Tensor Network Lattice Boltzmann Method for Data-Compressed Fluid Simulations

Lukas Gross[a,1,2], Elie Mounzer[a], David M. Wawrzyniak[b,1], Josef M. Winter[c], Nikolaus A. Adams[b,c]

[a]*German Research Center for Artificial Intelligence, Robotics Innovation Center, Robert-Hooke-Str. 1, Bremen, 28359, , Germany*
[b]*Technical University of Munich, Munich Institute of Integrated Materials, Energy and Process Engineering, Lichtenbergstr. 4a, Garching, 85748, , Germany*
[c]*Technical University of Munich, School of Engineering and Design, Chair of Aerodynamics and Fluid Mechanics, Boltzmannstr. 15, Garching, 85748, , Germany*

Abstract

Resolving unsteady transport phenomena in geometrically complex domains is traditionally constrained by polynomial scaling of computational cost with spatial resolution. While methods based on tensor-network data representations or matrix-product states (MPS) data encodings have emerged as a technique to systematically reduce degrees of freedom, existing formulations do not extend to complex geometries and complex flow physics. Both capabilities are offered by lattice Boltzmann methods, for which we develop a generalized MPS formulation. This development marks a paradigm shift from classical methods that rely on explicit grid refinement for data reduction. Instead, our approach exploits non-local correlations in the MPS representation to systemically compress the global fluid state directly without modifying the underlying grid. We benchmark the proposed solver against classical LBM using three-dimensional flows through structured media and vascular geometries. The results confirm that the MPS formulation reproduces the reference solution with high fidelity while achieving compression ratios exceeding two orders of magnitude, positioning tensor networks or MPS encodings as a scalable paradigm for continuum mechanics on high-performance GPU hardware.

Keywords: lattice Boltzmann method, matrix product states, CFD

[1]These authors contributed equally to this work.
[2]Corresponding author, email: lukas.gross@dfki.de

1. Introduction

In computational fluid dynamics (CFD), the numerical solution to the Navier-Stokes equations (NSE) is key to dealing with engineering problems and for investigating physical phenomena. Complex flows involve a broad range of scales that need to be resolved directly or approximated. With direct numerical simulation (DNS) the computational mesh resolution is sufficiently high to resolve the smallest possible flow scales, independent of internal scale relations. DNS implies extreme computational effort, which can be reduced by multi-resolution strategies, but nevertheless requires the use of high-performance computing systems. Some of the classical strategies for DNS are gradient-based adaptive mesh refinement [1] and wavelet-based multiresolution algorithms [2, 3, 4, 5]. Mesh-independent data compression methods have found new recent interest with the quantum-inspired introduction of matrix product states (MPS) [6, 7, 8], and the density matrix renormalization group (DMRG) [9] for constituted state encodings with inherent error control. DMRG and its MPS variants belong to a class of algorithms, originally developed for simulations of many-body quantum systems with small entanglement [10, 11]. Gourianov et al. [12] have introduced an algorithm based on MPS decomposition and a DMRG-like approach to solve a variational formulation of the incompressible NSE, drawing an analogy between spatial correlations of quantum states under local Hamiltonians [13] and interscale correlations of turbulent flows. Several extensions have since been published. Kiffner and Jaksch [14] introduced resting and moving wall boundary conditions and proposed a more explicit algorithm using a DMRG-like routine [15] to solve the Poisson equation. Peddinti et al. [16] proposed methods to treat immersed solid boundaries and for efficient evaluation of solutions while using a similar DMRG-like solver for the Poisson equation. An extension to curvilinear grids was proposed by van Hülst et al. [17], and an implementation for GPUs was proposed by Hölscher et al. [18]. Moreover, Ghahremani and Babaee [19] utilized cross interpolation for high-dimensional dynamical systems. Thai Tran et al. [20] developed an MPS-based[3] isogeometric solver for large-scale 3D Poisson problems. MPS-based approaches have so far been developed primarily for data compression of turbulent flows in relatively simple geometries, leaving their applicability

[3] Thai Tran et al. use the term tensor train, which is another name for the same form of tensor decomposition.

to complex domains and non-turbulent flows largely unexplored.

Known for its capabilities of representing geometrically and physically complex fields, the lattice Boltzmann method (LBM) is a widely adopted approach for solving the NSE. In this framework, discrete velocity sets are employed to represent single-particle distribution functions. A low Mach-number approximation is achieved through a multiscale expansion, where the collision term can be formulated as a linear relaxation towards an equilibrium using the Bhatnagar–Gross–Krook (BGK) model [21]. This term accounts for local non-equilibrium effects, thereby introducing nonlinearity through an approximated local equilibrium distribution. Compared to directly solving the NSE, LBM exhibits algorithmic simplicity through operator splitting between linear streaming and nonlinear collision, at the cost of higher dimensionality than typical finite-difference or finite-volume schemes. In two dimensions, LBM typically evolves nine distinct distribution functions, while in three dimensions it requires at least fifteen, making it a memory-intensive approach. The advection of distribution functions is inherently linear and exact, and implementing complex boundary conditions within the LBM framework is comparably easy. Moreover, the method often employs a uniform computational grid. Although LBM, as a memory-intensive method, offers high potential for data compression, few compression techniques have been developed for it [22, 23, 24], and none of them are independent of the discretization mesh. A major challenge arises from the coupling between spatial and temporal discretization on uniform grids, which prevents straightforward adaptation of local grid refinement and requires special treatment [25]. These characteristics make LBM an excellent candidate for MPS data encoding. Using the MPS formulation, we introduce a computational framework where data compression is achieved through low-rank approximation of non-local correlations, without manipulation of the underlying grid. The substantial memory demand associated with storing distribution functions can thus be effectively mitigated through MPS compression while preserving the explicit weakly compressible formulation of LBM to circumvent the need to solve a Poisson equation. Furthermore, the uniform grid structure intrinsic to LBM is naturally compatible with the MPS representation.

We propose a novel MPS-based method for solving LBM (MPS-LBM). We overcome limitations of previous quantum-inspired CFD approaches by exploiting the flexible formulation of LBM to realize MPS-compressed simulations in complex domains. Decomposing distribution functions as MPS enables high compression rates while maintaining excellent accuracy. Efficient low-rank matrix product operators (MPO) are employed for the streaming step. See Figure 1 for a

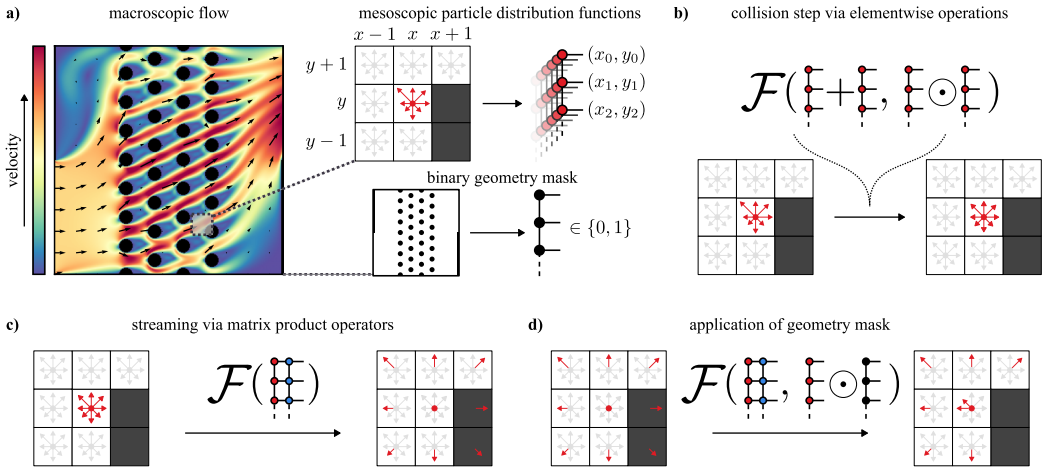

Figure 1: **Overview of the tensor-network lattice Boltzmann method. a)** The macroscopic flow is represented by mesoscopic particle distribution functions. Each particle distribution function is individually decomposed into a tensor network or matrix product state (MPS). The geometry is defined by a binary object mask, which is also decomposed into an MPS of the same shape as the particle distribution functions. **b)** The collision step models local particle interactions through element-wise operations. These are rewritten in order to be efficiently executed in the MPS-manifold via element-wise summation and multiplication. **c)** The streaming step propagates the particle distribution functions in their respective directions. In the MPS-manifold this step is realized through exact low-rank matrix product operators (MPO). **d)** Boundary conditions at complex geometries are enforced by the bounce-back scheme. MPS-LBM encodes boundaries in the MPS manifold by masking the particle distribution functions and applying additional streaming to masked regions.

graphical overview. Additionally, we implement both temporally varying inflow boundary conditions as well as pressure-based outflow conditions. Despite these extensions, MPS-LBM maintains logarithmic scaling in the spatial resolution and quartic scaling in bond dimension, which is characteristic of quantum-inspired CFD methods. The proposed algorithm is validated on the example of the Taylor-Green vortex in three dimensions. Furthermore, a three-dimensional simulation of blood flow in a realistic aneurysm geometry illustrates a typical application of LBM in engineering applications. Finally, an industrially relevant case of a geometrically complex flow is presented. Additional test cases are provided in Appendix A, demonstrating the full capabilities of the proposed algorithm. We show that MPS-LBM for the complex geometry case exhibits exceptionally high compression while maintaining high accuracy, highlighting the utility of MPS decompositions in complex flows around such geometries.

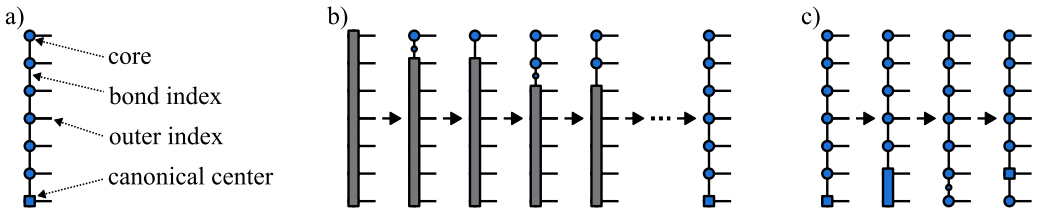

Figure 2: **Diagrammatic explanation of Matrix Product States. a)** Tensor network diagram of an MPS with annotations. **b)** Decomposition of a larger tensor into an MPS as a tensor network diagram. **c)** Shift of the canonical center of an MPS as a tensor network diagram

The remainder of this paper is organized as follows. The fundamentals of the MPS formalism are introduced in Section 2, focusing on specific aspects of MPS-LBM. This section explicitly derives key algorithmic operations: the scale ordering transformation, addition and multiplication, and the closed-form Matrix Product Operator (MPO) for the streaming step. Section 3 establishes the theoretical framework of the Lattice Boltzmann Method, detailing the governing equations as well as their integration with the MPS formalism. It also details the implementation of boundary conditions within the MPS framework. Section 4 defines the numerical benchmarks, specifically the three-dimensional Taylor-Green vortex, blood flow through an aneurysm, and a pin-fin array, and the error metrics used for validation. Section 5, we present a quantitative comparison between the proposed MPS-LBM and a reference classical LBM implementation. Finally, Section 6 discusses the implications of these results for scalable fluid simulation in complex geometries.

2. The Matrix Product State Formalism

2.1. Compression with Matrix Product States

An alternative to grid-based data encodings is given by Matrix Product States (MPS) [6, 7, 8]. An MPS is a specific type of tensor decomposition that allows effective compression and efficient manipulation of large tensors. Specifically, let $\mathbf{T}^{\omega_1,\ldots,\omega_n}$ be an order-n tensor, then there exists a set of tensors $\mathbf{A}_i$ such that

$$\mathbf{T}^{\omega_1,\ldots,\omega_n} = \sum_{\{\boldsymbol{\alpha}\}} (\mathbf{A}_1)^{\omega_1}_{\alpha_1} (\mathbf{A}_2)^{\omega_2}_{\alpha_1,\alpha_2} \ldots (\mathbf{A}_n)^{\omega_n}_{\alpha_{n-1}} . \tag{1}$$

The tensor $(\mathbf{A}_i)$ is called *core* at *site* i. The maximum dimension χ of any *bond* index α_l is called *bond dimension* of the MPS, i.e. $\chi = \max_l |\alpha_l|$. It is common

to depict these kinds of tensor decompositions in a diagrammatic language, where basic geometric shapes represent a tensor and a link between shapes represents a common index that is summed over (see Figure 2a). An MPS can be constructed iteratively from a larger tensor via a sequence of singular value decompositions (SVD). It starts by reshaping $\mathbf{T}$ to a matrix, then applying SVD, contracting the singular values $\mathbf{S}$ into the right resulting matrix $\mathbf{B}$, and repeating the same on $\mathbf{SB}$. In the following, we will denote independent indices as $\alpha, \beta, ...$ and multiple indices combined into a single one as $(\alpha\beta...)$.

$$
\begin{aligned}
\mathbf{T}^{\omega_1,(\omega_2\ldots\omega_n)} &\xrightarrow{\text{SVD}} \sum_{\alpha_1} (\mathbf{A}_1)^{\omega_1}_{\alpha_1} (\mathbf{S}_1)_{\alpha_1} (\mathbf{B}_1)^{(\omega_2\ldots\omega_n)}_{\alpha_1} \\
&\xrightarrow{\text{contract and reshape}} \sum_{\alpha_1} (\mathbf{A}_1)^{\omega_1}_{\alpha_1} (\mathbf{S}_1\mathbf{B}_1)^{(\alpha_1\omega_2),(\omega_3\ldots\omega_n)} \\
&\xrightarrow{\text{SVD}} \sum_{\alpha_1,\alpha_2} (\mathbf{A}_1)^{\omega_1}_{\alpha_1} (\mathbf{A}_2)^{\omega_2}_{\alpha_1,\alpha_2} (\mathbf{S}_2)_{\alpha_2} (\mathbf{B}_2)^{(\omega_3\ldots\omega_n)}_{\alpha_2} \\
&\xrightarrow{\text{contract and reshape}} \sum_{\alpha_1,\alpha_2} (\mathbf{A}_1)^{\omega_1}_{\alpha_1} (\mathbf{A}_2)^{\omega_2}_{\alpha_1,\alpha_2} (\mathbf{S}_2\mathbf{B}_2)^{(\alpha_2\omega_3),(\omega_4\ldots\omega_n)} \\
&\ldots \\
&\to \sum_{\{\boldsymbol{\alpha}\}} (\mathbf{A}_1)^{\omega_1}_{\alpha_1} (\mathbf{A}_2)^{\omega_2}_{\alpha_1,\alpha_2} \ldots (\mathbf{C}_n)^{\omega_n}_{\alpha_{n-1}} ,
\end{aligned}
\tag{2}
$$

where $\mathbf{C}_n = \mathbf{S}_n\mathbf{B}_n$. See also Figure 2b. Truncating singular values at each step allows to reduce the number of elements of the $\mathbf{A}_i$, thus compressing the data encoded in $\mathbf{T}$. SVD ensures that upon truncation to a certain rank the truncation error is minimal in the l^2-norm at each step and scales with the largest omitted singular value. Since all $\mathbf{A}_i$ are left singular matrices they are all left-orthogonal i.e.

