Computers and Fluids 272 (2024) 106191

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Lattice Boltzmann method with artificial bulk viscosity using a neural collision operator

Jan Tobias Horstmann [a,*], Mario Christopher Bedrunka [b,c], Holger Foysi [b]

[a] German Aerospace Center (DLR), Institute of Propulsion Technology, Bismarckstraße 101, 10650 Berlin, Germany
[b] Department of Mechanical Engineering, University of Siegen, Paul-Bonatz-Straße 9-11, 57076 Siegen-Weidenau, Germany
[c] Institute of Technology, Resource and Energy-efficient Engineering (TREE), Bonn-Rhein-Sieg University of Applied Science, Grantham-Allee 20, 53757 Sankt Augustin, Germany

ARTICLE INFO

Keywords:
Neural collision operator
MRT-LBM
Artificial bulk viscosity
Numerical stability

ABSTRACT

The lattice Boltzmann method (LBM) stands apart from conventional macroscopic approaches due to its low numerical dissipation and reduced computational cost, attributed to a simple streaming and local collision step. While this property makes the method particularly attractive for applications such as direct noise computation, it also renders the method highly susceptible to instabilities. A vast body of literature exists on stability-enhancing techniques, which can be categorized into selective filtering, regularized LBM, and multi-relaxation time (MRT) models. Although each technique bolsters stability by adding numerical dissipation, they act on different modes. Consequently, there is not a universal scheme optimally suited for a wide range of different flows. The reason for this lies in the static nature of these methods; they cannot adapt to local or global flow features. Still, adaptive filtering using a shear sensor constitutes an exception to this. For this reason, we developed a novel collision operator that uses space- and time-variant collision rates associated with the bulk viscosity. These rates are optimized by a physically informed neural net. In this study, the training data consists of a time series of different instances of a 2D barotropic vortex solution, obtained from a high-order Navier–Stokes solver that embodies desirable numerical features. For this specific text case our results demonstrate that the relaxation times adapt to the local flow and show a dependence on the velocity field. Furthermore, the novel collision operator demonstrates a better stability-to-precision ratio and outperforms conventional techniques that use an empirical constant for the bulk viscosity.

1. Introduction

Computational fluid dynamics (CFD) has become an indispensable technology in the field of aeronautics [2,3]. For the vast majority of fluid dynamic problems in the aviation industry, the motion of a fluid may accurately be described by the Navier–Stokes (NS) equations, which translate to the macroscopic conservation of mass and momentum. Despite the simplified view of a fluid as a continuum, accurate fully-resolved direct numerical (DNS) or large-eddy (LES) simulations at operating conditions (*i.e.* subject to high Reynolds numbers) are still cost-prohibitive due to the need for fine models imposed by turbulence. The chaotic motions in a flow that cover a wide range of length scales (the ratio of the smallest to the largest scales in one dimension is estimated to scale with $Re^{3/4}$ in isotropic turbulence) must not be ignored as they are crucial for the correct prediction of fluid behavior. The presence of turbulence thus puts severe constraints on the spatio-temporal resolution of a simulation. In a DNS of wall-bounded flows, the number of grid points scales with $\sim Re^{37/14}$ [4]. Within the Reynolds number range relevant to the aircraft industry ($Re = 10^7 - 10^9$) such a simulation would require $10^8 - 10^{10}$ CPU hours on today's supercomputers [5]. Regardless of the infeasibility of such a simulation due to time and current HPC constraints, the energy requirements for such an endeavour would be unjustifiable in the age of climate change[1]. It is therefore more than ever necessary to develop more efficient simulation methods that maintain a high fidelity. A less cost-intensive alternative to DNS are large eddy simulations (LES), where only the large, often anisotropic turbulent structures are resolved, and the unresolved small scales are accounted for by a so-called sub-grid scale model. However, a recent study has shown that simulating an entire aircraft at operating conditions would still take 600 years to accomplish using modern HPC

* Corresponding author.
E-mail address: horstmann.tobias@googlemail.com (J.T. Horstmann).

[1] A single CPU node of the DLR HPC cluster CARO consumes 600 watts [1], amounting to roughly 15 kWh per day, which is more than the energy consumption of the average European household (10 kWh per day).

https://doi.org/10.1016/j.compfluid.2024.106191
Received 27 September 2023; Received in revised form 20 December 2023; Accepted 26 January 2024
Available online 2 February 2024

infrastructure [5]. At present, such a task can only be achieved within a reasonable turnaround time using the Reynolds-averaged Navier–Stokes (RANS) approach. In the case of steady RANS, for example, the flow variables are temporally averaged resulting in unclosed terms due to nonlinearity (e.g., Reynolds stress terms for the momentum equations). However, because of this simplified representation of turbulence, it often fails to correctly predict turbulence transition and flow separation; two phenomena that largely influence lift and drag characteristics, for example.

The ultimate goal in CFD is, therefore, to achieve high-fidelity results at reasonable costs. In 2013, NASA ventured a quick look at the CFD landscape of the year 2030, concluding that LES-based methods will still not be feasible by then [6]. However, the past 10 years have seen two important developments in the field of CFD.

On the one hand, further advancement of the lattice Boltzmann method has led to its establishment as a valuable LES tool, outperforming conventional (continuum-based) approaches in this discipline by a factor of five to ten [7–9].

On the other hand, the recent burst of interest in machine learning (ML) has led to a reinvigorated thrust for innovation in the CFD community [10].

Successful implementations of machine learning in CFD encompass (but are not limited to) areas such as turbulence modeling [11,12], optimization and control [13], and the development of Physics-Informed Neural Networks (PINNs), which offer a novel approach to solving complex fluid dynamics problems by incorporating underlying physical principles directly into their learning algorithms [14,15]. Consequently, it did not take long for research groups to begin exploring the integration of LBM and ML, to expedite achieving the grand challenges in CFD, as mentioned in [6]. It is critical in this domain to distinguish between surrogate models and hybrid models, where deep learning is applied to only a part of the governing equation. The former, like PINNs, have demonstrated the ability to both solve forward simulations as well as infer solutions from sparse velocity fields without relying on boundary conditions [16]. Another innovative approach is Lat-Net, employing convolutional layers to efficiently compress flow field information [17]. Despite their promising results, these models face challenges in generalizing across different flow conditions, particularly varying Reynolds numbers and levels of turbulence. Furthermore, ensuring adherence to the governing equations (in particular in case of Lat-Net) remains challenging in the complete replacement of these equations with machine learning models. Due to the high efficiency and parallelizability of the streaming step in LBM, an alternative to surrogate models is to leverage deep learning exclusively for the collision operator. Previous research by Corbetta et al. [18] and Prins [19] demonstrated that the Bhatnagar–Gross–Krook (BGK) operator can be effectively learned using a deep neural network, which takes the pre-collision states of the distribution functions as input and predicts the post-collision states. Another study by Bedrunka et al. [20] explored using deep learning to optimize the non-hydrodynamic relaxation times of a multi-relaxation time (MRT) collision operator. A particularity of this approach is that the relaxation times are space and time dependent and vary as a function of the local velocity moments. They showed that their neural collision operator outperforms the classical MRT model in terms of stability and accuracy.

In this study we adopt a very similar approach, aiming to learn a function that maps the local velocity moments to the relaxation times associated with the bulk viscosity. Our focus on this aspect is driven by two primary factors: firstly, a locally increased bulk viscosity provides a general mechanism for stabilizing the simulation of weakly compressible [21] and compressible [22] flows (shock wave smearing). Secondly, as demonstrated in [23], a velocity dependent bulk viscosity can mitigate the violation of Galilean invariance inherent in the standard BGK-LBM approach.

Through this research, we seek to expand on the study of neural collision operators in LBM. A novel aspect of this study lies in the training data, which stems from a Galilean invariant, isothermal Navier–Stokes solver. We will therefore address the question of how far it is possible to transfer numerical properties between different fluid dynamic models. The paper is organized as follows: the theoretical part briefly recaps the different stability enhancing measures that are commonly used in LBM. This is followed by a presentation of the isothermal finite-volume Navier–Stokes (FV-NS) solver that is used to generate the ground truth data (Section 2.5). The theory part concludes with a section on the machine learning framework applied in this study (Section 2.8) together with an overview of the ML-LBM algorithm (Section 2.9). Section 3 presents the results of the neural collision operator. A conclusion is drawn in Section 4.

2. Theory

The Navier–Stokes equations (NSE) are the macroscopic equivalent to the Boltzmann equation (BE) up to second order in the Knudsen number $\mathcal{O}(\mathrm{Kn}^2)$, as demonstrated by a Chapman–Enskog analysis [24]. Defined as the ratio of mean free path to the representative physical length scale, the Knudsen number is usually very small for the large part of aeronautic applications ($\mathrm{Kn} \ll 0.01$). In such cases, both models exhibit a similar level of fidelity. However, to make the Boltzmann approach accessible and computationally competitive to conventional methods in CFD, the number of particle velocities must be reduced to a finite set (lattice), resulting in the discrete velocity Boltzmann equation (DVBE). Using a standard lattice like the D2Q9 this reduction introduces a cubic error term into the momentum and energy equations, limiting the standard approach to isothermal, weakly compressible flows. This is in contrast to the Navier–Stokes numerical solver, where the discretization of velocity space does not apply and the numerical characteristics are solely determined by the discretization of space and time. The following sections will summarize the most prominent techniques to control the numerical behavior of the lattice Boltzmann method and present the discrete Navier–Stokes model used in this study. The theory is restricted to a consideration in two-dimensional space for brevity, and all formulas are presented in dimensionless, or rather lattice units.

