed approaches extremely high MLUPs, which coincide with a high number of solid nodes. Conversely, as the ratio of boundary nodes increases, it should asymptotically approach the MLUPs of a hypothetical situation where all the nodes have boundary conditions.

The domain size also plays a crucial role in computing speed. For instance, when N = 64, there is a reduction in MLUPs. However, it asymptotically approaches a maximum value as the domain size increases. In this case, N = 128 and 256 yield similar results, a behavior commonly observed in other performance analyses of LBM documented in the literature [11,19]

4 Conclusions

This study focused on evaluating the impact of boundary conditions on the moment representation of the lattice Boltzmann method. The investigation used a heterogenous porous media generated by applying a Gaussian blur to a randomly generated domain. The findings highlighted that the method's performance is influenced by the number of lattices affected by boundary conditions and the number of fluid lattices. The worst-case scenario occurred when the porosity of the media was 50%, resulting in the maximum number of nodes affected by boundary conditions. The impact of boundary nodes is mitigated by observing that increasing the Gaussian blur's standard deviation helped reduce the number of boundary nodes while maintaining a constant porosity. Therefore, selecting the highest feasible standard deviation is recommended as long as it does not adversely affect the flow structures of interest in the porous media. Additionally, a suggestion for future work is to investigate the impact of indirect addressing on computing speed and overall memory usage.

Acknowledgments.

M. A. Ferrari acknowledges the financial support by the Human Resource Program of The Brazilian National Agency for Petroleum, Natural Gas, and Biofuels – PRH-ANP – FINEP Management, contractual instrument n° 0.1.19.0240.00, Ref 0431/19 - PRH21-UTFPR. L.A. Hegele Jr. would like to thank Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC) through grant number 2021TR789. The authors acknowledge the financial support of the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES), Finance Code 001.

Authorship statement.

The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, 2001.
[2] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E.M. Viggen, The Lattice Boltzmann Method, 1st ed., Springer International Publishing, Cham, 2017.
[3] M. Geier, M. Schönherr, Esoteric Twist: An Efficient in-Place Streaming Algorithmus for the Lattice Boltzmann Method on Massively Parallel Hardware, Computation. 5 (2017) 19.
[4] M. Lehmann, Esoteric Pull and Esoteric Push: Two Simple In-Place Streaming Schemes for the Lattice Boltzmann Method on GPUs, Computation. 10 (2022) 92.
[5] M. Mohrhard, G. Thäter, J. Bludau, B. Horvat, M.J. Krause, Auto-vectorization friendly parallel lattice Boltzmann streaming scheme for direct addressing, Comput. Fluids. 181 (2019) 1–7.
[6] A. Perepelkina, V. Levchenko, A. Zakirov, New Compact Streaming in LBM with ConeFold LRnLA Algorithms, in: Commun. Comput. Inf. Sci., Springer International Publishing, 2020: pp. 50–62.
[7] P. Valero-Lara, F.D. Igual, M. Prieto-Matías, A. Pinelli, J. Favier, Accelerating fluid-solid simulations (Lattice-Boltzmann and Immersed-Boundary) on heterogeneous architectures, J. Comput. Sci. 10 (2015) 249–261.
[8] J. Latt, C. Coreixas, J. Beny, Cross-platform programming model for many-core lattice Boltzmann simulations, PLoS One. 16 (2021) e0250306.
[9] J. Gounley, M. Vardhan, E.W. Draeger, P. Valero-Lara, S. V. Moore, A. Randles, Propagation pattern for moment representation of the lattice Boltzmann method, IEEE Trans. Parallel Distrib. Syst. 33 (2022) 642–653.
[10] M. Lehmann, M.J. Krause, G. Amati, M. Sega, J. Harting, S. Gekle, On the accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit and novel 16-bit number formats, (2021).
[11] T. Tomczak, R.G. Szafran, A new GPU implementation for lattice-Boltzmann simulations on sparse geometries, Comput. Phys. Commun. 235 (2019) 258–278.
[12] M. Namvar, S. Leclaire, Simple lattice Boltzmann method algorithm with low memory usage, J. Comput. Sci. 62 (2022) 101723.
[13] M. Vardhan, J. Gounley, L. Hegele, E.W. Draeger, A. Randles, Moment representation in the lattice Boltzmann method on massively parallel hardware, in: Proc. Int. Conf. High Perform. Comput. Networking, Storage Anal., ACM, New York, NY, USA, 2019: pp. 1–21.
[14] M.A. Ferrari, W.B. de Oliveira, A. Lugarini, A.T. Franco, L.A. Hegele, A graphic processing unit implementation for the moment representation of the lattice Boltzmann method, Int. J. Numer. Methods Fluids. (2023) 1–14.
[15] L.A. Hegele, A. Scagliarini, M. Sbragaglia, K.K. Mattila, P.C. Philippi, D.F. Puleri, J. Gounley, A. Randles, High-Reynolds-number turbulent cavity flow using the lattice Boltzmann method, Phys. Rev. E. 98 (2018) 043302.
[16] J. Ávila, J. Pagalo, M. Espinoza-Andaluz, Evaluation of geometric tortuosity for 3D digitally generated porous media considering the pore size distribution and the A-star algorithm, Sci. Rep. 12 (2022) 1–22.
[17] A. Liberzon, Y. Feldman, A.Y. Gelfgat, Experimental observation of the steady-oscillatory transition in a cubic lid-driven cavity, Phys. Fluids. 23 (2011).
[18] J.K. Zhang, M. Cui, Z.L. Zuo, S.Y. Luo, J.X. Guo, Z.Z. Qiua, Prediction on steady-oscillatory transition via Hopf bifurcation in a three-dimensional (3D) lid-driven cube, Comput. Fluids. 229 (2021) 105068.
[19] W.B. Oliveira, A. Lugarini, A.T. Franco, Performance Analysis of the Lattice Boltzmann Method Implementation on GPU, XL Ibero-Latin-American Congr. Comput. Methods Eng. (CILAMCE 2019). (2019).