$$
\sum_{\alpha,\omega_i} (\mathbf{A}_i)^{\omega_i}_{\alpha,\beta} (\mathbf{A}_i)^{\omega_i}_{\alpha,\gamma} = \mathbb{I}_{\beta,\gamma}. \tag{3}
$$

Because of this the squared norm of $\mathbf{T}$ reduces to

$$
\sum_{\{\omega\}} \mathbf{T}^{\omega_i,\ldots,\omega_n} \mathbf{T}^{\omega_i,\ldots,\omega_n} = \sum_{\alpha_{n-1},\omega_n} (\mathbf{C}_n)^{\omega_n}_{\alpha_{n-1}} (\mathbf{C}_n)^{\omega_n}_{\alpha_{n-1}} , \tag{4}
$$

where the core at site n is called *canonical center*. This canonical center can be shifted to the left using SVD:

$$\begin{aligned} &\ldots (\mathbf{A}_{n-1})^{\omega_{n-1}}_{\alpha_{n-2},\alpha_{n-1}} (\mathbf{C}_n)^{\omega_n}_{\alpha_{n-1}} \\ \xrightarrow{\text{contract and reshape}} &\ldots (\mathbf{A}_{n-1}\mathbf{C}_n)^{(\alpha_{n-2}\omega_{n-1}),(\omega_n)} \\ \xrightarrow{\text{SVD}} &\ldots (\mathbf{A'}_{n-1})^{\omega_{n-1}}_{\alpha_{n-2},\alpha_{n-1}} (\mathbf{S})_{\alpha_{n-1}} (\mathbf{B}_n)^{\omega_n}_{\alpha_{n-1}} \\ \xrightarrow{\text{contract and reshape}} &\ldots (\mathbf{C}_{n-1})^{\omega_{n-1}}_{\alpha_{n-2},\alpha_{n-1}} (\mathbf{B}_n)^{\omega_n}_{\alpha_{n-1}} , \end{aligned} \tag{5}$$

where $\mathbf{C}_{n-1} = \mathbf{A'}_{n-1}\mathbf{S}$. See also Figure 2c. Analogously to Equation 3, $\mathbf{B}_n$ is now right-orthogonal and thus

$$\sum_{\{\omega\}} \mathbf{T}^{\omega_i,\ldots,\omega_n} = \sum_{\alpha_{n-2},\alpha_{n-1},\omega_{n-1}} (\mathbf{C}_{n-1})^{\omega_{n-1}}_{\alpha_{n-2},\alpha_{n-1}} . \tag{6}$$

Similar to Equation 2, this operation can be performed iteratively, and the singular values can again be truncated to achieve additional compression. Generally, compression should only be performed on the canonical center, as from Equation 6 follows that the canonical center encodes all information with respect to the l^2-norm of the entire MPS. A truncated SVD on a site other than the canonical center produces a locally valid compression but degrades the global accuracy of the compression. For more details on different construction approaches as well as further methods to compress an MPS, we refer to [6, 7, 8].

2.2. Algebra on Matrix Products States

In addition to compression, the MPS formalism allows for manipulation of data within the compressed representation without the necessity to contract all cores. Many operations can be efficiently performed in a site-wise manner. In the following we consider two tensors $\mathbf{A}$ and $\mathbf{B}$ with respective MPS representations with cores $(\mathbf{A}_i)^{\omega_i}_{\alpha_{i-1},\alpha_i}$ and $(\mathbf{B}_i)^{\omega_i}_{\beta_{i-1},\beta_i}$. Elementwise addition [6, 7, 8] of $\mathbf{A}$ and $\mathbf{B}$ is achieved by combining the respective cores into block matrices as

$$((\mathbf{A}+\mathbf{B})_i)^{\omega_i}_{\alpha_{i-1},\alpha_i} = \begin{pmatrix} \mathbf{A}_i^{\omega_i} & 0 \\ 0 & \mathbf{B}_i^{\omega_i} \end{pmatrix}_{\alpha'_{i-1},\alpha'_i} , \tag{7}$$

where the new indices α'_i append the entries of β_i to α_i and therefore $|\alpha'_i| = |\alpha_i| + |\beta_i|$. To perform elementwise multiplication, the respective pairs of cores

are contracted with a Kronecker delta tensor $\delta^{i,j,k}$, which has entries 1 if $i = j = k$ and 0 otherwise.

$$((\mathbf{A} \odot \mathbf{B})_i)^{\omega_i'}_{\alpha_{i-1}',\alpha_i'} = \sum_{l,m} \delta^{\omega_i',l,m} \, (\mathbf{A}_i)^{l}_{\alpha_{i-1},\alpha_i} \, (\mathbf{B}_i)^{m}_{\beta_{i-1},\beta_i} \,, \tag{8}$$

where $\alpha_i' = (\alpha_i \beta_i)$ and thus the bond dimensions are multiplied, $|\alpha_i'| = |\alpha_i| \, |\beta_i|$. The equivalent of a linear operator mapping tensors to tensors is a Matrix Product Operator (MPO). Compared to an MPS, an MPO has two outer indices at each site but otherwise the same structure.

$$\mathbf{O}^{\nu_1,\ldots,\nu_n,\omega_1,\ldots,\omega_n} = \sum_{\{\gamma\}} (\mathbf{O}_1)^{\nu_1,\omega_1}_{\gamma_1} \, (\mathbf{O}_2)^{\nu_2,\omega_2}_{\gamma_1,\gamma_2} \ldots (\mathbf{O}_n)^{\nu_n,\omega_n}_{\gamma_{n-1}} \,. \tag{9}$$

An MPO is applied to an MPS site-wise by contracting one of the MPO sets of physical indices with the physical indices of the MPS, resulting again in an MPS.

$$((\mathbf{O}\mathbf{A})_i)^{\nu_i}_{\gamma_{i-1}',\gamma_i'} = \sum_{\omega_i} (\mathbf{O}_i)^{\nu_i,\omega_i}_{\gamma_{i-1},\gamma_i} \, (\mathbf{A}_i)^{\omega_i}_{\alpha_{i-1},\alpha_i} \,, \tag{10}$$

where the bond dimensions multiply aigain $|\gamma_i'| = |\gamma_i| \, |\alpha_i|$.

All of the above operations increase the bond dimensions. To maintain compression during manipulations, it is necessary to control bond dimensions χ. The naive approach to iteratively apply Equation 5 with truncation scales as $\mathcal{O}(\chi^3)$, which is feasible for the mild increase upon a single addition operation or the application of MPO of low bond dimension. However, for the multiplication of two MPS of similar bond dimension χ, this naive approach scales as $\mathcal{O}(\chi^6)$, which demands more efficient methods as discussed below.

2.3. Compressed Elementwise Multiplication

We detail the algorithm employed for element-wise multiplication with simultaneous compression, the single-site version of the algorithm proposed in [26]. We also briefly discuss possible alternatives. Consider the element-wise product of two MPS $\mathbf{B}$ and $\mathbf{C}$ and a candidate MPS $\mathbf{A}$, that has the desired bond dimension. $\mathbf{A}$ is constructed to approximate $\mathbf{B} \odot \mathbf{C}$ by minimizing the squared l^2-norm

$$\begin{aligned}
&\min_{\mathbf{A}} \|\mathbf{A} - (\mathbf{B} \odot \mathbf{C})\|_2^2 \,, \\
\Leftrightarrow &\nabla_{\mathbf{A}} \|\mathbf{A} - (\mathbf{B} \odot \mathbf{C})\|_2^2 = 0, \\
\Leftrightarrow &\nabla_{\mathbf{A}} (\mathbf{A} \cdot \mathbf{A}) - \nabla_{\mathbf{A}} (2\mathbf{A} \cdot (\mathbf{B} \odot \mathbf{C})) + \underbrace{\nabla_{\mathbf{A}} ((\mathbf{B} \odot \mathbf{C}) \cdot (\mathbf{B} \odot \mathbf{C}))}_{=0} = 0.
\end{aligned} \tag{11}$$

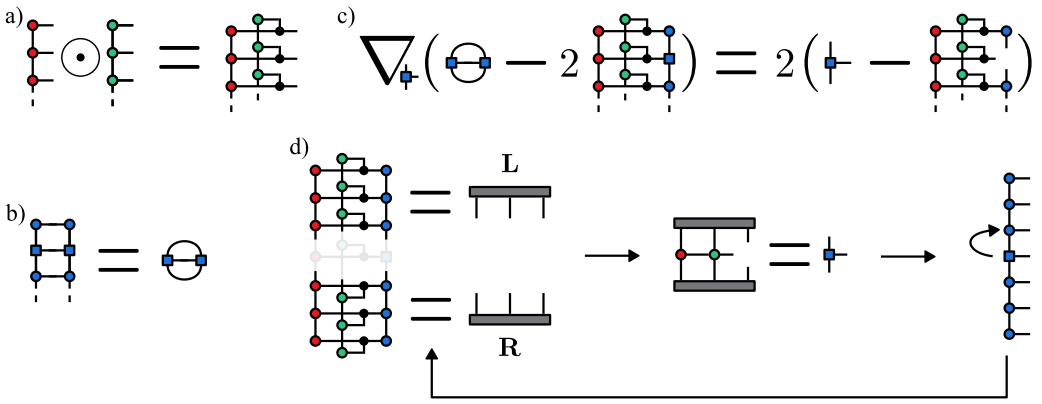

Figure 3: **Diagrammatic explanation of compressed element-wise multiplication.** The product MPS to be multiplied is depicted with red and green sites, the candidate MPS has blue sites, and small black sites are Kronecker delta tensors. **a)** Diagram for exact element-wise multiplication. Physical dimensions of each MPS site are connected via Kronecker delta tensors (black dots). **b)** Orthogonality conditions simplify the inner product of MPS to the inner product of canonical centers. **c)** Local gradient with respect to the candidate canonical center of the term to be minimized. **d)** Iteration loop: First the left L and the right side R of the tensor network are contracted. The sites of the product MPS corresponding to the candidate canonical center are contracted with R and L to form the new canonical center. The new center is inserted in the candidate, and the canonical center is shifted.

Following Equation 8, we can write the element-wise product as the tensor network diagram shown in Figure 3a, where black dots denote Kronecker delta tensors. Now let $\mathbf{A}$ be in canonical form with the center at site c. Then $\mathbf{A} \cdot \mathbf{A}$ reduces to the scalar product of $\mathbf{A}_c$ with itself as in Figure 3b. Instead of the gradient with respect to the whole of $\mathbf{A}$ in Equation 11, we consider the local gradient at site c, and derive the respective tensor network as in Figure 3c. The local minimum is found by setting $\mathbf{A}_c$ equal to the contraction of the right-most term in Figure 3c. Minimization is achieved iteratively as in Figure 3d: Contract the tensor network, replace the canonical center, shift the canonical center, and repeat. The contraction of the tensor network scales as $\mathcal{O}(n\chi^4)$ and the shifting of the canonical center can be performed via a QR-decomposition scaling as $\mathcal{O}(\chi^3)$. In all applications of this method, we take the first product MPS as our candidate $\mathbf{A} = \mathbf{B}$ and sweep through the sites $c = 1 \rightarrow c = n \rightarrow c = 1$ two times.

Alternatives would be the zip-up algorithm proposed by Stoudenmire et al. [27], which also scales in the bond dimension as $\mathcal{O}(\chi^4)$, or the algorithm proposed by Michaelidis et al. [28] with an improved asymptotic scaling of $\mathcal{O}(\chi^3)$. We chose the iterative method over the alternatives mainly because of the intention to

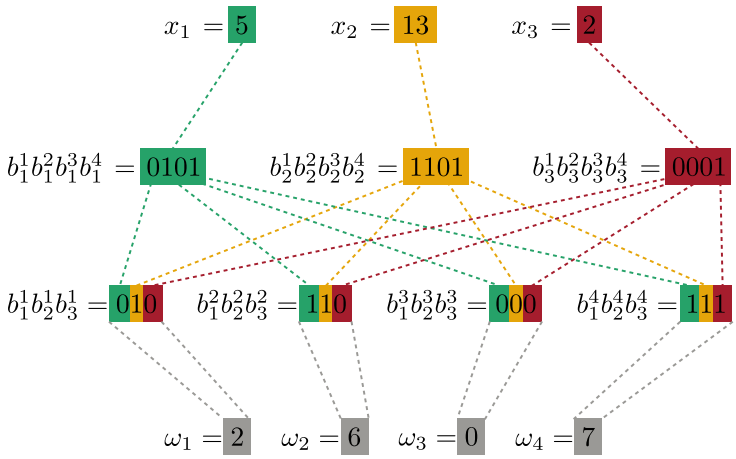

Figure 4: **Example of scale ordering of indices.** Exemplary reordering of indices with $D = 3$ and $n = 4$. The integer indices x_i are represented in binary. The binary digits are regrouped in terms of their value. The resulting binary numbers are reinterpreted in terms of integers.

perform MPS-LBM simulations on GPU. Using QR-decomposition, the iterative method can be implemented without SVD, which has been shown to be preferable for MPS computations on GPU [29]. Moreover, both alternative algorithms rely on intermediate results of preferably flexible bond dimensions, which we found to significantly impede performance gains through JAX [30] just-in-time compilation, from which we expect further computational performance gains.