2.1. Standard LBM

The lattice Boltzmann method that is usually found in literature and referred to as *standard* LBM is a simple update rule for the populations f_α residing on a D2Q9 lattice, where α indicates the lattice-velocity direction. In the absence of a source term, it can be written (in lattice units) as

$$f_\alpha(\boldsymbol{x}+\boldsymbol{c}_\alpha, t+1) = f_\alpha(\boldsymbol{x},t) - \frac{1}{\tau}\left[f(\boldsymbol{x},t) - f_\alpha^{\mathrm{eq}}(\boldsymbol{x},t)\right], \tag{1}$$

where the right-hand side of (1) describes the collision of the populations using the BGK operator. The post-collision states are then streamed to the neighboring nodes during one time step. The discrete Maxwellian f_α^{eq} provides a probability for a resting fluid of finding a particle at position $\boldsymbol{x}$ and time t, traveling at the discrete velocity $\boldsymbol{c}_\alpha$. It is usually truncated at second order, *i.e.*

$$f_\alpha^{eq} = \rho\omega_\alpha\left(1 + \frac{\boldsymbol{c}_\alpha\cdot\boldsymbol{u}}{c_0^2} + \frac{(\boldsymbol{c}_\alpha\cdot\boldsymbol{u})^2}{2c_0^4} - \frac{|u|^2}{2c_0^2} + \mathcal{O}(\mathrm{Ma}^3)\right), \tag{2}$$

where $c_0 = \sqrt{1/3}$ is the isothermal speed of sound (in lattice units). During collision, the populations f_α are relaxed towards this probability value, controlled by the relaxation time

$$\tau = \tau_s + \frac{1}{2}, \tag{3}$$

with $\tau_s = \nu/c_0^2$, and ν being the kinematic viscosity of the fluid (in lattice units). This type of collision is referred to as single relaxation time (SRT) collision. An alternative formulation to (1) involves a collision in moment space such that

$$f_\alpha(\boldsymbol{x}+\boldsymbol{c}_\alpha, t+1) = \mathrm{row}_\alpha(\boldsymbol{T}^{-1})\cdot\left\{\boldsymbol{m}(\boldsymbol{x},t) - \boldsymbol{\Lambda}\cdot\left[\boldsymbol{m}(\boldsymbol{x},t) - \boldsymbol{m}^{\mathrm{eq}}(\boldsymbol{x},t)\right]\right\}, \tag{4}$$

where $\mathcal{T}$ is a transformation matrix that maps the population vector $\boldsymbol{f}$ and $\boldsymbol{f}^{eq}$ onto moment space, *i.e.*

$$\boldsymbol{m} = \mathcal{T} \cdot \boldsymbol{f} \quad \text{and} \quad \boldsymbol{m}^{eq} = \mathcal{T} \cdot \boldsymbol{f}^{eq}. \tag{5}$$

The relaxation rate is now prescribed in a diagonal collision matrix $\boldsymbol{\Lambda}_{\text{SRT}}$ containing the relaxation frequency $\omega = 1/\tau$ on the diagonal. Concretely,

$$\boldsymbol{\Lambda}_{\text{SRT}} = \text{diag}(\underbrace{\omega, \dots, \omega}_{q-\text{times}}), \tag{6}$$

where q is the number of discrete velocity directions. The subscript SRT indicates that all moments are, *a priori*, relaxed at the same rate. It should be noted that depending on the choice of $\mathcal{T}$ the properties of the model may slightly vary compared to (1). This is particularly true when using central moments [25] or cumulants [26] instead of raw moments. This, however, shall not be the subject of this study. Due to the low dissipation rate, the standard LBM is not very robust in withstanding disturbances (errors) that are introduced during the discretization process. It is for that reason that (1) or rather (4) are mostly used in the academic context and LBM solvers used for simulations at industrial scale typically employ some sort of stability enhancing techniques that will briefly be presented in the following.

2.2. Filtered LBM

In order to widen the stability envelop of LBM one can either increase the numerical dissipation of certain modes (e.g. shear, acoustic, and or ghost) or specifically address and reduce the errors introduced during the discretization process, or do both. The universal means of choice to increase numerical dissipation is selective filtering [27,28], which can be applied to the distribution functions f_α or the moments m_α. Choosing ϕ as placeholder, a classical filtering operation reads

$$\widehat{\phi} = \phi - \sigma \sum_{j}^{\text{D}} \sum_{n=-N}^{N} d_n \phi(\boldsymbol{x} + n\boldsymbol{x}_j, t), \tag{7}$$

with $\boldsymbol{x}_j$ being the unit vector, σ denotes the strength of the filter, N is the number of points of the stencil and d_n are the filter coefficients.

Obviously, other possibilities exist to increase the numerical dissipation, including time-splitting [29] and off-lattice schemes [30–32]. Nevertheless, these methods – in particular the latter – compromise the efficiency of the steam-and-collide algorithm [33, p. 81].

2.3. Regularized LBM

The second technique specifically addresses a major source of instability in LBM, which is the presence of so-called *ghost modes*. Due to a disparity between the number of distribution functions and macroscopic variables that are relevant to Navier–Stokes dynamics (density, velocity and strain rate tensor), the solution space in standard LBM contains *non-hydrodynamic* moments that can lead to instabilities if not controlled properly. A very simple but elegant method to damp these non-hydrodynamic moments is to reconstruct the non-equilibrium distribution f^{neq} only from the strain rate tensor $\boldsymbol{\Pi}^{\text{neq}}$, i.e.

$$f^{\text{neq}} \approx f^{(1)} = w_\alpha \frac{1}{2c_0^4} \boldsymbol{\mathcal{H}}^{(2)} : \boldsymbol{\Pi}^{\text{neq}}, \tag{8}$$

where $\boldsymbol{\mathcal{H}}^{(2)} = \boldsymbol{c}_\alpha \boldsymbol{c}_\alpha - c_0^2 \boldsymbol{I}$ is a second-order Hermite polynomial and ":" denotes a full index contraction. This regularized LBM was first proposed in [34]. The early truncation nevertheless entails a loss of relevant information. A higher order truncation without reintroducing the ghost modes relies on a recursive relation that enables an approximation of any non-equilibrium moment of order n from the equilibrium moment of order $n+1$ [35]. A last improvement to regularized LBM applies a blending function to the strain rate tensor, so that a small fraction of $\boldsymbol{\Pi}^{\text{neq}}$ is computed by a macroscopic finite difference scheme [36], *i.e.*

$$\boldsymbol{\Pi}^{neq}_{\text{HRR}} = \sigma_{\text{HRR}} \sum_{\alpha=1}^{q} \boldsymbol{c}_\alpha \boldsymbol{c}_\alpha f_\alpha^{\text{neq}} + (1 - \sigma_{\text{HRR}}) \left[-\mu \left(\nabla_h \boldsymbol{u} + \nabla_h \boldsymbol{u}^T \right) \right], \tag{9}$$

where ∇_h denotes here a second-order finite difference operator. This approach is referred to as hybrid recursive regularization (HRR) and has proven to yield stable results at Reynolds numbers beyond 10^6 [37].

2.4. MRT-LBM

The last stability enhancing method and certainly the most versatile one is the multi-relaxation time (MRT) approach. It relies on a relaxation in moment space, while the relaxation times as in (6) are not identical but tuned individually in order to specifically control certain moments. Using a tensor product (TP) basis in the transformation matrix [38], the non-equilibrium moment vector reads

$$\boldsymbol{m}^{\text{neq}} = \mathcal{T}_{\text{TP}} \cdot \boldsymbol{f}^{\text{neq}} = \left[0, 0, 0, \Pi_{xx}^{\text{neq}}, \Pi_{yy}^{\text{neq}}, \Pi_{xy}^{\text{neq}}, Q_{xxy}^{\text{neq}}, Q_{xyy}^{\text{neq}}, A_{xxyy}^{\text{neq}}\right]^T, \tag{10}$$

where the subscript identifies a unique moment in 2D, *i.e.* $Q_{xxy}^{\text{neq}} = \sum_{\alpha=1}^{q} c_{x,\alpha} c_{x,\alpha} c_{y,\alpha} f_\alpha^{neq}$. The first three off-equilibrium moments are zero in accordance with the solvability conditions of standard LBM, *i.e.*

$$m_1^{\text{neq}} = \sum_{\alpha=1}^{q} f_\alpha^{\text{neq}} = 0 \quad \text{and} \quad m_i^{\text{neq}} = \sum_{\alpha=1}^{q} c_{i,\alpha} f_\alpha^{\text{neq}} = 0. \tag{11}$$

The moments four to six are the three distinct elements of the symmetric stain rate tensor. The last three elements are the off-equilibrium parts of the aforementioned non-hydrodynamic moments, or rather ghost modes. Thus, one possibility to attenuate the ghost modes is to set the respective relaxation frequencies equal to 1, *i.e.*

$$\boldsymbol{\Lambda}_{\text{MRT}} = \text{diag}\left(\omega_1, \omega_x, \omega_y, \omega_{xx}, \omega_{yy}, \omega_{xy}, 1, 1, 1\right), \tag{12}$$

so that their post-collision state equals the equilibrium moment (*cf.* (4)). Actually, this is equivalent to the standard regularization procedure described in the previous section. Note that in the above matrix the remaining frequencies are all equal to ω as defined in (3). For the sake of better comprehensibility we have adopted the same notation as used for the elements of the moment vector $\boldsymbol{m}^{\text{neq}}$. The advantage of MRT-LBM lies in the fact that the relaxation frequencies can be chosen between 1 and ω and can be tuned individually. This, however, has lead to the publication of numerous studies with a different choice of relaxation time values and there still seems to be no consensus about the *optimal* parameterization. As a matter of fact this poses an interesting optimization problem, which can be addressed using ML techniques as demonstrated in [20].