2.4. Vector Fields in Scale-Ordered Matrix Product State Representation

For numerical analysis, the scalar fields encountered in continuum mechanics are commonly discretized on a uniform grid within a computational domain. For example, consider the scalar field $f(\boldsymbol{x})$ discretized on a uniform grid within a computational domain of spatial dimension D. Coordinates are given as integer-valued indices x_d, such that $\sum_d x_d \boldsymbol{e}_d$ corresponds to the respective continuous coordinate, where $\boldsymbol{e}_d$ are the basis vectors of the lattice. For simplicity let the number of grid points in each spatial dimension be constrained to 2^n, implying each discretized scalar field to be a tensor $\mathbf{f}^{x_1,\dots,x_D}$ of shape $(2^n)^{\times D}$. Commonly $D = 3$, thus naively decomposing $\mathbf{f}$ into an MPS would result in only three cores, limiting the possible compression from truncation. However, n needs to be large to resolve small details in the field. Following the scale-ordering scheme introduced in [12], consider a binary representation of each spatial index as $x_i = b_i^1 \dots b_i^n$.

Reordering the binary indices as

$$\begin{aligned} & x_1, \ldots, x_D = (b_1^1 \ldots b_1^n), \ldots, (b_D^1 \ldots b_D^n) \\ \xrightarrow{\text{scale-ordering}} \quad & (b_1^1 \ldots b_D^1), \ldots, (b_1^n \ldots b_D^n) = \omega_1, \ldots, \omega_n, \end{aligned} \tag{12}$$

allows to reshape $\mathbf{f}$ into an order-n tensor

$$\mathbf{f}^{x_1,\ldots,x_D} \xrightarrow{\text{scale-ordering}} \mathbf{f}^{\omega_1,\ldots,\omega_n}, \tag{13}$$

which is well suited to MPS-decomposition as in Equation 1. See Figure 4 for an example with $D = 3$ and $n = 4$.

2.5. Matrix Product Operators for Shifting

For the streaming step in LBM, we formulate shift MPO, i.e. linear operators $\mathbf{S_d}$ that shift all entries of a discretized scalar field by one grid node in a given direction d.

$$\mathbf{f}^{x_1,\ldots,x_d,\ldots,x_D} \xrightarrow{\mathbf{S_d}} \mathbf{f}^{x_1,\ldots,x_{d+1},\ldots,x_D}. \tag{14}$$

We construct the shift MPO for cyclic and non-cyclic shifts, and show that the MPO are of low bond dimension. The shift MPO is based on earlier work [12, 18], and most of this derivation closely follows that of Kazeev et al. [31].

Consider for $D = 2^n$ cyclic $\mathbf{S}^{(n)}$ and non-cyclic $\hat{\mathbf{S}}^{(n)}$ shift matrices. For $n = 2$, these matrices are defined as

$$\mathbf{S}^{(2)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad \hat{\mathbf{S}}^{(2)} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{15}$$

Using the 2×2 matrices

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad J = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad J' = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{16}$$

the shift matrices can be constructed as

$$\begin{aligned} \mathbf{S}^{(2)} &= \begin{pmatrix} J & J' \\ J' & J \end{pmatrix} = I \otimes J + P \otimes J', \\ \hat{\mathbf{S}}^{(2)} &= \begin{pmatrix} J & J' \\ 0 & J \end{pmatrix} = I \otimes J + J \otimes J'. \end{aligned} \tag{17}$$

For $n = 3$, the shift matrices can be expressed in terms of $\mathbf{S}^{(2)}$ and $\hat{\mathbf{S}}^{(2)}$ as

$$\begin{aligned}\mathbf{S}^{(3)} &= \begin{pmatrix} \hat{\mathbf{S}}^{(2)} & J'^{\otimes 2} \\ J'^{\otimes 2} & \hat{\mathbf{S}}^{(2)} \end{pmatrix} = I \otimes \hat{\mathbf{S}}^{(2)} + P \otimes J'^{\otimes 2}, \\ \hat{\mathbf{S}}^{(3)} &= \begin{pmatrix} \hat{\mathbf{S}}^{(2)} & J'^{\otimes 2} \\ 0 & \hat{\mathbf{S}}^{(2)} \end{pmatrix} = I \otimes \hat{\mathbf{S}}^{(2)} + J \otimes J'^{\otimes 2},\end{aligned} \tag{18}$$

leading to the recursive definition

$$\begin{aligned}\mathbf{S}^{(n)} &= \begin{pmatrix} \hat{\mathbf{S}}^{(n-1)} & J'^{\otimes (n-1)} \\ J'^{\otimes (n-1)} & \hat{\mathbf{S}}^{(n-1)} \end{pmatrix} = I \otimes \hat{\mathbf{S}}^{(n-1)} + P \otimes J'^{\otimes (n-1)}, \\ \hat{\mathbf{S}}^{(n)} &= \begin{pmatrix} \hat{\mathbf{S}}^{(n-1)} & J'^{\otimes (n-1)} \\ 0 & \hat{\mathbf{S}}^{(n-1)} \end{pmatrix} = I \otimes \hat{\mathbf{S}}^{(n-1)} + J \otimes J'^{\otimes (n-1)}.\end{aligned} \tag{19}$$

We introduce the rank core product as defined in [31]. Consider 4-dimensional tensors $\mathbf{A}^{ab}_{\alpha\beta}$ and $\mathbf{B}^{ab}_{\alpha\beta}$, and explicitly expand the upper indices as

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \dots \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}, \quad \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} & \dots \\ \mathbf{B}_{21} & \mathbf{B}_{22} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}, \tag{20}$$

where each entry is a 2-dimensional tensor. The rank core product "$\bowtie$" of these two tensors is

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \bowtie \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} \otimes \mathbf{B}_{11} + \mathbf{A}_{12} \otimes \mathbf{B}_{21} & \mathbf{A}_{11} \otimes \mathbf{B}_{12} + \mathbf{A}_{12} \otimes \mathbf{B}_{22} \\ \mathbf{A}_{21} \otimes \mathbf{B}_{11} + \mathbf{A}_{22} \otimes \mathbf{B}_{21} & \mathbf{A}_{21} \otimes \mathbf{B}_{12} + \mathbf{A}_{22} \otimes \mathbf{B}_{22} \end{bmatrix}. \tag{21}$$

For connecting this to MPO, consider a 3-dimensional tensor $\mathbf{T} \in \mathbb{R}^3$ and a linear operator

$$\begin{aligned}\mathbf{O} &: \mathbb{R}^3 \to \mathbb{R}^3 \\ \mathbf{T}^{abc} &\mapsto \sum_{a,b,c} \mathbf{O}^{def,abc}\, \mathbf{T}^{abc}.\end{aligned} \tag{22}$$

We define the MPO decomposition of $\mathbf{O}$ as

$$\mathbf{O}^{abc,def} = \sum_{\alpha,\beta} \mathbf{A}^{ad}_{\alpha} \mathbf{B}^{be}_{\alpha\beta} \mathbf{C}^{cf}_{\beta}, \tag{23}$$

which can be written in terms of the rank core product as

$$\mathbf{O} = \mathbf{A} \bowtie \mathbf{B} \bowtie \mathbf{C}. \tag{24}$$

Similarly, we decompose $\mathbf{S}$ and $\hat{\mathbf{S}}$ with the rank core product and rewrite Equation 19 as

$$\begin{aligned}\mathbf{S}^{(n)} &= \begin{bmatrix} I & P \end{bmatrix} \bowtie \begin{bmatrix} \hat{\mathbf{S}}^{(n-1)} \\ J'^{\otimes(n-1)} \end{bmatrix}, \\ \hat{\mathbf{S}}^{(n)} &= \begin{bmatrix} I & J \end{bmatrix} \bowtie \begin{bmatrix} \hat{\mathbf{S}}^{(n-1)} \\ J'^{\otimes(n-1)} \end{bmatrix},\end{aligned} \tag{25}$$

which indicates that only the first core differs for cyclic and non-cyclic shift MPO. $\hat{\mathbf{S}}$ can be further decomposed as

$$\begin{aligned}\hat{\mathbf{S}}^{(n)} &= \begin{bmatrix} I & J \end{bmatrix} \bowtie \begin{bmatrix} I & J \\ 0 & J' \end{bmatrix} \bowtie \begin{bmatrix} \hat{\mathbf{S}}^{(n-2)} \\ J'^{\otimes(n-2)} \end{bmatrix} \\ &= \begin{bmatrix} I & J \end{bmatrix} \bowtie \underbrace{\begin{bmatrix} I & J \\ 0 & J' \end{bmatrix} \bowtie \cdots \bowtie \begin{bmatrix} I & J \\ 0 & J' \end{bmatrix}}_{(n-2)-\text{times}} \bowtie \begin{bmatrix} \hat{\mathbf{S}}^{(1)} \\ J' \end{bmatrix}.\end{aligned} \tag{26}$$

Using $\hat{\mathbf{S}}^{(1)} = J$ we can define

$$\mathbf{A} = \begin{bmatrix} I & P \end{bmatrix}, \quad \hat{\mathbf{A}} = \begin{bmatrix} I & J \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} I & J \\ 0 & J' \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} J \\ J' \end{bmatrix}, \tag{27}$$

such that

$$\begin{aligned}\mathbf{S}^{(n)} &= \mathbf{A} \bowtie \mathbf{B} \bowtie \cdots \bowtie \mathbf{B} \bowtie \mathbf{C}, \\ \hat{\mathbf{S}}^{(n)} &= \hat{\mathbf{A}} \bowtie \mathbf{B} \bowtie \cdots \bowtie \mathbf{B} \bowtie \mathbf{C}.\end{aligned} \tag{28}$$

It is straightforward to show that a shift in the opposite direction is achieved by simply interchanging J with J' and vice versa in all cores.

The above derivation only produces shift matrices on vectors. However, for MPS-LBM, we need to shift in all physical dimensions separately. Note that in Equation 12, the indices are split into binary digits such that the i-th bit of each core corresponds to the i-th dimension. Thus, the cores of the shift MPO are to act on the i-th index only to result in a shift in the i-th dimension. This is achieved by padding the individual cores with identities. Say we want a shift in the first of three dimensions, then the first core of the respective MPO is

$$A \bowtie \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \bowtie \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}, \tag{29}$$

and analogously $\mathbf{B}$ and $\mathbf{C}$ are padded.

3. Matrix Product States in the Lattice Boltzmann Method

3.1. Lattice Boltzmann Method

The lattice Boltzmann method (LBM) is a mesoscopic model of fluid dynamics, approximating the evolution of macroscopic flow governed by the Navier-Stokes equations through discretized particle distribution functions f_i. The evolution of single components of particle distribution functions is governed by the lattice Boltzmann equation [32, 33]

$$f_i(\boldsymbol{x} + \boldsymbol{c}_i\Delta t, t + \Delta t) = f_i(\boldsymbol{x}, t) + \Omega_i(\boldsymbol{x}, t), \tag{30}$$

with Ω_i representing the collision operator. This process consists of two steps: (i) *collision*, a local relaxation modeling local particle collision by the redistribution of particles among distribution functions f_i (given by the RHS of Equation 30), and (ii) *streaming*, which propagates particle distribution functions along the lattice (shifting of space and time coordinates on the LHS of Equation 30). For simplicity, we employ the BGK collision operator [21] which involves a linear relaxation with rate τ towards an equilibrium f_i^{eq}

$$\Omega_i(\boldsymbol{x}, t) = \frac{\Delta t}{\tau}\left(f_i^{eq}(\boldsymbol{x}, t) - f_i(\boldsymbol{x}, t)\right), \tag{31}$$

with the *equilibrium distribution* f^{eq} defined as

$$f_i^{eq}(\boldsymbol{x}, t) = w_i\rho\left(1 + \frac{\boldsymbol{u}\cdot\boldsymbol{c}_i}{c_s^2} + \frac{(\boldsymbol{u}\cdot\boldsymbol{c}_i)^2}{2c_s^4} - \frac{\boldsymbol{u}\cdot\boldsymbol{u}}{2c_s^2}\right). \tag{32}$$

The *relaxation time* τ is determined by the kinematic viscosity ν via $\tau = \frac{\Delta t}{2} + \frac{\nu}{c_s^2}$. Macroscopic quantities such as density ρ and momentum $\rho\boldsymbol{u}$ are recovered from the hydrodynamic moments [34, 35] of these particle distribution functions

$$\rho = \sum_{i=1}^{Q} f_i(\boldsymbol{x}, t), \qquad \rho\boldsymbol{u} = \sum_{i=1}^{Q} \boldsymbol{c}_i f_i(\boldsymbol{x}, t), \tag{33}$$

where $Q = |\{\boldsymbol{c}_i\}|$ denotes the number of discrete microscopic velocities $\boldsymbol{c}_i$.

3.2. MPS Lattice Boltzmann Method

The lattice Boltzmann method based on the MPS framework (MPS-LBM) leverages the mostly local formulation of LBM. Distribution functions f_i are spatially discretized on uniform Cartesian meshes. An MPS-decomposition of spatial

dimensions enables compression, reducing the number of variables parameterizing the state with controllable accuracy.

All considered scalar fields, i.e, the particle distribution functions f_i and macroscopic variables $\boldsymbol{u}$ and ρ, are discretized on a uniform grid within a computational domain of spatial dimension D. Following Equation 12 they are brought into scale-ordered form to enable effective MPS decomposition. Local collision interactions consist mostly of element-wise addition and multiplication, while the streaming step admits an exact formulation as a low-rank matrix product operator. As shown in Section 2, these operations can be performed efficiently without leaving the compressed MPS manifold. The only operation that cannot be directly handled by the MPS formalism is the computation of velocity u via the first moment of the particle distribution functions in Equation 33. We approximate the normalization factor $1/\rho$ to circumvent the lack of a closed-form algorithm for elementwise inversion of an MPS. Sanavio and Succi [36] proposed a Taylor expansion of this term around the constant value 1 for a Carleman linearization algorithm of the LBM, which is a suitable approximation for weakly compressible single-phase flows. We build upon this idea by introducing a second-order Taylor expansion of $1/\rho$ around the global mean density ρ_0 at each time step t

$$\frac{1}{\rho} \approx \frac{1}{\rho_0} - \frac{\delta\rho}{\rho_0^2} + \frac{(\delta\rho)^2}{\rho_0^3} + \mathcal{O}\left((\delta\rho)^3\right), \tag{34}$$

where $\delta\rho = \rho - \rho_0$ denotes the density fluctuation. The mean density ρ_0 is computed by contracting the ρ-MPS with a rank-1 unit MPS and normalizing the result. For weakly compressible flows, the relative density fluctuation scales with the Mach number Ma as $\frac{\delta\rho}{\rho_0} \sim \mathcal{O}(\mathrm{Ma}^2)$ in the limit $\mathrm{Ma} \to 0$ [37]. Consequently, the second-order approximation of $1/\rho$ introduces an error that scales as $\mathcal{O}(\mathrm{Ma}^6)$. The introduced error is several magnitudes smaller than the inherent compressibility error $\mathcal{O}(\mathrm{Ma}^2)$ of the LBM [38]. This approach is validated by numerical experiments presented in Appendix B. While this approximation is robust within the weakly compressible regime, its accuracy deteriorates outside of it. For simulations involving large density variations, e.g. multiphase flows with steep density gradients or fully compressible flows, the condition $\delta\rho/\rho_0 \ll 1$ no longer applies, rendering the low-order Taylor expansion inaccurate.

The runtime of MPS-LBM is dominated by elementwise addition, multiplication, and the application of the low-rank shift MPO. Addition of two MPS is performed as in Equation 7, which yields bond dimensions on the order of 2χ. Reducing the bond dimension via an iteration of singular value decompositions

(SVD) introduces a complexity scaling of $\mathcal{O}(n\chi^3)$ for this operation, where n is the length of the MPS determined by the grid resolution. As shown in Section 2.5, the bond dimension of shift MPO is $\chi \leq 3$, and hence the application of shift MPO to MPS followed by an iteration of SVD also results in a scaling of $\mathcal{O}(n\chi^3)$. Following our discussion in Section 2.3, elementwise multiplication of two MPS scales as $\mathcal{O}(n\chi^4)$. In total, we find that our algorithm is dominated by elementwise multiplication and thus exhibits an overall cost scaling of $\mathcal{O}(n\chi^4)$ matching the asymptotic runtime of previous quantum-inspired approaches to CFD. Note that any of the aforementioned primitives can be straightforwardly substituted with further improvements in terms of MPS operations and their implementation in high-performance libraries. We provide the results of numerical runtime experiments on current GPU hardware in Appendix C.

3.2.1. Masking of Objects and Boundaries

In lattice Boltzmann simulations, the computation of immersed objects or non-periodic boundary conditions requires localized computations that only affect cells on the edge of the domain or cells at the boundary of an object. In conventional implementations, these nodes are readily accessible via explicit indexing. However, the inherently non-local and compressed structure of MPS representations prevents direct access to individual boundary cells. In MPS-LBM, a combination of non-cyclic shifts and low-rank binary MPS marking boundary cells is employed to enable boundary-specific computations.

Non-Periodic Boundaries. Non-periodic boundaries are treated by replacing the cyclic shift matrices $\mathbf{S}$ used in the streaming step with non-cyclic ones $\hat{\mathbf{S}}$ at the corresponding boundaries. Upon shifting, this introduces zero rows on the set of boundary cells $\mathcal{B}$ corresponding to particle distributions originating from outside of the computational domain. Inside the domain, $\hat{\mathbf{S}}$ continues to perform shifts consistent with the streaming operation. We employ a binary mask $m^{\boldsymbol{x}}_{\mathcal{B}}$, that is 1 for all $x \in \mathcal{B}$, and 0 elsewhere. If $x_i = 0$ ($x_i = N$) is a periodic boundary and $a_{i,d} = 1$ ($a_{i,d} = -1$) for some d, then modified streaming

$$f_i^{\boldsymbol{x}} \mapsto \left(\bigotimes_{d=1}^{D} \mathbf{S}(a_{i,d}) \right) f_i^{\boldsymbol{x}} \tag{35}$$

is applied. Otherwise we apply

$$f_i^{\boldsymbol{x}} \mapsto m^{\boldsymbol{x}}_{\mathcal{B}}\, b(f_{\hat{i}}^{\boldsymbol{x}}, \rho_b, \boldsymbol{u}_b) + \left(\bigotimes_{d=1}^{D} \hat{\mathbf{S}}(a_{i,d}) \right) f_i^{\boldsymbol{x}}. \tag{36}$$

Here, $b(f_{\hat{i}}^{\boldsymbol{x}}, \rho_b, \boldsymbol{u}_b)$ denotes a general boundary function that computes the appropriate values based on the incoming distribution $f_{\hat{i}}^{\boldsymbol{x}}$, boundary density ρ_b, and velocity $\boldsymbol{u}_b$. For a no-slip boundary [39, 40] the function $b_{\text{no-slip}}$ is given by the bounce-back rule as

$$b_{\text{no-slip}}(f_{\hat{i}}^{\boldsymbol{x}}, \rho_b, \boldsymbol{u}_b) = f_{\hat{i}}^{\boldsymbol{x}}. \tag{37}$$

Dirichlet boundaries with prescribed velocity $\boldsymbol{u}_b$ are given by

$$b_{\text{velocity}}(f_{\hat{i}}^{\boldsymbol{x}}, \rho_b, \boldsymbol{u}_b) = f_{\hat{i}}^{\boldsymbol{x}} - \frac{2w_i\rho_b}{c_s^2}\boldsymbol{c}_i \cdot \boldsymbol{u}_b, \tag{38}$$

where ρ_b is the density at the boundary. We set $\rho_b = \rho_0$, assuming weak compressibility in the computational domain. Outlet boundaries [41] are computed by describing a wall density ρ_b and using the velocity $\boldsymbol{u}(\boldsymbol{x} \in \mathcal{B}, t) = \boldsymbol{u}_b$

$$b_{\text{pressure}}(f_{\hat{i}}^{\boldsymbol{x}}, \rho_b, \boldsymbol{u}_b) = -f_{\hat{i}}^{\boldsymbol{x}} + 2w_i\rho_b\left(1 + \frac{(\boldsymbol{c}_i \cdot \boldsymbol{u}_b)^2}{2c_s^4} - \frac{\boldsymbol{u}_b \cdot \boldsymbol{u}_b}{2c_s^2}\right). \tag{39}$$

Immersed Objects. For immersed objects, we apply the same no-slip condition as defined in Equation 37. However, unlike the approach used for non-cyclic boundaries in Equation 36, we do not modify the streaming step directly. Instead, we allow the streaming to proceed as if no obstacles were present, and subsequently reverse the streaming for the solid internal boundary cells. This reversal is conditioned by a binary MPS mask $m_{\mathcal{O}}^{\boldsymbol{x}}$, which takes the value 1 at all cells $x \in \mathcal{O}$ inside the object and 0 elsewhere.

The implementation is

$$f_i^{\boldsymbol{x}} \mapsto (1 - m_{\mathcal{O}}^{\boldsymbol{x}})\, f_i^{\boldsymbol{x}} + \left(\bigotimes_{d=1}^{D} \mathbf{S}(a_{i,d})\right) m_{\mathcal{O}}^{\boldsymbol{x}}\, f_{\hat{i}}^{\boldsymbol{x}}. \tag{40}$$

While the multiplication with $m_{\mathcal{O}}^{\boldsymbol{x}}$ ensures bounce-back, $1 - m_{\mathcal{O}}^{\boldsymbol{x}}$ deletes the remaining non-zero entries inside the solid object.

4. Experimental Setup

4.1. Test Settings

In LBM, a unit conversion from *physical* to *lattice* units is applied. For all experiments, this conversion is given by the velocity conversion factor $\Delta u = \frac{u_{\text{physical}}}{u_{\text{lattice}}} = \frac{u_0}{\text{Ma}\ c_s}$. Length is discretized using $\Delta x = \frac{L}{N}$ and the time step is determined by $\Delta t = \frac{\Delta x}{\Delta u}$. Here, u_0 is the characteristic velocity of the flow problem, L

is the side length of the domain, and N is the number of lattice sites per side. Two-dimensional and three-dimensional simulations use the D2Q9 and D3Q15 lattice schemes, respectively, which define the lattice speed of sound as $c_s = 1/\sqrt{3}$.

The three-dimensional Taylor-Green vortex (TGV) is defined on a cube of edge length $L = 2\pi$ and a characteristic velocity $u_0 = 1$ with initial conditions

$$\begin{aligned} u(x,y,z) &= -\sin(x)\cos(y)\cos(z)\,, \\ v(x,y,z) &= \cos(x)\sin(y)\cos(z)\,, \\ w(x,y,z) &= 0, \\ \rho(x,y,z) &= 1 + \frac{1}{16\Delta u^2}\left(\cos(2x) + \cos(2y)\right)\cos(2z)\,. \end{aligned} \tag{41}$$

In accordance with similar experiments in [42, 12], we define the Reynolds number as $\mathrm{Re} = \frac{u_0 L}{2\pi\nu}$. All 3D TGV simulations were conducted with $\mathrm{Ma} = 0.1/c_s$ at a resolution of $N^3 = 256^3$.

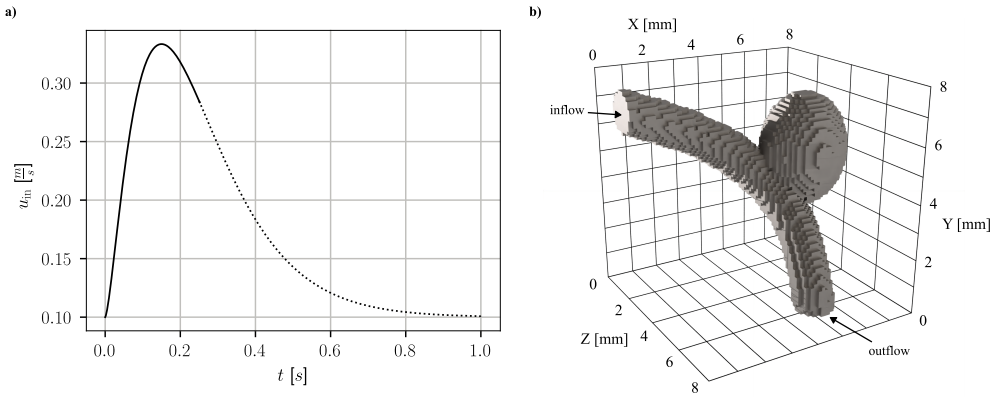

Figure 5: **Setup of the aneurysm simulation**. **a)** Time-dependent magnitude of the flow velocity at the inflow. The simulations were performed until $t = 0.25s$ depicted by the solid line. For reference, the dotted line shows the remaining part of the $1s$ heartbeat cycle. **b)** Full geometry as used in the aneurysm test case. The time-dependent inflow is at $z = 0$, and the outflow boundary is at $y = 0$.

To ensure realistic flow conditions within the aneurysm geometry, we base our simulation parameters on those reported by Horvat et al. [43], with appropriate adjustments to account for the specifics of our computational setup. The velocities averaged over the inflow surface, ranging between 0.1 m/s and 0.34 m/s, are approximated by $v_{\text{in}}(t) = 0.1 + 18t^{1.5}e^{-10t}$ m/s (see Figure 5a). The domain is a cube of edge length 8 mm discretized at a resolution of $N^3 = 64^3$. The

idealized aneurysm geometry is shown in Figure 5b. The aneurysm has a size of approximately $l_{\text{an}} \approx 4\text{mm}$. The relaxation time is given by $\tau = \frac{\text{Ma}N\nu}{Lu_0c_s}$, where $u_0 \approx 0.78\ \text{m/s}$ is the maximum velocity observed at the center of the blood vessel during simulation. The viscosity was adjusted to $\nu = 1.12 \times 10^{-5}\ \text{m}^2/\text{s}$ to ensure numerical stability of the computational setup. This results in a Reynolds number of $\text{Re} = \frac{l_{\text{an}}u_0}{\nu} \approx 280$. As an initial condition for the particle distribution functions, a stationary flow was precomputed using a constant inflow velocity of $0.1\ \text{m/s}$ over a duration of $0.15\ \text{s}$.

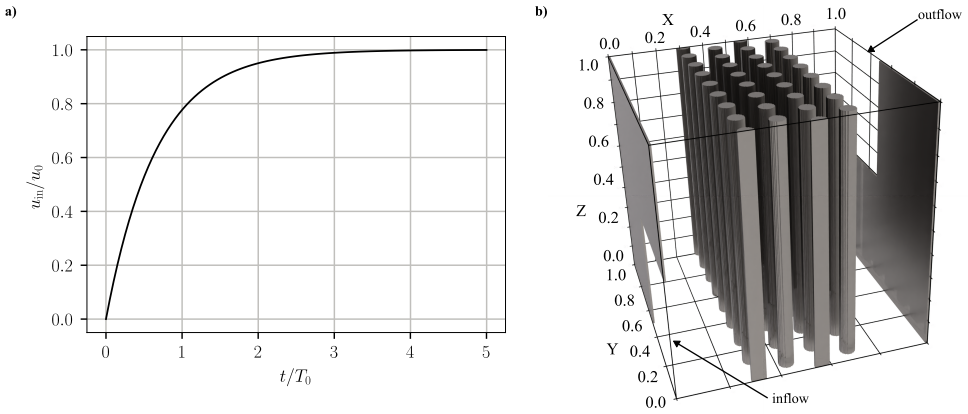

Figure 6: **Setup of the pin-fin heat sink simulation**. **a)** Time-dependent magnitude of the flow velocity at the inflow. **b)** Full geometry as used in the pin-fin test case. The time-dependent inflow is at $x = 0$, and the outflow boundary is at $x = L$.

In developing the parameterization of the pin-fin heat sink, we follow [44]. To ensure applicability across a range of geometric length scales, the heat-sink configuration is defined on a normalized cubic domain of edge length L (see Figure 6b). The simulation is initiated with a vanishing velocity field, and the inflow velocity is ramped up as $u_{\text{in}}(t) = u_0\,(1 - \exp(-1.5t))$ (see Figure 6a). The Reynolds number is $\text{Re} = \frac{3u_0L}{16\nu} = 400$ with respect to the distance between the layers of pins $l_{\text{layer}} = L/8$ and the maximum velocities occurring towards the end of the simulation of around $u_{\max} \approx \frac{3}{2}u_0$. The simulations were run up to $t/T_0 = 5$ with $T_0 = L/u_0$.