As stated in the introduction, a flow quantity that is important to the stability of LBM is the bulk viscosity. If we separate the dynamic viscosity μ from the bulk viscosity μ_b in the definition of the strain rate tensor we obtain following expression

$$\boldsymbol{\Pi}^{\text{neq}} = \mu \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T - \frac{2}{\text{D}} (\nabla \cdot \boldsymbol{u}) \boldsymbol{I} \right) + \mu_b (\nabla \cdot \boldsymbol{u}) \boldsymbol{I}. \tag{13}$$

From (13) we can immediately deduce that in the standard SRT-LBM model μ_b is equal to μ in 2D or rather amounts to $\mu_b = 2/3\mu$ in 3D. In other words, shear and deviatoric normal stresses (acoustic propagation) dissipate at a similar rate. A von Neumann stability analysis has revealed that this setting is prone to instabilities due to a phenomenon called *eigenvalue-collision* [39] that is the entanglement of the shear mode and the two acoustics modes, which can trigger instabilities. Remedy may therefore be provided by artificially increasing the bulk viscosity so that the acoustic modes are damped preventing a collision with the shear mode. Strictly speaking, the bulk viscosity does not relate to a specific collision frequency in (12) given $\boldsymbol{m}^{\text{neq}}$, since it acts on the trace of the strain rate tensor. The control of the bulk viscosity through the collision operator can nonetheless be achieved

in two different ways: one possibility is to change the fourth and the fifth element of the non equilibrium moment vector $\boldsymbol{m}^{\text{neq}}$ (cf. (10)) to $m_4^{\text{neq}} = \Pi_{xx}^{\text{neq}} + \Pi_{yy}^{\text{neq}}$ and $m_5^{\text{neq}} = \Pi_{xx}^{\text{neq}} - \Pi_{yy}^{\text{neq}}$ and subsequently reduce ω_4 [40]. Alternatively, the collision matrix can be modified such that

$$\boldsymbol{\Lambda}_{\text{MRT}} \cdot \boldsymbol{m}^{\text{neq}} = \begin{pmatrix} \ddots & & & \\ & \frac{\omega_b+\omega_\nu}{2} & \frac{\omega_b-\omega_\nu}{2} & \\ & \frac{\omega_b-\omega_\nu}{2} & \frac{\omega_b+\omega_\nu}{2} & \\ & & & \ddots \end{pmatrix} \cdot \begin{pmatrix} \vdots \\ \Pi_{xx}^{\text{neq}} \\ \Pi_{yy}^{\text{neq}} \\ \vdots \end{pmatrix},$$

with ω_ν and ω_b being the relaxation frequencies related to the kinematic and bulk viscosity, respectively [24, p. 418]. Reducing the latter will enhance the stability of the scheme.

An MRT model also allows to correct the violation of Galilean invariance. The standard discretization of velocity space (e.g. D2Q9) introduces an error into the higher-order (≥ 3) equilibrium moments, which can be quantified as a $\mathcal{O}(\text{Ma}^3)$-error at dominant order. Due to the previously mentioned recursive relation between the non-equilibrium moments of order n and the equilibrium moments of order $n+1$, the cubic defect (Ma^3-error) can also be found in the non-equilibrium moments of order $n \geq 2$. In other words, it is contained in the strain rate tensor of the momentum equation. The correction of the diagonal and off-diagonal elements of this tensor is achieved in a different manner. For the latter is sufficient to use a partial third-order extension of the Maxwellian (indicated by $p3$) of the form

$$f_\alpha^{eq,p3} = \rho\omega_\alpha \left(1 + \frac{\boldsymbol{c}_\alpha \cdot \boldsymbol{u}}{c_0^2} + \frac{(\boldsymbol{c}_\alpha \cdot \boldsymbol{u})^2}{2c_0^4} - \frac{|u|^2}{2c_0^2} + \frac{\boldsymbol{c}_\alpha \cdot \boldsymbol{u}}{6c_0^4} \left(\frac{(\boldsymbol{c}_\alpha \cdot \boldsymbol{u})^2}{c_0^2} - 3|u|^2 \right) - (\Psi_{x,\alpha} + \Psi_{y,\alpha}) \right), \tag{14}$$

with

$$\Psi_{x,\alpha} = \frac{c_{x,\alpha}}{6c_0^4} \left(\frac{c_{x,\alpha}^2 u_x^3}{c_0^2} - u_x^3 \right), \quad \text{and} \quad \Psi_{y,\alpha} = \frac{c_{y,\alpha}}{6c_0^4} \left(\frac{c_{y,\alpha}^2 u_y^3}{c_0^2} - u_y^3 \right).$$

The error in the diagonal elements, however, cannot be restored by means of expanding the Maxwellian. On a standard lattice a Chapman–Enskog expansion at second-order in the Knudsen number yields

$$\Pi_{ii}^{\text{neq}} = -2\tau\rho c_0^2 \frac{\partial u_i}{\partial x_i} + \underbrace{\tau \frac{\partial \rho u_i^3}{\partial x_i}}_{\text{Ma}^3\text{-error}}. \tag{15}$$

We see that the first term on the right hand side represents the diagonal elements of the strain rate tensor $2\mu\boldsymbol{S} = \mu\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right)$ with $\mu = \tau\rho c_0^2$. The second term is an error term, which is the cubic defect due to the discretization of velocity space. Using the chain rule and neglecting the terms of order $\mathcal{O}(\text{Ma}^5)$ it can be shown that

$$\Pi_{ii}^{\text{neq}} = -2\mu \left(1 - \frac{3u_i^2}{2c_0^2} \right) \frac{\partial u_i}{\partial x_i} \tag{16}$$

[41, p. 47]. The effective viscosity

$$\mu_{\text{eff}} = \mu \left(1 - \frac{3}{2}\text{Ma}^2 \right) \tag{17}$$

thus decreases with increasing Mach number, which explains the limitation of standard LBM to the weakly compressible regime. It should be noted that the modified viscosity only concerns the diagonal elements of the strain rate tensor. It has thus the character of a bulk viscosity. In contrast to its physical counterpart it is, however, space and time variant as it depends on the velocity squared in x and y-direction. A proper correction of the cubic defect therefor requires two independent relaxation times. More precisely

$$\tau_{ii}^{\text{c}}(\boldsymbol{x}, t) = \tau_s \left[1 - \frac{3u_i^2(\boldsymbol{x}, t)}{2c_0^2} \right]^{-1} + 0.5, \tag{18}$$

[23], where the superscript c indicates a correction in order to recover the bulk viscosity of an isothermal Navier–Stokes model.

We conclude that for MRT-LBM on a standard D2Q9 lattice, there are up to five relaxation frequencies that may be modified, *i.e.* those relaxing Π_{xx}^{neq}, Π_{yy}^{neq}, and the three ghost modes. Moreover, the relaxation frequencies ω_1, ω_x and ω_y can be neglected due to the solvability condition ((11)). We are therefore left with a single relaxation frequency, which is bound by physical constraints and that is the one relaxing Π_{xy}, *i.e.* the shear stress.

2.5. Isothermal finite-volume Navier–Stokes model

This section elaborates the different steps of discretizing the isothermal Navier–Stokes equation in space and time. This Navier–Stokes model is composed of the conservation equations for mass and momentum, *i.e.*

$$\frac{\partial \boldsymbol{m}}{\partial t} + \nabla \cdot \boldsymbol{m}_{\mathcal{F}} = 0 \tag{19}$$

with

$$\boldsymbol{m} = \begin{pmatrix} \rho \\ \rho\boldsymbol{u} \end{pmatrix} \quad \text{and} \quad \boldsymbol{m}_{\mathcal{F}} = \begin{pmatrix} \rho\boldsymbol{u} \\ \rho\boldsymbol{u}\boldsymbol{u} + p\boldsymbol{I} - 2\mu\boldsymbol{S} \end{pmatrix}, \tag{20}$$

where the subscript $\mathcal{F}$ indicates the respective fluxes of the moments in $\boldsymbol{m}$. It should be noted that in this model $\mu_b = \frac{2}{3}\mu$ as for example shown in [42]. In contrast to the standard lattice Boltzmann method, the space and time derivative of the NSE are treated separately, owing to the non-linearity of the advection term.