4.2. Quantitative Metrics

We evaluate the MPS-LBM accuracy using the relative error of the macroscopic velocity field $\boldsymbol{u}$ in the l^2-norm given by

$$\epsilon_{\mathrm{l}^2} = \frac{\sqrt{\sum_{d,\boldsymbol{x}} \left(u_{d,\boldsymbol{x}} - \hat{u}_{d,\boldsymbol{x}}\right)^2}}{\sqrt{\sum_{d,\boldsymbol{x}} \hat{u}_{d,\boldsymbol{x}}^2}}, \tag{42}$$

where $u_{d,\boldsymbol{x}}$ is the result of MPS-LBM, index d is the spatial dimension of the velocity field at position $\boldsymbol{x}$ and $\hat{\boldsymbol{u}}_{\boldsymbol{d},\boldsymbol{x}}$ is the reference result.

The total kinetic energy E_{kin} for TGV is given by

$$E_{\text{kin}} = \frac{\rho V_c}{2} \sum_{d} u_{d,\boldsymbol{x}}^2, \tag{43}$$

where V_c is the volume per lattice site. Density is $\rho = 1/L^3$ such that the total mass is normalized to 1. The total energy dissipation $-\frac{d}{dt} E_{\text{kin}}$ is numerically evaluated using second-order finite differences.

Pressures p are normalized in terms of the prescribed pressure at the outflow p_0 and averaged over the unmasked parts of the region of interest $\mathcal{D}$

$$\Delta p / p_0 = \frac{1}{p_0 |\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \left(p_{\boldsymbol{x}} - p_0\right). \tag{44}$$

In the aneurysm case, we average over the region around the aneurysm $0.53L < x < 0.97L, 0.39L < y < 0.93L, 0.33L < z < 0.69L$. In the pin-fin configuration, we are interested in the pressure drop from inflow to outflow, so we average over the inflow region $0 < x < 2L/256, 0 < y < L/2, 0 < z < L/2$.

The compression ratio CR is

$$\text{CR} = \frac{\text{NVPS}_{\text{ref}}}{\text{NVPS}}, \tag{45}$$

where the number of variables parametrising the solution (NVPS) of a component of the standard LBM on a $(2^n)^D$ grid is $\text{NVPS}_{\text{LBM}} = (2^n)^D$. For MPS-LBM, it is given as

$$\text{NVPS}_{\text{MPS}} = \sum_{i=1}^{n} \min\left(\left(2^D\right)^{i-1}, \left(2^D\right)^{n-i+1}, \chi\right) \times 2^D \times \min\left(\left(2^D\right)^{i}, \left(2^D\right)^{n-i}, \chi\right). \tag{46}$$

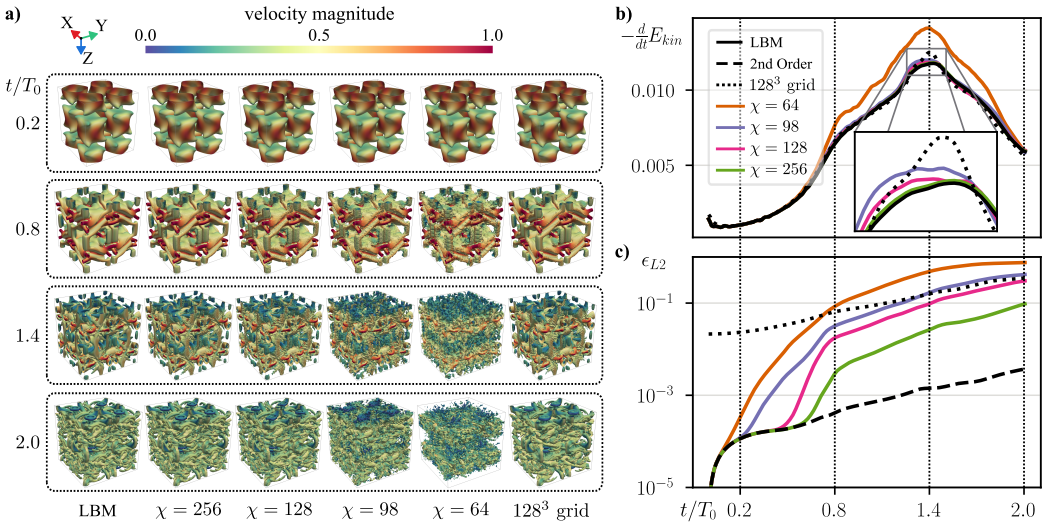

Figure 7: **Results of the 3D Taylor-Green vortex**. a) Vortical structures, identified by the Q-criterion isosurfaces, are shown at four distinct times $t/T_0 = 0.2, 0.8, 1.4, 2$ for LBM on 256^3 mesh, MPS-LBM with $\chi = 256, 128, 98, 64$, and a coarse LBM. **b)** Total kinetic energy dissipation $-\frac{d}{dt}E_{\text{kin}}$, colored solid lines depict MPS-LBM results, solid black line fine LBM, dotted black line coarse LBM. **c)** The relative l^2 error ϵ_{l^2} with respect to fine LBM. Solid colored lines represent MPS-LBM, dotted black corresponds to coarse LBM. For comparison, the values of fine LBM with second-order approximation of $1/\rho$ are given as the dashed black line.

5. Results

5.1. Three Dimensional Taylor-Green Vortex

We simulate the three-dimensional Taylor-Green vortex (TGV) at a Reynolds number of $\text{Re} = 800$ on a grid of size 256^3. To investigate the impact of compression on simulation accuracy, we conduct experiments using MPS-LBM with different bond dimensions $\chi = \{256, 128, 98, 64\}$. Reference results are obtained using three configurations: (i) a fine LBM on a 256^3 mesh, (ii) fine LBM with a second-order approximation of $1/\rho$ on 256^3 mesh, and (iii) coarse LBM on a 128^3 mesh.

Figure 7a depicts all simulation results at four different times $t/T_0 = 0.2, 0.8, 1.4, 2.0$, where $T_0 = L/u_0$. Here, u_0 denotes the maximum velocity magnitude in the initial conditions, and L is the edge length of the domain. All simulations capture the large-scale structures at time $t/T_0 = 0.2$ well. The solution trajectory deviates from the reference for $\chi = 64$ at $t/T_0 = 0.8$ and for $\chi = 98$ at $t/T_0 = 1.4$, as compression prohibits the formation of smaller-scale structures that emerge

later in the flow. Simulations using $\chi = 128$ and $\chi = 256$ reproduce all relevant details throughout the simulation, with $\chi = 128$ showing only minor deviations from fine LBM. In comparison, the coarse LBM falls between the MPS-LBM results with $\chi = 256$ and with $\chi = 128$.

In Figure 7b, the energy dissipation $-\frac{d}{dt}E_{\text{kin}}$ is shown. All simulations qualitatively follow the fine LBM and produce physically consistent behavior. The simulation with $\chi = 64$ deviates significantly from the baseline, with the largest errors near the dissipation peak. All MPS-LBM simulations show slightly elevated dissipation before the peak at $t/T_0 \approx 1.3$, while the under-resolved 128^3 LBM exhibits the highest dissipation, marked by a sharp peak at $t/T_0 = 1.4$.

Figure 7c shows the relative error in the L^2-norm ϵ_{l^2} of the macroscopic velocity field for all simulation cases compared to the reference LBM result. As an additional reference, the error of fine LBM with second-order approximation of $1/\rho$ is included. This comparison helps to distinguish the influence of MPS compression from that of the $1/\rho$ approximation. At $t/T_0 = 0.2$, the curve for $\chi = 64$ already separates from the second-order approximation, while higher bond dimensions introduce no noticeable additional inaccuracies. This changes at $t/T_0 = 0.8$, where simulations with lower bond dimensions exhibit higher errors. From that point onward, all simulations show a gradual increase at error, with approximately half an order of magnitude separating $\chi = 64$, $\chi = 98$, and $\chi = 256$. The coarse LBM starts with a substantially higher initial error. However, it matches $\chi = 64$ at $t/T_0 = 0.8$ and $\chi = 98$ at $t/T_0 = 1.4$. The final error at $t/T_0 = 2$ is close to that of MPS-LBM with a bond dimension of $\chi = 128$.

For bond dimensions $\chi = 256$ and $\chi = 128$, the compression ratios are $\text{CR}_{256} \approx 13$ and $\text{CR}_{128} \approx 42$, respectively. A 128^3 grid yields CR ≈ 8, while $\chi = 98$ achieves $\text{CR}_{98} \approx 64$, equivalent to a 64^3 grid. Lower grid resolutions are infeasible due to stability issues arising from a small relaxation time τ. These results demonstrate that MPS-LBM outperforms simple grid-size reduction in both accuracy and compression efficiency.

5.2. Aneurysm

To demonstrate the versatility of MPS-LBM, we simulate blood flow through an aneurysm using time-dependent boundary conditions and a complex geometry encoded via a precomputed MPS boundary mask. Simulations were performed on a 64^3 grid with bond dimensions $\chi = 104, 103, 102, 101, 100$ and compared to uncompressed LBM with exact $1/\rho$ and 2nd-order $1/\rho$ approximated. To isolate the effect of compressing the flow field on the overall accuracy, the mask bond dimension was fixed at $\chi_{\text{mask}} = 98$, which is sufficient to represent the full 64^3

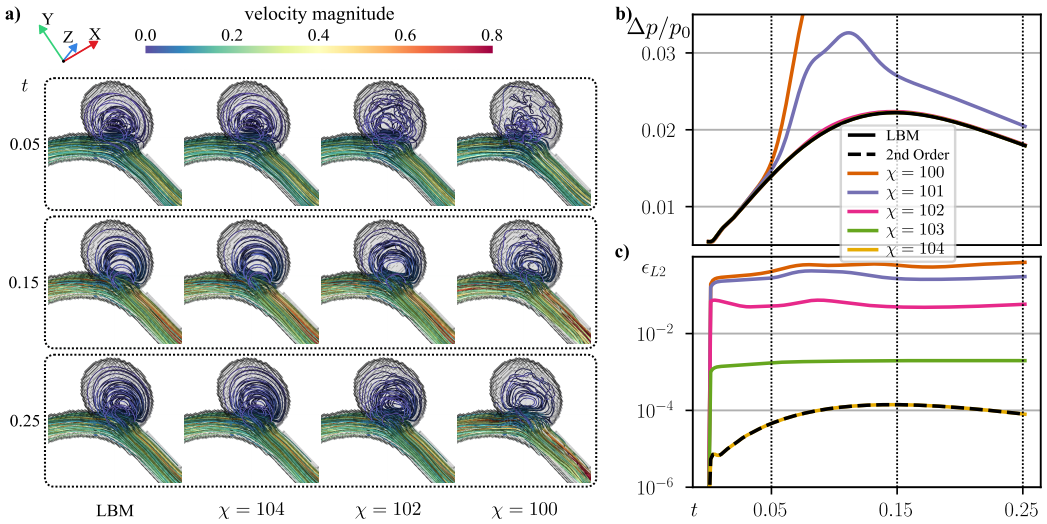

Figure 8: **Results for flow through a 3D aneurysm**. **a)** Streamlines of the flow at $t = 0.05, 0.15, 0.25$, for the fine LBM (first column) and MPS-LBM at $\chi = 104, 102, 100$ (columns 2-4). **b)** Normalized pressure difference $\Delta p/p_0$ averaged over the region of the aneurysm. **c)** Relative l^2 error ϵ_{l^2} with respect to the fine LBM. Solid colored lines represent MPS-LBM, the solid black line fine LBM, the dashed black line fine LBM with second-order approximation of $1/\rho$.

mask to machine precision. The simulation captures the first quarter of a one-second heartbeat, including peak inflow dynamics.

Streamline plots from the LBM simulation at $t = 0.05, 0.15, 0.25$ are shown in the first column of Figure 8a. At $t = 0.05$, the flow exhibits a weak vortical pattern with low velocities. By $t = 0.15$, near peak inflow, the vortex intensifies with higher velocities. At $t = 0.25$, the structure persists but weakens. Compared to this baseline, MPS-LBM at $\chi = 104$ yields nearly identical results. Cases with bond dimensions of $\chi = 100$ and $\chi = 102$ exhibit distorted vortex structures within the aneurysm and fail to reproduce its detailed flow features.

As a relevant measure, we show the average pressure $\Delta p/p_0$ inside the aneurysm normalized with the prescribed outflow pressure in Figure 8b. The standard LBM shows a smooth increase of pressure towards the peak inflow at $t = 0.15$ and a slight decrease afterwards. At $t = 0.05$, the pressures obtained with MPS-LBM at $\chi \leq 101$ diverge from the baseline. At higher bond dimensions, MPS-LBM results are nearly indistinguishable from standard LBM. This suggests that the reproduced vortex structures at $\chi = 102$ can maintain relevant physical aspects of the flow.

The l^2 error ϵ_{l^2} in Figure 8c supports these observations. For $\chi = 100$ and 101,

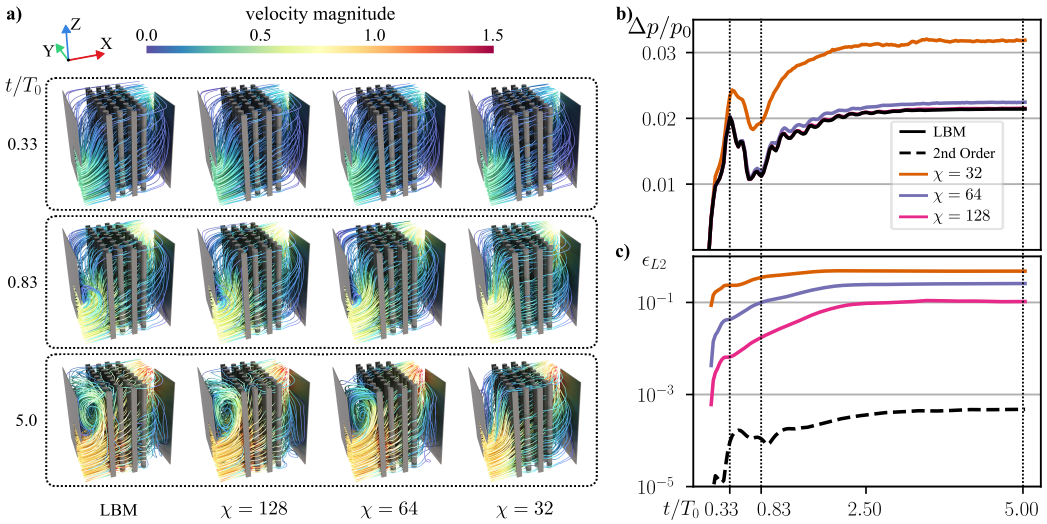

Figure 9: **Results for flow through a pin-fin configuration**. **a)** Streamlines of the flow at $t/T_0 = 0.33, 0.83, 5.0$, for standard LBM (first column) and MPS-LBM at $\chi = 128, 64, 32$ (columns 2-4). **b)** Normalized pressure difference $\Delta p/p_0$ averaged over the inflow region. **c)** Relative l^2-error ϵ_{l^2} with respect to the standard LBM. Solid colored lines represent MPS-LBM, solid black line standard LBM, the dashed black line standard LBM with second-order approximation of $1/\rho$.

errors exceed 0.2 and remain high throughout the simulation. At $\chi = 102$, errors drop by nearly an order of magnitude, and at $\chi = 103$ they stabilize around 10^{-3}. With $\chi = 104$, MPS-LBM becomes indistinguishable from LBM with second-order $1/\rho$ approximation, indicating that MPS compression does not introduce significant additional error at this bond dimension.