Comparative studies examining different Navier–Stokes methods with respect to the numerical properties of LBM indicate that the dispersion of standard LBM is comparable to a three-stage Runge–Kutta in time and a third-order approximation in space [43]. In order to match the numerical dissipation of standard LBM a higher order discretization is required [44]. In this study, we choose the classical four-stage Runge–Kutta scheme in conjunction with a quadratic upwind interpolation scheme for convective kinematics (QUICK) [45], which has proven to be stable at the Courant–Friedrichs–Lewy (CFL) condition imposed by LBM. This bears the advantage that the training data has the same spatio-temporal resolution ($\Delta t_{\text{LBM}} = \Delta t_{\text{NS}}$ for given Δx) as the LBM simulation and no additional data preparation is necessary. The temporal update of mass and momentum can then be summarized as

$$\begin{aligned} \boldsymbol{m}^{(1/4)} &= \boldsymbol{m}^t - \frac{\Delta t}{2}\mathcal{R}(\boldsymbol{m}_{\mathcal{F}}^t) \\ \boldsymbol{m}^{(2/4)} &= \boldsymbol{m}^t - \frac{\Delta t}{2}\mathcal{R}(\boldsymbol{m}_{\mathcal{F}}^{(1/4)}) \\ \boldsymbol{m}^{(3/4)} &= \boldsymbol{m}^t - \Delta t\mathcal{R}(\boldsymbol{m}_{\mathcal{F}}^{(2/4)}) \\ \boldsymbol{m}^{t+\Delta t} &= \boldsymbol{m}^t - \frac{\Delta t}{6}\left[\mathcal{R}(\boldsymbol{m}_{\mathcal{F}}^t) + 2\mathcal{R}(\boldsymbol{m}_{\mathcal{F}}^{(1/4)}) + 2\mathcal{R}(\boldsymbol{m}_{\mathcal{F}}^{(2/4)}) + \mathcal{R}(\boldsymbol{m}_{\mathcal{F}}^{(3/4)}) \right], \end{aligned}$$

where $\mathcal{R}$ is the finite-volume discretization operator. In order to update the density ρ and the momentum ρu_i in x and y-direction, following expression needs to be evaluated

$$\mathcal{R}\begin{pmatrix} \rho u_j \\ \rho u_x u_j + p\delta_{xj} - \mu 2 S_{xj} \\ \rho u_y u_j + p\delta_{yj} - \mu 2 S_{yj} \end{pmatrix}.$$

Using a central difference approximation, the discrete pressure gradient and the Laplacian of the deviatoric stress term are computed as

$$\begin{aligned} \mathcal{R}(p\delta_{ij}) &= \delta_{ix}\frac{c_0^2\rho(\text{E}) - c_0^2\rho(\text{W})}{2\Delta x} + \delta_{iy}\frac{c_0^2\rho(\text{N}) - c_0^2\rho(\text{S})}{2\Delta y} \\ \mathcal{R}(\mu 2 S_{ij}) &= \mu\frac{u_i(\text{E}) + u_i(\text{W}) + u_i(\text{N}) + u_i(\text{S}) - 4u_i(\text{P})}{\Delta x \Delta y}, \end{aligned}$$

where the letters E, W, N, and S indicate neighboring nodes of grid point P $= (x, y)$. In particular, E $= (x + \Delta x, y)$, W $= (x - \Delta x, y)$, N $= (x, y + \Delta y)$, and S $= (x, y - \Delta y)$.

The third-order (QUICK) scheme evaluates the surface fluxes of mass and momentum with respect to the flow direction using a five-point-stencil incorporating additional information from the grid points

EE = $(x+2\Delta x, y)$, WW = $(x-2\Delta x, y)$, NN = $(x, y+2\Delta y)$, and SS = $(x, y-2\Delta y)$. The fluxes in x-direction are then

$$\mathcal{R}(\phi u_x) = u_x(\mathrm{e})\left(\frac{6}{8}\phi(\mathrm{P}) + \frac{3}{8}\phi(\mathrm{E}) - \frac{1}{8}\phi(\mathrm{W})\right) - u_x(\mathrm{w})\left(\frac{6}{8}\rho(\mathrm{W}) + \frac{3}{8}\phi(\mathrm{P}) - \frac{1}{8}\phi(\mathrm{WW})\right)$$

for $u_x > 0$ and

$$\mathcal{R}(\phi u_x) = u_x(\mathrm{e})\left(\frac{6}{8}\phi(\mathrm{E}) + \frac{3}{8}\phi(\mathrm{P}) - \frac{1}{8}\phi(\mathrm{EE})\right) - u_x(\mathrm{w})\left(\frac{6}{8}\phi(\mathrm{P}) + \frac{3}{8}\phi(\mathrm{W}) - \frac{1}{8}\phi(\mathrm{E})\right)$$

for $u_x < 0$ with $\phi = \{\rho, \rho u_x, \rho u_y\}$. The fluxes in y-direction are computed following the same principle. The flux balance over all faces is then obtained as $\mathcal{R}(\phi u_j) = \mathcal{R}(\phi u_x) + \mathcal{R}(\phi u_y)$. In the following the above described discretization will be referred to as RK4-QUICK.

2.6. Method comparison

In order to conclude the presentation of the different numerical schemes we assess their numerical characteristics on a decaying turbulence test case [46] choosing a $(n_x \times n_y) = (128 \times 128)$ periodic domain with Re = 10000 (based on the domain length) and Ma = 0.1. The initial solution is shown in Fig. 1(a). The energy peak at $t = 0$ is located at a wavenumber of 10. The single relaxation time model corresponds to (4), with $\boldsymbol{m} - \boldsymbol{m}^{\mathrm{eq}} = \mathcal{T}_{\mathrm{TP}} \cdot (\boldsymbol{f} - \boldsymbol{f}^{\mathrm{eq},p3})$ and ω defined as in (3). SRT($\sigma = 0.01$) is the same SRT model with the only difference that mass and momentum are filtered using a 7-point stencil ((7)) and a filter coefficient of $\sigma = 0.01$. The hybrid recursive regularized model presented in Section 2.3 uses a blending coefficient σ_{HRR} equal to 0.98 in (9). The multi relaxation time model does a full over-relaxation of the three non-hydrodynamic moments i.e. $\omega_{xxy} = \omega_{xyy} = \omega_{xxyy} = 1$. Moreover, the fourth moment, m_{xx}^{neq}, is replaced by the trace of $\boldsymbol{\Pi}^{\mathrm{neq}}$ and also relaxed with 1. The fifth moment m_{yy}^{neq} is consequently replaced by $\Pi_{xx}^{\mathrm{neq}} - \Pi_{yy}^{\mathrm{neq}}$ and relaxed with the physical relaxation frequency ω.

Fig. 1(b) shows the energy spectrum after 3000 iterations obtained with the different methods. The SRT model features the lowest dissipation, however, we observe an accumulation of energy in the high wavenumber regime. This phenomenon is also be observed in case of HRR despite being the method with the highest dissipation, particularly at high wavenumbers. This behavior may be attributed to the finite difference scheme used in Eq. (9). The modified wavenumber of central finite difference schemes reduces to zero at the highest wavenumber [47]. It is thus possible that those wavenumbers are finally not subject to sufficient physical or numerical dissipation such that aliasing increases energy at the largest wavenumbers. The energy accumulation observed when using SRT is likely related to the discretization of velocity space. It is well-known that aliasing effects occur as continuous moments are wrongly represented in the discrete LBM case due to the limited number of discrete velocities. MRT and filtered SRT show a very similar, intermediate dissipation rate at high wavenumbers. Turbulent structures with a wavenumber of around 10 dissipate more quickly with MRT and HRR compared to SRT, and RK4-QUICK, which could be attributed to the damping of the ghost modes that is absent in the other models. Interestingly, the Navier–Stokes model performs very well, showing only a slightly higher dissipation than SRT without the accumulation of energy near the Nyquist frequency. These findings further motivate the training of the neural collision operator with data from our NS solver, in the anticipation that the favorable numerical characteristics (stable at low dissipation) will be passed on to the neural collision operator.

2.7. Neural LBM

In this paper, neural LBM denotes a variation of the MRT-LBM model, where certain relaxation times are trained using a neural net. As elaborated in Section 2.4 it would be possible to only fix ω_{xy} and let the ML algorithm take care of the remaining relaxation rates. This, however, would require the training of at least 4 different neural nets (Q_{xyy} and Q_{xxy} are symmetric). Moreover, a very similar study exists that has already successfully demonstrated the training of the relaxation frequencies related to the three ghost modes (in 2D) using high resolution LBM data [20]. Since our training data consists of Galilean invariant solutions from an isothermal Navier–Stokes solver, we decide to focus on ω_{xx} and ω_{yy}. This will allow the net to learn the correction of the cubic defect as well as to tune the bulk viscosity to adapt to the numerical dissipation of the RK4-QUICK solver. The ghost modes in this study are simply relaxed with ω, so that the framework for the neural LBM model is the following matrix

$$\boldsymbol{\Lambda}_{\mathrm{NN}} = \mathrm{diag}\left(\ldots, \omega_{xx}^{\mathrm{NN}}, \omega_{yy}^{\mathrm{NN}}, \omega, \omega, \omega, \omega\right), \tag{21}$$

together with the non equilibrium moment vector $\boldsymbol{m}^{\mathrm{neq}}$ defined in (10). The sub-/superscript NN indicates an output of the neural net.