Notably, the bond dimension χ of the MPS decomposed particle distribution function must exceed χ_{mask} by only a small margin to achieve vanishing errors. However, as χ approaches χ_{mask}, the simulation results quickly deviate strongly from the reference, suggesting that χ_{mask} serves as a lower bound for computationally meaningful χ. At $\chi = 104$, a compression ratio of $\text{CR}_{104} \approx 2.3$ is achieved.

5.3. Pin-Fin Configuration

A pin-fin array, commonly used in industrial heat exchangers, exhibits translational symmetry and achieves high compression rates with MPS-LBM. Simulations were conducted on a 256^3 grid. We compare MPS-LBM with bond dimensions $\chi = 128, 64, 32$ to standard LBM with exact and 2nd-order $1/\rho$ approximation. The mask bond dimension was set to $\chi_{\text{mask}} = 16$ to ensure machine-precision accuracy. Inflow velocity ramps from zero to $u_{\text{in}} = 1$, and simulations

run until $t/T_0 = 5$, where $T_0 = L/u_{\text{in}}$, allowing the flow to become steady. The resulting vortex structures reveal that the flow becomes increasingly complex at higher Reynolds numbers.

The streamline plots in Figure 9a depict the velocity field of uncompressed LBM and MPS-LBM simulations. At startup $t/T_0 = 0.33$, all simulations capture the flow accurately. By $t/T_0 = 0.83$, vortices begin to form above the inflow, which are resolved by simulations with $\chi \geq 64$. At the final time $t/T_0 = 5$, all cases except $\chi = 32$ reproduce detailed velocity structures, including the inflow vortex. The $\chi = 32$ simulation fails to capture these features.

The normalized pressure drop $\Delta p/p_0$ from inflow to outflow is shown in Figure 9b. All simulations accurately predict the pressure drop up to $t/T_0 = 0.33$. Beyond this point, $\chi = 32$ overshoots, following the overall trend but overestimating even after convergence to the steady state. Other simulations track the reference closely, with $\chi = 64$ slightly overshooting and $\chi = 128$ matching the pressure drop exactly.

Figure 9c shows a clear separation with respect to the examined bond dimensions in terms of the l^2-error ϵ_{l^2} of the velocity field. $\chi = 32$ stabilizes around 0.5, $\chi = 64$ around 0.2 and $\chi = 128$ at 0.1. The error of the standard LBM with 2nd-order $1/\rho$ approximation lies below 10^{-3}, clearly showing that in this test case, the errors are dominated by the MPS decomposition.

Putting this in relation to the achieved compression, we find that at $\chi = 64$ with a compression ratio of $\mathrm{CR}_{64} \approx 120$, MPS-LBM maintains an accuracy of 95% in terms of the inflow-outflow pressure difference. At a compression of $\mathrm{CR}_{128} \approx 42$ with $\chi = 128$ the deviation becomes negligible. Although the exact time evolution of the velocity field is not captured by compressed simulations, cases with $\chi \geq 64$ reproduce the pressure difference accurately, indicating that even with high compression rates, the underlying physics is preserved from startup to steady state.

6. Discussion

Quantum-inspired lattice Boltzmann based on matrix product state decomposition (MPS-LBM) addresses the memory bottleneck of LBM while it preserves high accuracy even for complex flow configurations. By compressing the MPS representation, MPS-LBM allows for substantial reductions in memory without altering the main algorithmic or grid structure. The method supports standard inflow/outflow boundary conditions and can handle arbitrary complex three-dimensional geometries within the computational domain, which was not

demonstrated with MPS-based approaches before. Its capabilities are demonstrated through three 3D simulations designed to benchmark accuracy, flexibility, and compression performance.

The Taylor–Green vortex benchmark under varying compression ratios demonstrates that already at $\mathrm{CR}_{98} = 64$, the temporal evolution of the kinetic energy is accurately reproduced. With $\mathrm{CR}_{256} = 13$, fully resolved benchmark data are recovered. Complex geometries can be handled, as shown in the example of the flow in an aneurysm. The compression ratio of the complex geometry mask imposes a lower bound on the achievable compression of the fluid domain, while only a modest increase in bond dimension beyond that of the mask is required to reproduce the flow accurately. A pin-fin array geometry represents the practical application to a heat exchanger. Exploiting translational symmetry in the boundary mask, MPS-LBM achieves high data compression with accurate predictions. For instance at $\mathrm{CR}_{64} = 120$, pressure drop error is less than 5%, while at $\mathrm{CR}_{128} = 42$ reference data are recovered. While there is not a definitive way to determine a suitable choice of χ a priori, our results strongly indicate that the bond dimension necessary to accurately represent the geometry marks a lower bound. Apart from that further research is necessary to develop heuristics for the choice of χ or methods for automated adaptation. We anticipate that MPS-LBM delivers high computational performance in settings where the compressibility of translational symmetries (see Appendix D for further details) can be exploited to reduce computational cost without compromising accuracy, thereby enabling simulations that would otherwise be infeasible due to computational constraints.

Due to the weakly compressible formulation, MPS-LBM does not require iterative solutions of a Poisson equation. Instead, it relies exclusively on element-wise operations within the MPS decomposition, which ensures that any algorithmic improvements to fundamental MPS operations translate directly into performance gains for MPS-LBM. The modular structure of the underlying LBM framework naturally supports algorithmic extensions, increasing the applicability of MPS-LBM. Heat transport [45], including heat transfer, can be incorporated. Multi-relaxation-time schemes [46] enable higher Reynolds numbers, while multiphase flow models [47] extend the method to chemically relevant systems. Because the MPS decomposition introduces only minimal changes to the core LBM architecture, such extensions are straightforward while significantly expanding the range of MPS-LBM applications.

Another promising direction involves optimizing the decomposition of the binary geometry mask. Instead of minimizing the standard l^2-norm via singular value decomposition, alternative metrics such as the Hamming distance may be

employed. Ideally, the decomposition scheme ensures that the contracted MPS mask remains binary. Advances in this area enable accurate simulations at even higher compression ratios, further extending the efficiency of MPS-LBM.

Finally, the lattice Boltzmann method has recently attracted attention as a candidate for fluid simulations on quantum computers [36, 48, 49]. One of the primary challenges remains the treatment of collision, which is an active area of research [50]. MPS-LBM, particularly the results presented in section B of the supplementary information, reduces this nonlinearity to element-wise multiplications with minimal loss of accuracy. More importantly, our results show that low-rank operations suffice to impose structured geometries onto LBM simulations, implying the feasibility of efficient implementations on quantum hardware.

Acknowledgments

EM and LG would like to thank Hans Hohenfeld, Benedikt Placke, and Maximilian Kiefer-Emmanouilidis for helpful discussions. EM and LG acknowledge funding from the Federal Ministry for Economic Affairs and Energy (BMWE) and the German Aerospace Center (DLR) in the project QuMAL-KI under project number 50RA2208A, and from the Federal Ministry for Research Technology and Space (BMFTR) in the project QuaSA under project number 13N17300 administered by the VDI/VDE Innovation + Technik GmbH (VDI). NAA acknowledges funding from ERC Advanced Grant Project No. 101094463. The funders played no role in study design, data collection, analysis and interpretation of data, or the writing of this manuscript.

Author Contributions

The project was conceptualized by DMW, JMW, EM, and NAA, and planned by LG, DMW, JMW, and EM. DMW and LG developed the approximation for inverse density. LG developed, implemented, and validated the software. Methodological and implementation expertise for LBM was provided by DMW and JMW. Code review and minor development were conducted by DMW. Numerical experiments for the pin-fin heat sink case were planned by DMW and JMW, and those for the aneurysm and TGV cases by LG, DMW, JMW, and EM. LG set up and conducted all numerical experiments and performed data curation. LG and DMW performed post-processing of the data. The results were analyzed and interpreted by LG, DMW, JMW, and EM. DMW and JMW evaluated the physical consistency of the results. LG prepared the figures. The 3D flow visualizations were prepared

by JMW and DMW. LG and DMW wrote the manuscript and the supplementary information, with major revisions by DMW. Revisions to the manuscript were conducted by JMW, EM, and NAA. The project was supervised by JMW, EM, and NAA.

Data Availability

Due to size, the full experimental data will be made available upon reasonable request.

Code Availability

The underlying code for this study is available at [51]

Competing Interest

The authors declare no competing interests.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process.

During the preparation of this work the authors used Microsoft Copilot and OpenAI ChatGPT to improve readability and language. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Appendix A. Two-Dimensional Test Cases

We simulate three benchmark cases: lid-driven cavity, flow around a cylinder, and flow through a porous medium. Each case is computed using both standard LBM and MPS-LBM algorithms at mesh resolutions of 128^2.

All three scenarios are depicted in Figure A.10. The bond dimension for all cases is $\chi = 16$, which corresponds to a compression ratio of CR $= 4.53$. The lid-driven cavity is evaluated at time $t = 10\frac{L}{u_0}$ on a square domain with side length L. The fluid is initially at rest and driven by the top boundary moving at constant velocity, the Reynolds number is Re $= 800$. The flow around a cylinder of diameter d is simulated at Reynolds number Re $= \frac{l_s u_0}{\nu} = 66$. The domain has edge length $L = 16d$, and the velocity magnitude is shown at $t = \frac{5}{3}\frac{L}{u_0}$. Finally,

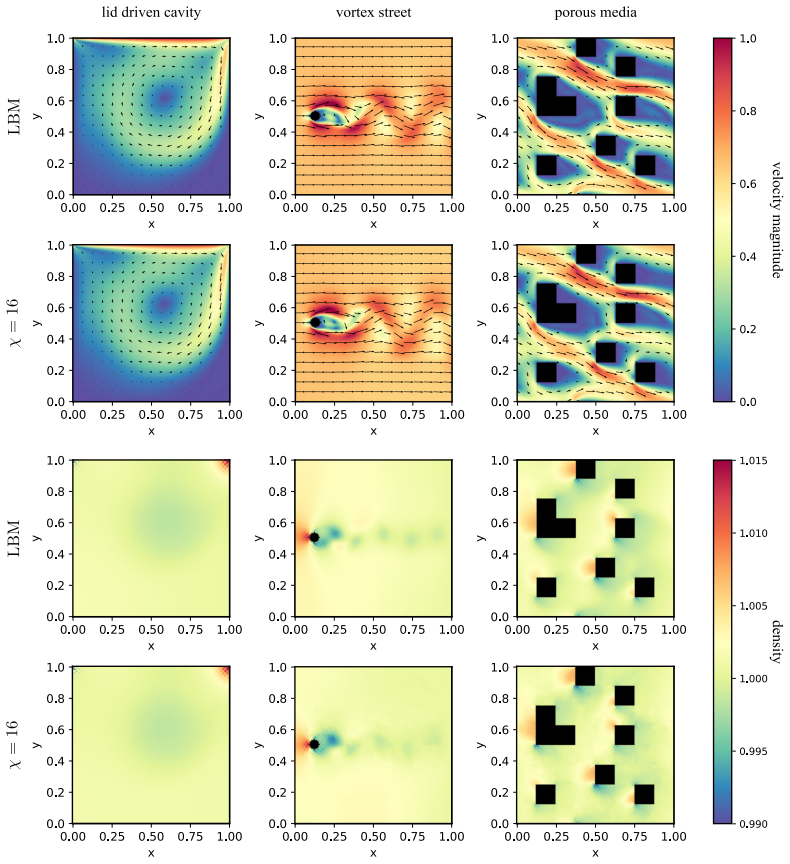

Figure A.10: **Velocity magnitude and density of three common 2D test cases.** Shown from left to right: lid driven cavity, vortex street, and porous media. Top row shows standard LBM simulation, bottom row MPS-LBM at $\chi = 16$ (CR ≈ 4.53). Color map indicates velocity magnitude / density.

flow through a porous medium is driven by a constant volume force $(F_x, F_y)^T = (0.24, -0.1)^T$ N/m^2. Following the Shan–Chen forcing scheme [52], forces are incorporated via a perturbation term in the macroscopic velocity, which enters the equilibrium distribution function as

$$\boldsymbol{u} = \frac{1}{\rho} \sum_i f_i + \frac{\tau \boldsymbol{F}}{\rho}, \tag{A.1}$$

where we apply the same second-order approximation for $1/\rho$. The maximum velocity reaches u_{max}, and the depicted velocity magnitude is shown at $t = 5\frac{L}{u_{\max}}$. For all three scenarios from the reference, a bond dimension of $\chi = 16$ is suffi-

cient to capture the flow dynamics with high fidelity. Only minor deviations are observed in the porous media and cylinder flow case.

Appendix B. Approximation of Inverse Density

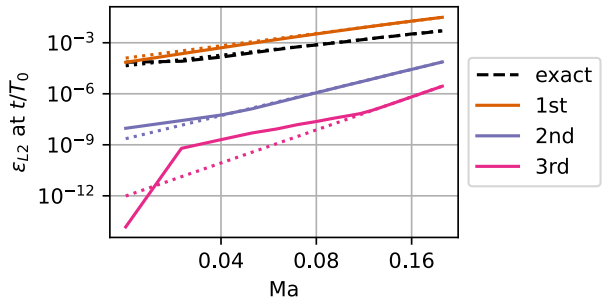

Figure B.11: **Scaling of the l^2-error ϵ_{l^2} with Mach number Ma.** Results are from simulations of the 2D Taylor-Green vortex at different Ma. Solid colored lines show the error introduced by varying orders of approximation of $1/\rho$ with respect to the standard LBM. The dashed black line depicts the error of standard LBM with respect to the analytic solution of the TGV. Dotted lines indicate the respective linear fit.

We conducted a series of two-dimensional Taylor-Green vortex (TGV) simulations with edge length $L = 2\pi$, characteristic velocity $u_0 = 1$, and initial conditions

$$\begin{aligned} u(x,y) &= -\sin(x)\cos(y)\,, \\ v(x,y) &= \cos(x)\sin(y)\,, \\ \rho(x,y) &= 1 + \frac{1}{4\Delta u^2}\left(\cos(2x) + \cos(2y)\right). \end{aligned} \tag{B.1}$$

The Reynolds number was set to $\mathrm{Re} = \frac{u_0 L}{\nu} = 125$. To maintain constant Re while varying Ma, the relaxation time was scaled as $\tau = 0.5 + \frac{N\mathrm{Ma}}{c_s\,\mathrm{Re}}$, with $N = 256$ cells per spatial dimension.

The results obtained using Taylor expansions of $1/\rho$ up to third order were compared with standard LBM solutions and the analytical solution of the 2D TGV.

In Figure B.11, the colored lines show relative errors of the flow fields in the l^2-norm ϵ_{l^2} (see Section 4.2 in the main text) of LBM utilizing Taylor expansions of varying order for $1/\rho$ compared to the standard LBM in a log-log plot. The dashed black line indicates ϵ_{l^2} of the exact LBM compared to the analytic solution. Dotted lines show a linear fit to the respective data in log-log representation. Overall, the

Table B.1: Fitted exponents of the scaling of ϵ_{l^2} with Mach number.

	Exact	**1st**	**2nd**	**3rd**
Exponent	2.059	2.401	4.513	6.457

second-order approximation offers a favorable trade-off between computational efficiency and numerical accuracy relative to the standard approach.

The scaling exponents derived from these numerical experiments are summarized in Table B.1. These were computed via a least squares method. The function $\epsilon_{l^2}(\mathrm{Ma}, \alpha, \beta) = \alpha \mathrm{Ma}^{\beta}$ was fitted to the computed set of data $(\epsilon_k, \mathrm{Ma}_k)$ by minimizing the sum of squared residuals

$$\min_{\alpha,\beta} \sum_i \left(\epsilon_k - \epsilon_{l^2}(\mathrm{Ma}_k, \alpha, \beta)\right)^2 . \tag{B.2}$$

Appendix C. Runtime Measurements

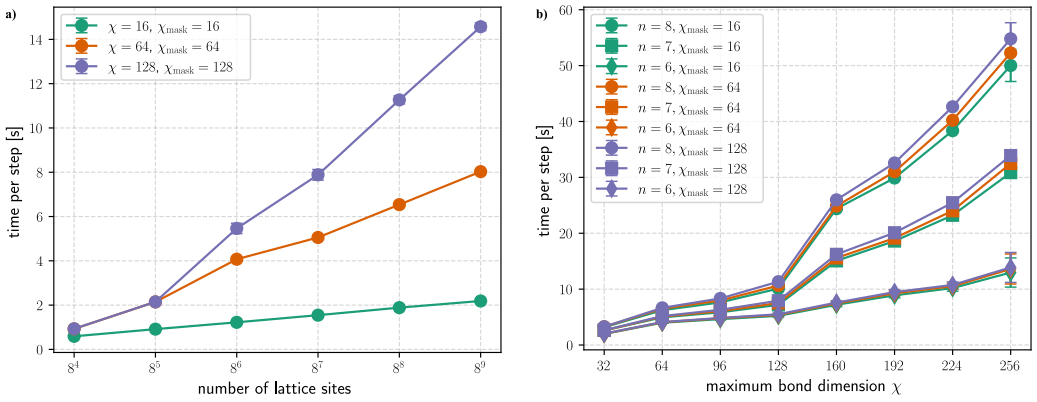

Figure C.12: **Runtime scaling** of **a)** D3Q15 MPS-LBM with respect to the number of lattice sites **b)** and the maximum bond dimension.

In this subsection, we provide data on measured runtimes and discuss them. For all measurements, we performed 100 time steps of the D3Q15 MPS-LBM to compute the mean and standard deviation. In- and outflow conditions, as well as a geometry mask, are enabled. Tensor manipulations were implemented using the Python packages JAX [30] and opt_einsum [53] and all simulations were conducted on an NVIDIA A100 GPU with 80GB of VRAM.

In Figure C.12a, the scaling with respect to the number of lattice sites $N = 8^n$ is shown, where n is the number of sites in each MPS for different bond dimension

$\chi \in \{16, 64, 128\}$. The expected linear scaling in n is clearly visible. The only deviation at $N = 8^4$ and $N = 8^5$ for $\chi = 128$ is explained by the fact that MPS of these sizes are already exact at a bond dimension of 64, and MPS-LBM hence operates on MPS smaller than the allowed maximum bond dimension of $\chi = 128$.

The scaling with the bond dimension, shown in Figure C.12b, is more complex. The scaling with χ^4 cannot clearly be discerned due to an inconsistency between $\chi = 128$ and $\chi = 160$, as well as between $\chi = 224$ and $\chi = 256$. Presumably, these irregularities originate from GPU saturation effects or optimization decisions made by the compiler of JAX [30].

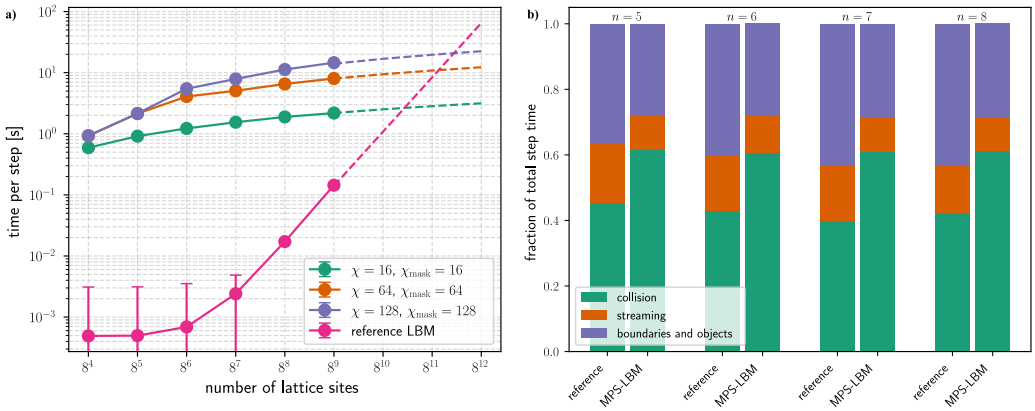

Figure C.13: **Runtime comparison with the reference LBM. a)** Time in seconds per step including extrapolations (dashed lines) of the runtime crossover of MPS-LBM and the reference. **b)** Comparison of the fraction of time spent in the collision, streaming and the handling of obstacles and boundaries.

As comparison with the reference LBM Figure C.13a shows runtimes on a log-scale. Initially the reference LBM is several orders of magnitude faster. After saturation effects of the GPU below $N = 8^7$, the data exhibits the expected exponential growth. Dashed lines show the extrapolation of our data beyond $N = 8^9$. Given our implementation the data suggest that for moderate χ, MPS-LBM exhibits a runtime improvement at system sizes larger than 8^{11}. It is important to note that this observation depends on specific implementation and hardware.

Additionally, in Figure C.13b the fraction of time spent on the collision step, the streaming step and the handling of boundaries and obstacles respectively, is shown for various n. In the reference LBM collision and handling of obstacles and boundaries have a similar cost, while the streaming takes roughly half as much time. In MPS-LBM the cost of the collision increases compared to the reference,

marking this operation a candidate for future improvement. These observations do not vary significantly with varying n.

Appendix D. Symmetric Geometries in Matrix Product State Decomposition

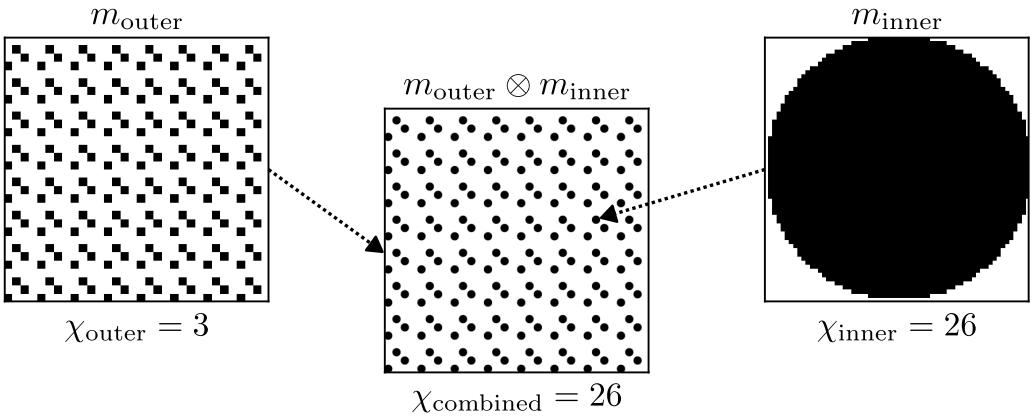

Figure D.14: **Combination of masks on separate length scales.**

We examine the relation between MPS decomposition and translationally symmetric geometries using an explicit example, and show that the approach generalizes to realistic cases. Consider a geometry with separation of large- and small-scale structures. We consider an inner geometry represented as a binary mask m_{inner} of resolution 2^{kD}, repeated according to an outer mask m_{outer} on a 2^{lD} grid. The combined mask inserts the inner geometry at each site where $m_{\text{outer}} = 1$, leaving other sites zero. An example is shown in Figure D.14, where the inner mask is a circle on a 64^2 grid, arranged in the repetition pattern defined by an outer mask on a 32^2 grid. For simplicity, in 1D the value at position x is given by

$$m^x_{\text{combined}} = m^{\lfloor x/2^k \rfloor}_{\text{outer}} \; m^{x \bmod 2^k}_{\text{inner}} = (m_{\text{outer}} \otimes m_{\text{inner}})^x. \tag{D.1}$$

Thus, it is straightforward to see that the combined mask is in fact the tensor product of the outer and inner masks. To relate this structure to MPS, we examine the index representation. Consider the scale-ordered decomposition

$$\begin{aligned} & \boldsymbol{x} = x_1 \dots x_D = b^1_1 \dots b^n_1 \dots b^1_D \dots b^n_D \\ \rightarrow \quad & b^1_1 \dots b^1_D \dots b^n_1 \dots b^n_D = w_1 \dots w_n, \end{aligned} \tag{D.2}$$

as used in Section 2 of the main text. Let us denote the scale ordered indices of m_{inner} and m_{outer} as $\boldsymbol{x}_{\text{in}} \to u_1 \dots u_k$ and $\boldsymbol{x}_{\text{out}} \to v_1 \dots v_l$ and consider both masks to be MPS decomposed, then we can write out the tensor product as

$$\begin{aligned}(m_{\text{outer}} \otimes m_{\text{inner}})^{\boldsymbol{x}_{\text{out}}\boldsymbol{x}_{\text{in}}} &= m_{\text{outer}}^{v_1 \dots v_l} m_{\text{inner}}^{u_1 \dots u_k} \\ &= \sum_{\{\boldsymbol{\alpha}\},\{\boldsymbol{\beta}\}} (\mathbf{A}_1)^{v_1}_{\alpha_1} \dots (\mathbf{A}_l)^{v_l}_{\alpha_{l-1}} \; (\mathbf{B}_1)^{u_1}_{\beta_1} \dots (\mathbf{B}_k)^{u_k}_{\beta_{k-1}} \,.\end{aligned} \tag{D.3}$$

Therefore, the maximum bond dimension of the combined mask equals the larger of the two masks $\chi_{\text{combined}} = \max(\chi_{\text{outer}}, \chi_{\text{inner}})$. In the example shown in Figure D.14, the combined mask has a resolution of 2048^2 and is accurately represented as an MPS with $\chi_{\text{combined}} = 26$. Together with the results from our simulations, this suggests that MPS-LBM would produce physically accurate flows around $\chi \approx 64$ (CR ≈ 46). This construction directly relates to the pin-fin scenario, where the inner mask defines a single pin-fin and the outer mask specifies the array layout. The concept generalizes to other applications.