2.8. Machine learning framework

The advent of machine learning has led to the development of countless platforms to kick-start machine-learning augmented projects. One such platform is the *Python* based open-source software *lettuce* (https://github.com/lettucecfd/lettuce), which is a modular LBM framework that can easily be extended and coupled with machine learning infrastructure based on *pytorch*. In previous studies, the framework was used to train relaxation times that are associated with the non-hydrodynamic modes based on a refined LBM reference solution [20, 48]. For this study, we have extended the framework by the previously described FV-NS method to enable the training with data that lacks the Ma^3-error. As a consequence, an interface is required to feed the macroscopic solution into the LBM algorithm. For this purpose a high-order recursive regularization step [49] is applied so that the distribution functions are initialized with

$$f_\alpha = f_\alpha^{\mathrm{eq}}(\rho_{\mathrm{NS}}, \boldsymbol{u}_{\mathrm{NS}}) + f_\alpha^{(1),\mathrm{RR}}(\rho_{\mathrm{NS}}, \boldsymbol{u}_{\mathrm{NS}}), \tag{22}$$

where $f_\alpha^{(1),\mathrm{RR}}$ is a fourth-order recursively regularized (RR) non-equilibrium distribution.

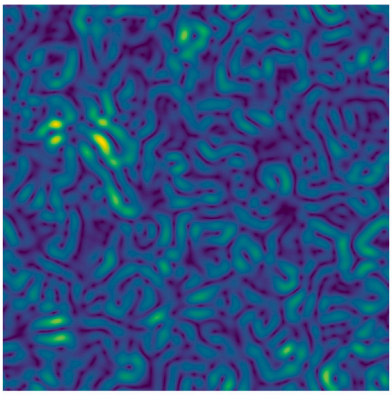
The input layer contains the nine velocity moments corresponding to the transformation matrix $\mathcal{T}_{\mathrm{TP}}$. The hidden layer is composed of 24 neurons with a rectified linear unit (ReLU) activation. Each network's output layer consists of a single neuron, providing an independent mapping function for each relaxation time τ_{xx} and τ_{yy} (Fig. 2). This configuration was adopted from previous studies [20]. The rationale for using separate networks, each with its own single output neuron, stems from (18), which indicates that the relaxation times might significantly differ depending on the flow. In a joint network with multiple output neurons, a dominant relaxation time could disproportionately influence the training of the others. Additionally, we hypothesize that the hidden units represent different combinations of velocity moments (features) that might be different for τ_{xx} and τ_{yy}.

Each net is evaluated in each node of the computational grid. This allows for the relaxation time to vary in space and time, which is expected as the correction of the cubic defect depends on the velocity field. This also implies that each node stores one training example ($\boldsymbol{m}$, τ_{ii}^{NN}) and that the resolution $(n_x \times n_y)$ of the simulation data corresponds to the number of training examples at time t.

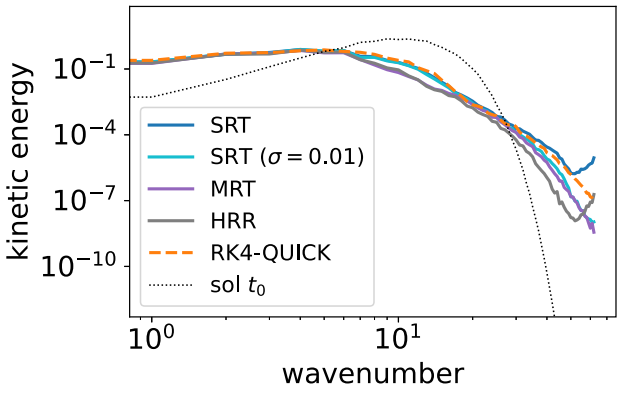
According to (18) the magnitude of the numerical error as a result of the violation of Galilean invariance cannot exceed the physical relaxation time τ_s in the subsonic regime. It is true that in a full overrelaxation the factor is $0.5/\tau_s$, however, this is applied to the trace of $\boldsymbol{\Pi}^{\mathrm{neq}}$ and not separately to Π_{xx}^{neq} and Π_{yy}^{neq}, which is different. Based on this observation, the activation function in the output neuron was chosen such that

$$\tau_{ii}^{\mathrm{NN}} = (1 + \tanh(a))\tau_s + \frac{1}{2}, \tag{23}$$

where a is the weighted sum of the hidden layer output. The neural relaxation time is thus constrained to vary on the interval $0.5 < \tau_{ii}^{\mathrm{NN}} < 0.5 + 2\tau_s$.



(a)



(b)

Fig. 1. Initial solution of the decaying turbulence test case with an energy peak at a wavenumber of 10 and the respective wavenumber spectrum after 3000 iterations.

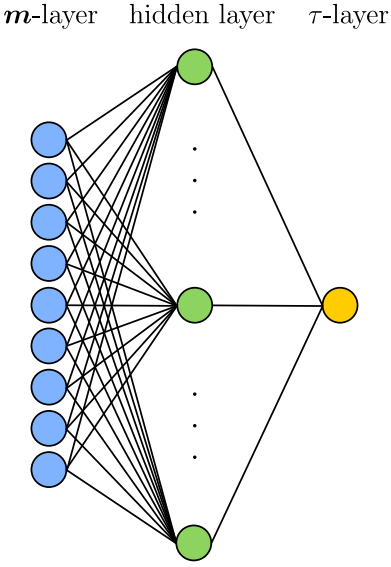


Fig. 2. Neural net.

2.9. Algorithm

In an iterative process, the algorithm seeks the relaxation times τ_{xx}^{NN} and τ_{yy}^{NN}, such that the difference between the two solvers (LBM and NS) is minimized with respect to some loss function $\mathscr{L}$. In this study, we simply chose the average of the conserved quantities as a measure for the performance of the model, *i.e.*

$$\mathscr{L}(t) = \frac{1}{2}\sum_{i}^{n_x \times n_y} \mathrm{MSE}\left(\rho_{\mathrm{LBM}}(\boldsymbol{x}_i, t), \rho_{\mathrm{NS}}\right) + \frac{1}{2}\sum_{i}^{n_x \times n_y} \mathrm{MSE}\left(|\boldsymbol{u}|_{\mathrm{LBM}}(\boldsymbol{x}_i, t), |\boldsymbol{u}|_{\mathrm{NS}}\right) \tag{24}$$

where MSE denotes the mean-squared error (see Fig. 2).

The training of the neural net begins with the generation of *true* simulation data for $n_{\Delta t}$ iterations using the RK4-QUICK scheme. In a next step, we define a time interval ΔT, which subdivides the simulation data into $n_{\Delta T}$ fragments. Then we choose a random number r with $r \in \mathbb{N}_0$ and $r \in [0, n_{\Delta T}[$ (without replacement) and initialize f_α at time $t = r\Delta T$ using RR (Eq. (22)). After initializing τ_{ii}^{NN} via forward propagation, the neural MRT-LBM algorithm is run for ΔT iterations. Once the simulation is completed, the result is compared to the NS solution at time $t = (r+1)\Delta T$ using (24). Finally, a backpropagation step updates the parameters of the net at learning rate β. This procedure is repeated $n_{\Delta T} - 1$ times, which completes a single epoch. An illustration of the single steps of an epoch is shown in Fig. 3 for the barotropic vortex test case [50] on a 64×64 grid. ΔT was chosen such that the vortex passes 0.75 times the periodic domain (from right to left) and $n_{\Delta T} = 5$. For the sake of better comprehensibility, r is not chosen randomly here but at increasing order.

Table 1
Values of the kinematic viscosity and the relaxation time for different grid resolutions (in lattice units).

N	64	96	144
ν	0.00064	0.00096	0.00144
τ	0.50192	0.50288	0.50432

3. Validation

As mentioned in Section 2, a single snapshot of simulation data already provides N^2 training examples, where N is the non-dimensional length of the quadratic, regular Cartesian grid. It is worth mentioning that even though the neural net is only fed with single point data it is nevertheless trained with gradients of ρ and $\boldsymbol{u}$, which are contained in the higher order velocity moments of f_α. In this study we limit the training to a single test case, the aforementioned barotropic vortex [50].

The following validation will include two versions of this case for training and testing, in the following referred to as *low Mach number case* and *high Mach number case*. All variables will be given in lattice units $\Delta x = \Delta t = 1$.

The high Mach number case was chosen such that the SRT-LBM algorithm becomes unstable, *i.e.* Ma > 0.3. In addition to the cubic Mach error, instabilities may also be triggered by the undamped ghost modes. In order to be consistent with regard to this additional source of instability, the two Mach number cases use the same relaxation time τ (in lattice units). In the low Mach number case the bulk velocity in x-direction is set to $u_{\mathrm{low}} = 0.1$ (in lattice units), so that $\mathrm{Ma}_{\mathrm{low}} = 0.173$. The velocity fields are shown in Fig. 4. The Reynolds number in this case is set to $\mathrm{Re}_{\mathrm{low}} = 10\,000$. In the high Mach number case, the bulk velocity is increased by a factor of 2.5, leading to a Mach number of $\mathrm{Ma}_{\mathrm{high}} = 0.433$. The Reynolds number $\mathrm{Re}_{\mathrm{high}}$ is increased by the same factor in order to maintain the same relaxation time. The vortex strength is set to 0.1 in both cases, leading to a maximum Mach number of 0.233 and 0.493, respectively.