References

[1] M. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, Journal of Computational Physics 82 (1) (1989) 64–84. doi:10.1016/0021-9991(89)90035-1.
URL https://linkinghub.elsevier.com/retrieve/pii/0021999189900351

[2] A. Harten, Multiresolution algorithms for the numerical solution of hyperbolic conservation laws, Communications on Pure and Applied Mathematics 48 (12) (1995) 1305–1342. doi:10.1002/cpa.3160481201.
URL https://onlinelibrary.wiley.com/doi/10.1002/cpa.3160481201

[3] L. Han, X. Hu, N. Adams, Adaptive multi-resolution method for compressible multi-phase flows with sharp interface model and pyramid data structure, Journal of Computational Physics 262 (2014) 131–152. doi:10.1016/j.jcp.2013.12.061.
URL https://linkinghub.elsevier.com/retrieve/pii/S0021999114000230

[4] J. Kaiser, D. Appel, F. Fritz, S. Adami, N. Adams, A multiresolution local-timestepping scheme for particle–laden multiphase flow simulations using a level-set and point-particle approach, Computer Methods in Applied Mechanics and Engineering 384 (2021) 113966. doi:10.1016/j.cma.2021.113966.
URL https://linkinghub.elsevier.com/retrieve/pii/S0045782521003030

[5] N. Hoppe, S. Adami, N. A. Adams, A parallel modular computing environment for three-dimensional multiresolution simulations of compressible flows, Computer Methods in Applied Mechanics and Engineering 391 (2022) 114486. doi:10.1016/j.cma.2021.114486.
URL https://linkinghub.elsevier.com/retrieve/pii/S004578252100699X

[6] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, Annals of Physics 326 (1) (2011) 96–192. doi:10.1016/j.aop.2010.09.012.
URL https://www.sciencedirect.com/science/article/pii/S0003491610001752

[7] I. V. Oseledets, Tensor-Train Decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317, publisher: Society for Industrial and Applied Mathematics. doi:10.1137/090752286.
URL https://epubs.siam.org/doi/10.1137/090752286

[8] R. Orús, A practical introduction to tensor networks: Matrix product states and projected entangled pair states, Annals of Physics 349 (2014) 117–158. doi:10.1016/j.aop.2014.06.013.
URL https://www.sciencedirect.com/science/article/pii/S0003491614001596

[9] S. R. White, Density matrix formulation for quantum renormalization groups, Physical Review Letters 69 (19) (1992) 2863–2866, publisher: American Physical Society. doi:10.1103/PhysRevLett.69.2863.
URL https://link.aps.org/doi/10.1103/PhysRevLett.69.2863

[10] G. Vidal, Entanglement Renormalization, Physical Review Letters 99 (22) (2007) 220405, publisher: American Physical Society.

doi:10.1103/PhysRevLett.99.220405.
URL https://link.aps.org/doi/10.1103/PhysRevLett.99.220405

[11] F. Verstraete, V. Murg, J. Cirac, Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems, Advances in Physics 57 (2) (2008) 143–224. arXiv:https://doi.org/10.1080/14789940801912366, doi:10.1080/14789940801912366.
URL https://doi.org/10.1080/14789940801912366

[12] N. Gourianov, M. Lubasch, S. Dolgov, Q. Y. van den Berg, H. Babaee, P. Givi, M. Kiffner, D. Jaksch, A quantum-inspired approach to exploit turbulence structures, Nature Computational Science 2 (1) (2022) 30–37, publisher: Nature Publishing Group. doi:10.1038/s43588-021-00181-1.
URL https://www.nature.com/articles/s43588-021-00181-1

[13] J. Eisert, M. Cramer, M. B. Plenio, Colloquium: Area laws for the entanglement entropy, Rev. Mod. Phys. 82 (2010) 277–306. doi:10.1103/RevModPhys.82.277.
URL https://link.aps.org/doi/10.1103/RevModPhys.82.277

[14] M. Kiffner, D. Jaksch, Tensor network reduced order models for wall-bounded flows, Physical Review Fluids 8 (12) (2023) 124101, publisher: American Physical Society. doi:10.1103/PhysRevFluids.8.124101.
URL https://link.aps.org/doi/10.1103/PhysRevFluids.8.124101

[15] I. V. Oseledets, S. V. Dolgov, Solution of Linear Systems and Matrix Inversion in the TT-Format, SIAM Journal on Scientific Computing 34 (5) (2012) A2718–A2739, publisher: Society for Industrial and Applied Mathematics. doi:10.1137/110833142.
URL https://epubs.siam.org/doi/abs/10.1137/110833142

[16] R. D. Peddinti, S. Pisoni, A. Marini, P. Lott, H. Argentieri, E. Tiunov, L. Aolita, Quantum-inspired framework for computational fluid dynamics, Communications Physics 7 (1) (2024) 1–7, publisher: Nature Publishing Group. doi:10.1038/s42005-024-01623-8.
URL https://www.nature.com/articles/s42005-024-01623-8

[17] N.-L. v. Hülst, P. Siegl, P. Over, S. Bengoechea, T. Hashizume, M. G. Cecile, T. Rung, D. Jaksch, Quantum-Inspired Tensor-Network Fractional-Step Method for Incompressible Flow in Curvilinear Coordinates, arXiv:2507.05222 [physics] version: 1 (Jul. 2025). doi:10.48550/arXiv.2507.05222.
URL http://arxiv.org/abs/2507.05222

[18] L. Hölscher, P. Rao, L. Müller, J. Klepsch, A. Luckow, T. Stollenwerk, F. K. Wilhelm, Quantum-inspired fluid simulation of two-dimensional turbulence with GPU acceleration, Physical Review Research 7 (1) (2025) 013112, publisher: American Physical Society. doi:10.1103/PhysRevResearch.7.013112.
URL https://link.aps.org/doi/10.1103/PhysRevResearch.7.013112

[19] B. Ghahremani, H. Babaee, Cross interpolation for solving high-dimensional dynamical systems on low-rank Tucker and tensor train manifolds, Computer Methods in Applied Mechanics and Engineering 432 (2024) 117385. doi:10.1016/j.cma.2024.117385.
URL https://linkinghub.elsevier.com/retrieve/pii/S0045782524006406

[20] Q. Thai Tran, D. P. Truong, Kim Ø Rasmussen, B. Alexandrov, A tensor train-based isogeometric solver for large-scale 3D poisson problems, Computer Methods in Applied Mechanics and Engineering 453 (2026) 118802. doi:10.1016/j.cma.2026.118802.
URL https://linkinghub.elsevier.com/retrieve/pii/S0045782526000769

[21] 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.

doi:10.1103/PhysRev.94.511.
URL https://link.aps.org/doi/10.1103/PhysRev.94.511

[22] A. Dupuis, B. Chopard, Theory and applications of an alternative lattice Boltzmann grid refinement algorithm, Physical Review E 67 (6) (2003) 066707. doi:10.1103/PhysRevE.67.066707.
URL https://link.aps.org/doi/10.1103/PhysRevE.67.066707

[23] G. Eitel-Amor, M. Meinke, W. Schröder, A lattice-Boltzmann method with hierarchically refined meshes, Computers & Fluids 75 (2013) 127–139. doi:10.1016/j.compfluid.2013.01.013.
URL https://linkinghub.elsevier.com/retrieve/pii/S0045793013000315

[24] Z. Liu, F.-B. Tian, X. Feng, An efficient geometry-adaptive mesh refinement framework and its application in the immersed boundary lattice Boltzmann method, Computer Methods in Applied Mechanics and Engineering 392 (2022) 114662. doi:10.1016/j.cma.2022.114662.
URL https://linkinghub.elsevier.com/retrieve/pii/S0045782522000573

[25] T. Bellotti, L. Gouarin, B. Graille, M. Massot, Multidimensional fully adaptive lattice Boltzmann methods with error control based on multiresolution analysis, Journal of Computational Physics 471 (2022) 111670. doi:10.1016/j.jcp.2022.111670.
URL https://linkinghub.elsevier.com/retrieve/pii/S0021999122007331

[26] F. Verstraete, J. I. Cirac, Renormalization algorithms for quantum-many body systems in two and higher dimensions (2004). arXiv:cond-mat/0407066.
URL https://arxiv.org/abs/cond-mat/0407066

[27] E. M. Stoudenmire, S. R. White, Minimally entangled typical thermal state algorithms, New Journal of Physics 12 (5) (2010) 055026. doi:10.1088/1367-2630/12/5/055026.
URL https://doi.org/10.1088/1367-2630/12/5/055026

[28] A. A. Michailidis, C. Fenton, M. Kiffner, Tensor Train Multiplication, arXiv:2410.19747 [physics] (Oct. 2024). doi:10.48550/arXiv.2410.19747.
URL http://arxiv.org/abs/2410.19747

[29] J. Unfried, J. Hauschild, F. Pollmann, Fast time evolution of matrix product states using the qr decomposition, Phys. Rev. B 107 (2023) 155133. doi:10.1103/PhysRevB.107.155133.
URL https://link.aps.org/doi/10.1103/PhysRevB.107.155133

[30] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018).
URL http://github.com/jax-ml/jax

[31] V. A. Kazeev, B. N. Khoromskij, Low-Rank Explicit QTT Representation of the Laplace Operator and Its Inverse, SIAM Journal on Matrix Analysis and Applications 33 (3) (2012) 742–758. doi:10.1137/100820479.
URL http://epubs.siam.org/doi/10.1137/100820479

[32] H. Chen, S. Chen, W. H. Matthaeus, Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method, Physical Review A 45 (8) (1992) R5339–R5342. doi:10.1103/PhysRevA.45.R5339.
URL https://link.aps.org/doi/10.1103/PhysRevA.45.R5339

[33] S. Chen, G. D. Doolen, LATTICE BOLTZMANN METHOD FOR FLUID FLOWS, Annual Review of Fluid Mechanics 30 (1) (1998) 329–364. doi:10.1146/annurev.fluid.30.1.329.
URL https://www.annualreviews.org/doi/10.1146/annurev.fluid.30.1.329

[34] T. Abe, Derivation of the Lattice Boltzmann Method by Means of the Discrete Ordinate Method for the Boltzmann Equation, Journal of Computational Physics 131 (1) (1997) 241–246. doi:10.1006/jcph.1996.5595.
URL https://linkinghub.elsevier.com/retrieve/pii/S0021999196955953

[35] X. He, L.-S. Luo, Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation, Physical Review E 56 (6) (1997) 6811–6817. doi:10.1103/PhysRevE.56.6811.
URL https://link.aps.org/doi/10.1103/PhysRevE.56.6811

[36] C. Sanavio, S. Succi, Lattice boltzmann–carleman quantum algorithm and circuit for fluid flows at moderate reynolds number, AVS Quantum Science 6 (2) (2024) 023802. doi:10.1116/5.0195549.
URL https://doi.org/10.1116/5.0195549

[37] X. He, L.-S. Luo, Lattice boltzmann model for the incompressible navier–stokes equation, Journal of statistical Physics 88 (3) (1997) 927–944.

[38] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E. M. Viggen, The Lattice Boltzmann Method: Principles and Practice, Graduate Texts in Physics, Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-44649-3.
URL http://link.springer.com/10.1007/978-3-319-44649-3

[39] X. He, Q. Zou, L.-S. Luo, M. Dembo, Analytic solutions of simple flows and analysis of nonslip boundary conditions for the lattice Boltzmann BGK model, Journal of Statistical Physics 87 (1-2) (1997) 115–136. doi:10.1007/BF02181482.
URL http://link.springer.com/10.1007/BF02181482

[40] D. P. Ziegler, Boundary conditions for lattice Boltzmann simulations, Journal of Statistical Physics 71 (5-6) (1993) 1171–1177. doi:10.1007/BF01049965.
URL http://link.springer.com/10.1007/BF01049965

[41] S. Izquierdo, N. Fueyo, Characteristic nonreflecting boundary conditions for open boundaries in lattice Boltzmann methods, Physical Review E 78 (4) (2008) 046707. doi:10.1103/PhysRevE.78.046707.
URL https://link.aps.org/doi/10.1103/PhysRevE.78.046707

[42] P. Nathen, D. Gaudlitz, M. Krause, N. Adams, On the stability and accuracy of the bgk, mrt and rlb boltzmann schemes for the simulation of turbu-

lent flows, Communications in Computational Physics 23 (2018) 846–876. doi:10.4208/cicp.OA-2016-0229.

[43] M. Horvat, S. B. Lunowa, D. Sytnyk, B. Wohlmuth, A lattice boltzmann method for non-newtonian blood flow in coiled intracranial aneurysms, in: A. Sequeira, A. Silvestre, S. S. Valtchev, J. Janela (Eds.), Numerical Mathematics and Advanced Applications ENUMATH 2023, Volume 1, Springer Nature Switzerland, Cham, 2025, pp. 473–483.

[44] A. Shahsavar, M. Shahmohammadi, I. B. Askari, Cfd simulation of the impact of tip clearance on the hydrothermal performance and entropy generation of a water-cooled pin-fin heat sink, International Communications in Heat and Mass Transfer 126 (2021) 105400. doi:https://doi.org/10.1016/j.icheatmasstransfer.2021.105400.
URL https://www.sciencedirect.com/science/article/pii/S0735193321002931

[45] H. Karani, C. Huber, Lattice boltzmann formulation for conjugate heat transfer in heterogeneous media, Phys. Rev. E 91 (2015) 023304. doi:10.1103/PhysRevE.91.023304.
URL https://link.aps.org/doi/10.1103/PhysRevE.91.023304

[46] D. d'Humières, I. Ginzburg, M. Krafczyk, P. Lallemand, L.-S. Luo, Multiple-relaxation-time lattice boltzmann models in three dimensions, Philosophical Transactions of the Royal Society A 360 (1792) (2002) 437–451. doi:10.1098/rsta.2001.0955.

[47] M. R. Swift, E. Orlandini, W. R. Osborn, J. M. Yeomans, Lattice boltzmann simulations of liquid-gas and binary fluid systems, Phys. Rev. E 54 (1996) 5041–5052. doi:10.1103/PhysRevE.54.5041.
URL https://link.aps.org/doi/10.1103/PhysRevE.54.5041

[48] A. Tiwari, J. Iaconis, J. Jojo, S. Ray, M. Roetteler, C. Hill, J. Pathak, Algorithmic Advances Towards a Realizable Quantum Lattice Boltzmann Method, arXiv:2504.10870 [quant-ph] (Apr. 2025). doi:10.48550/arXiv.2504.10870.
URL http://arxiv.org/abs/2504.10870

[49] D. Wawrzyniak, J. Winter, S. Schmidt, T. Indinger, C. F. Janßen, U. Schramm, N. A. Adams, A quantum algorithm for the lattice-Boltzmann method advection-diffusion equation, Computer Physics Communications 306 (2025) 109373. doi:10.1016/j.cpc.2024.109373.
URL https://www.sciencedirect.com/science/article/pii/S0010465524002960

[50] F. Tennie, S. Laizet, S. Lloyd, L. Magri, Quantum computing for nonlinear differential equations and turbulence, Nature Reviews Physics 7 (4) (2025) 220–230. doi:10.1038/s42254-024-00799-w.
URL https://doi.org/10.1038/s42254-024-00799-w

[51] L. Gross, D. M. Wawrzyniak, MPS-LBM (2026).
URL https://github.com/dfki-ric-quantum/MPS-LBM

[52] X. Shan, H. Chen, Lattice Boltzmann model for simulating flows with multiple phases and components, Physical Review E 47 (3) (1993) 1815–1819. doi:10.1103/PhysRevE.47.1815.
URL https://link.aps.org/doi/10.1103/PhysRevE.47.1815

[53] D. G. a. Smith, J. Gray, opt_einsum - a python package for optimizing contraction order for einsum-like expressions, Journal of Open Source Software 3 (26) (2018) 753. doi:10.21105/joss.00753.
URL https://doi.org/10.21105/joss.00753