The cases were run on a periodic square domain with three different resolutions that differ by a factor of 1.5. The respective collision times are provided in Table 1. If not stated otherwise, single resolution data is shown for the coarse grid ($N = 64$).

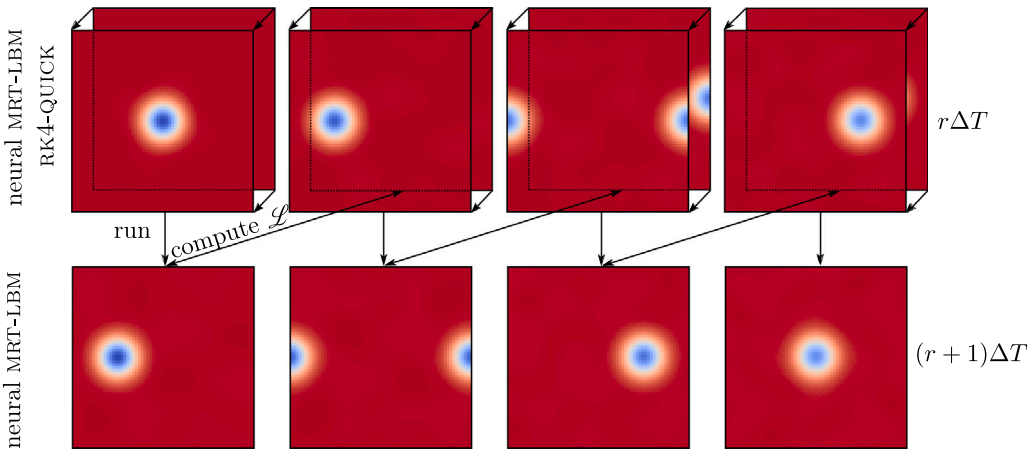


Fig. 3. Diagram of a single training epoch: the upper row shows the NS data in the background plane, which is used to initialize the LB simulation that is shown in the foreground. The lower row shows the LBM solution advanced by $\Delta T = 480$ iterations, resulting in 0.75 domain passages of the vortex in positive x - direction (periodic domain). In this example the vortex therefore passes the domain three times with $n_{\Delta T} = 5$. (Note that the last snapshot of the NS simulation data that is only required for the computation of $\mathscr{L}$ is omitted here.).

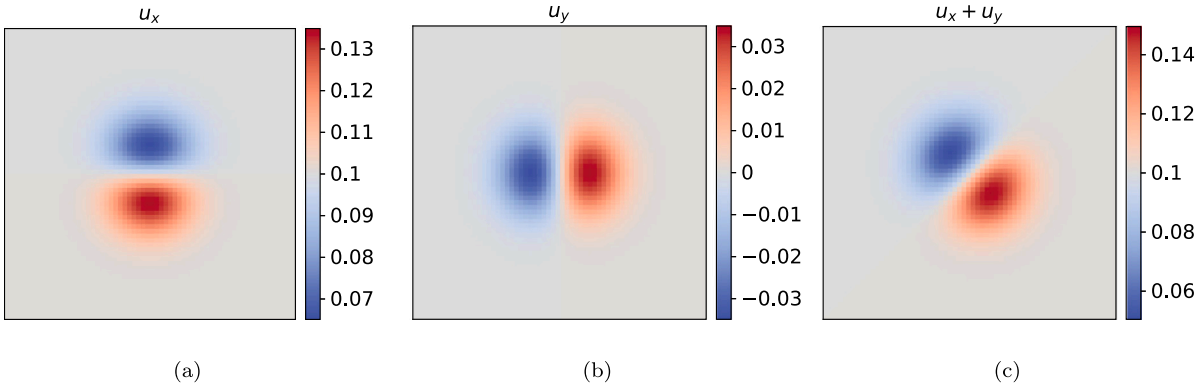


Fig. 4. Initial velocity fields of the barotropic vortex at low Mach number.

3.1. Low mach number case

The neural collision operator is trained with the algorithm described in Section 2.9 using three different learning rates. In fact, the training corresponds to the diagram in Fig. 3 with the only difference that the order of the training fragments within a single epoch is chosen randomly. A training interval ΔT corresponding to 0.75 domain passages of the vortex amounts to 480, 720 and 1080 iterations on the coarse, medium, and fine grid, respectively. The learning curves for the coarse grid case are shown in Fig. 5 and do not vary significantly among different resolutions.

The loss is normalized with respect to the difference between the RK4-QUICK scheme and the SRT-LBM model. We recall that the loss shown in Fig. 5 is the average loss over one epoch. By the end of the first epoch, the relative loss can therefore already be inferior to 1 as seen in case of $\beta = 0.0025$ and $\beta = 0.005$. After 25 epochs all learning curves have converged towards a relative loss of about 85%. In other words, the neural collision times τ_{xx}^{NN} and τ_{yy}^{NN} have brought the solutions of LBM and RK4-QUICK 15% closer to each other, with respect to the loss function defined in (24). The lower rates show a smooth convex convergence, while the higher rate converges the fastest but does so in a more irregular fashion, which indicates that this learning rate could be too high. A default learning rate of $\beta = 0.005$ is therefore used in the following.

In order to get a first intuition on how the neural collision operator behaves on the barotropic vortex test case, we do a qualitative

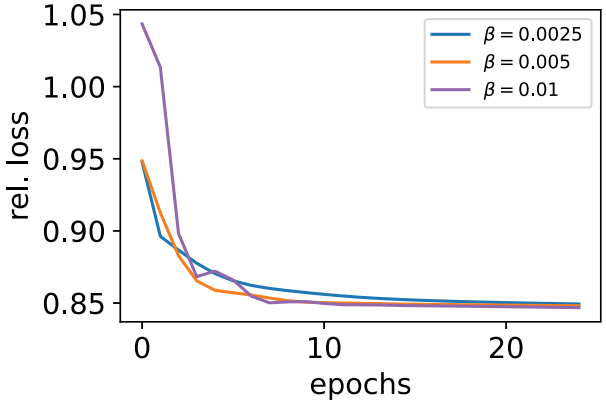


Fig. 5. Evolution of different learning rates over 25 epochs using 64 × 64 training examples.

cross-comparison of the density and the velocity fields. The results represented in Fig. 6 show the field differences after 5 domain passages between every pair of the tested schemes: RK4-QUICK (NS), SRT-LBM (SRT), and neural LBM (NN). The contours of the density difference $\Delta\rho_{\mathrm{NS\text{-}SRT}} = \rho_{\mathrm{NS}} - \rho_{\mathrm{SRT}}$ indicate that LBM is less dissipative than NS. Moreover, around the vortex we recognize a formation of patches with $\Delta\rho_{\mathrm{NS\text{-}SRT}} \neq 0$ that resembles a spiraling pinwheel.

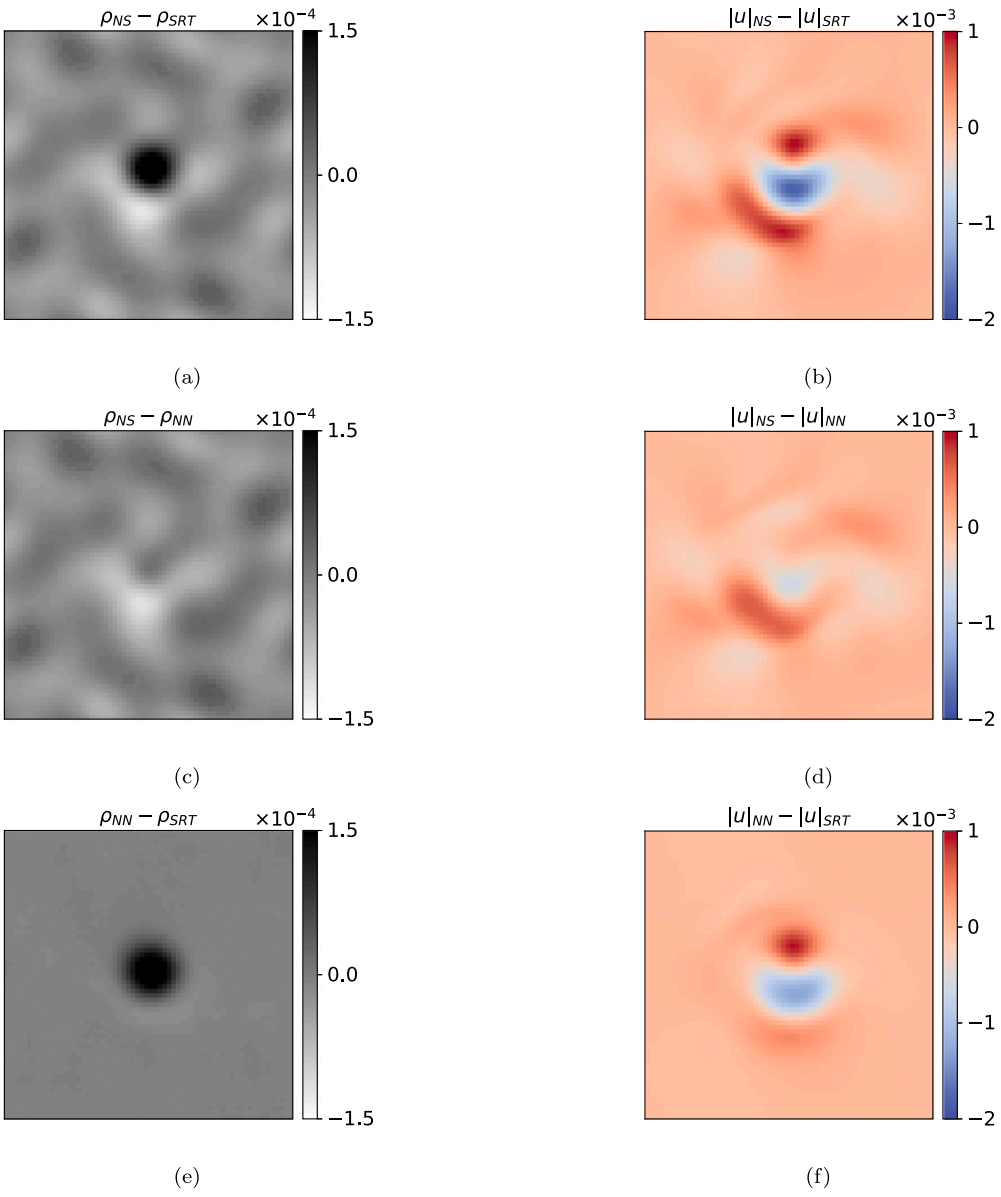


Fig. 6. Differences between the density and velocity fields of the different schemes after 5 domain passages.

When comparing NS with NN, we observe that the neural collision adds to the numerical dissipation, so that the difference at the vortex center disappears. The spiraling pattern in the vicinity of the vortex, however, remains unchanged. A comparison of the two LBM schemes in Fig. 6(e) confirms that NN is more dissipative than SRT-LBM. The density difference $\Delta\rho_{\text{NN-SRT}}$ around the vortex is zero. We suspect that the reason for the pinwheel-like structures lies in the different discretization of the advective terms (FV versus lattice) and can therefore not be corrected by the neural collision operator.

The interpretation of the velocity difference contours $\Delta|u|$ in less straightforward. The vortex rotates counter-clockwise, which means that in case of a translation in positive x-direction, rotational and bulk velocity are opposed above the vortex center, while below, the combined velocities excess the mean flow velocity (see Fig. 4(a)). The neural collision operator has therefore a decelerating effect on the vortex rotation as evidenced in Fig. 6(f). This is in agreement with a higher dissipation observed in Fig. 6(e). The difference $\Delta|u|_{\text{NS}-\text{SRT}}$ is more complex but it is again save to say that the macroscopic method is more dissipative than the particle based approach. In the periphery of the vortex we observe a similar spiral-like structure (as seen in the $\Delta\rho$ contours) that can also be seen when comparing NN with NS in Fig. 6(d). Again we attribute this to a conceptual difference in the discretization of the variable transport.

For a more quantitative assessment of the neural collision operator the different numerical methods are compared with respect to the L_2-norm after one (640 iterations) and five domain passages (3200 iterations) along the horizontal centerline, *i.e.*

$$\varepsilon_\phi = \sqrt{\frac{\sum_{x=1}^{N} \left(\phi(x, N/2, t) - \phi(x, N/2, t=0)\right)^2}{N}}, \tag{25}$$

where ϕ is a dummy variable. The L_2-error is computed with respect to the initial solution since there is no trivial solution for $t \neq 0$. It is therefore composed of the numerical error and the physical decay of the vortex that is proportional to $e^{-\nu k^2 t}$. The results should hence only be used for a relative estimate. The -2-slope is nonetheless shown in the bottom row of Fig. 7.

On the coarse grid (64×64), the reference SRT-LBM method shows the smallest error in all four graphs. The neural LBM algorithm (NN) shows a very similar error than the scheme it was trained with (RK4-QUICK) apart from ε_ρ after one domain passage, where the error

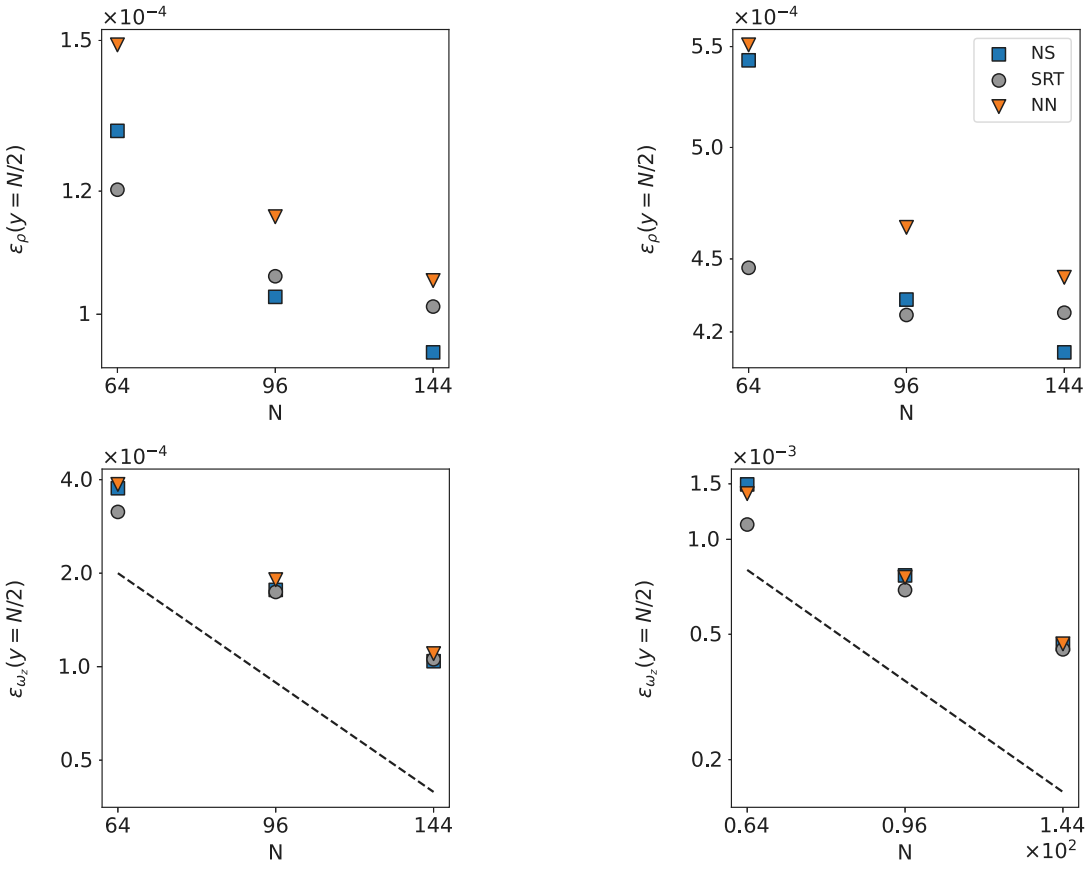


Fig. 7. L_2-error of the different numerical schemes after 1 (left column) and 5 (right column) domain passages with respect to the initial solution. The neural relaxation times were trained with $\beta = 0.005$ based on data from the NS scheme with QUICK approximation in space. The dashed line in the bottom row figures indicates the −2-slope.

is notably higher. This is nevertheless a possible outcome, since the neural collision operator was trained to reduce the error or rather loss on the entire domain (cf. (24)) and not only over the center line at $y = N/2$. Using a coarse mesh (64 × 64), NS and LBM show a very similar error in the density, while the solution obtained with the NN scheme has a higher error. On the fine mesh (144 × 144), the NS scheme shows the smallest error ε_ρ after one and five domain passages. The LBM algorithm with neural collision operator shows a slightly higher error ε_ρ than the base algorithm with single relaxation time. In case of ε_{ω_z}, where ω_z denotes the vorticity in the x–y plane, the L_2 errors differ only marginally among the different methods.

A second part of the quantitative validation investigates in how far the neural collision operator generalizes to lower or rather higher resolution data (coarse-graining and fine-graining). Concretely, the neural nets that were trained with fine resolution data are tested on the coarse and medium mesh and vice versa. We recall that the neural nets learn the mapping from the nine velocity moments $\boldsymbol{m}$ to the relaxation times τ_{xx}^{NN} and τ_{yy}^{NN}. A transfer to a different discretization step is therefore straightforward, as the nets do not contain information about the grid size (only the number of training examples varies). For each resolution in Fig. 8, we show the performance of the neural collision operation that was trained on a coarse ($N = 64$), medium ($N = 96$) and fine mesh ($N = 144$). In addition, results of the SRT-LBM (circle) and the RK4-QUICK (square) methods are shown once again for better comparability. The effect on ε_ρ and ε_{ω_z} is quite different. The vorticity error depends highly on the resolution of the training data and not so much on the resolution of the testing data. The error on the coarse mesh using an NN that was trained with high resolution data is therefore considerably smaller compared to the reference SRT-LBM scheme. Interestingly, this error reduction can only be traced to the collision times τ_{xx}^{NN} and τ_{yy}^{NN} – in other words a manipulation of the normal stresses of the strain rate tensor. The density error on the other hand depends more on the resolution used during training. In case of a *coarse-graining* the neural collision operator outperforms the standard method only at the first checkpoint. After 5 domain passages, the error lies between the result of SRT-LBM and the RK4-QUICK method. In terms of the density error, the NS scheme becomes better with increasing resolution up to the point where it outperforms the SRT-LBM model on a 144 × 144 grid. At such fine resolution, the NN net will try to reduce the dissipation by lowering the bulk viscosity, which is authorized by the output activation function in (23). This explains the better performance of NN versus SRT-LBM when trained on the fine mesh and applied to the coarse. At the same time we expect a deterioration of stability. *Fine-graining* leads to an expectedly worse performance and is only shown for the sake of completeness.

3.2. High mach number case

After a first evaluation of the neural collision operator, the Mach number and the Reynolds number are now increased by a factor of 2.5 to create a flow that lies outside the stability envelop of SRT-LBM. We recall that by increasing the Reynolds number and the Mach number simultaneously, the relaxation time (in lattice units) remains unaffected. A single domain passage now only requires 256, 384 and 576 iterations on the coarse, medium, and fine grid, respectively. The training intervals ΔT of a single epoch are chosen such that the traveled distance remains unchanged compared to the low Mach number case ($0.75N$). Fig. 9 shows the density and vorticity profiles of the vortex at $y = N/2$ after 3 (for a better distinctiveness of the results) and 5 domain passages on a 64 × 64 grid. In addition to the three schemes that were previously examined, we also present results for an MRT model that contains the correction of the cubic defect, denoted MRT^c [23]. In agreement with the L_2-error in Fig. 7 the SRT-LBM model shows the lowest

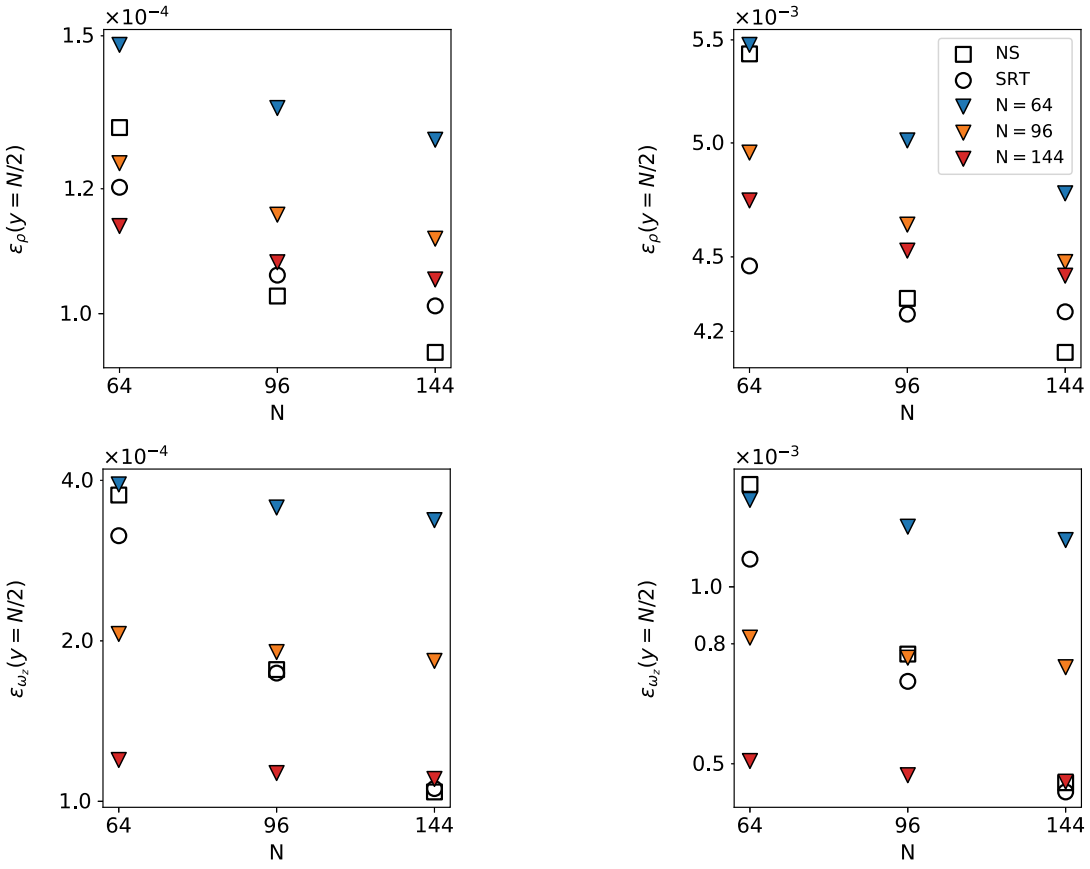


Fig. 8. L_2-error of the different numerical schemes after 1 (left column) and 5 (right column) domain passages with respect to the solution at $t = 0$. The neural collision operator was trained with differently resolved data (color coded triangles) and then tested on different mesh resolutions (abscissa).

dissipation. After three domain passages the results of SRT and MRT^c are superimposed. The dissipative behavior of the RK4-QUICK scheme and neural LBM are very similar with a slightly lower dissipation for the latter. After 5 domain passages the standard LBM scheme becomes unstable. This is also true for the mrt^c, although the instability appears less developed. The NS model and the neural LBM model show stable results.

The neural collision operator thus yields more dissipated but stable results compared to the standard approach with SRT. Now the question arises of how the neural LBM compares to other stability enhancing schemes presented in Section 2. For that reason we have repeated the simulation of the high Mach number test case with the HRR scheme ($\sigma_{\text{HRR}} = 0.98$) and the MRT scheme, where the ghost modes and the trace of the strain rate tensor are fully over-relaxed ($\omega = 1$). For the sake of consistency, the ghost modes in the NN collision model are exceptionally relaxed with $\omega = 1$ as well. The density profile and the turbulent kinetic energy spectrum after 10 domain passages are presented in Fig. 10. The increase in domain passage allows for a better distinction of the results. We find that all schemes remain stable and mainly differ in terms of dissipation. Concerning the density profile across the vortex (Fig. 10(a)), the HRR scheme shows the lowest dissipation (*i.e.* highest density drop), while the density fields obtained with MRT and NN are very similar with a slightly lower dissipation in case of the latter.

At first glance this is in contradiction to the results of the decaying turbulence in Fig. 1 and the kinetic energy spectrum shown in Fig. 10(b). However, the difference in the spectra only becomes visible in the high wavenumber regime, while the barotropic vortex case assembles most energy at low wavenumbers. At these wavenumbers an increased bulk viscosity (MRT) seems to have a bigger impact on the numerical dissipation than the hyperviscosity that is added by the partial finite difference approximation of the strain rate tensor in HRR. This difference nevertheless seems to be too small to be visible in Fig. 10(b).

We conclude the validation with some contour maps of the normalized relaxation times $\widetilde{\tau}_{xx}(\boldsymbol{x}, t) = (\tau_{xx}(\boldsymbol{x}, t) - 0.5)/\tau_s$ and $\widetilde{\tau}_{yy}(\boldsymbol{x}, t) = (\tau_{yy}(\boldsymbol{x}, t) - 0.5)/\tau_s$ after 5 domain passages. The top row in Fig. 11 shows the correction proposed in [23] and given in (18), while the bottom row shows the neural collision times that were informed by the results of an isothermal Navier–Stokes solver. According to (18), the analytical correction of the Mach error depends on the velocity field. In particular $\widetilde{\tau}^c_{xx}$ scales with $(1 - u_x^2)^{-1}$ and $\widetilde{\tau}^c_{yy}$ is proportional to $(1 - u_y^2)^{-1}$. Since the vortex travels in the x-direction, $\widetilde{\tau}^c_{xx}$ is greater than τ_s everywhere in the domain. In accordance with the u_x-velocity field shown in Fig. 4(a), it is the highest below the vortex, where the rotational and translational velocities superimpose and it is lowest above the vortex, where these velocities are opposed.

$\widetilde{\tau}^c_{yy}$ on the other hand equals the single relaxation time τ_s everywhere in the domain, apart from two semicircular regions conforming to the square of the zero-centered velocity field u_y (*cf.* Fig. 4(b)). However, the deviation from τ_s remains below 1% due to the comparatively low rotational velocity of the vortex ($\text{Ma}_{\text{max}} = 0.06$).

The contours of the normalized neural collision time $\widetilde{\tau}^{\text{NN}}_{xx}$ resemble more the velocity field $u = u_x + u_y$ (or a power of it) than u_x^2 (*cf.* Fig. 4(c)). Moreover, the deviation from τ_s is more pronounced. While the value of $\widetilde{\tau}^c_{xx}$ is about 26% to 57% higher than the reference value, $\widetilde{\tau}^{\text{NN}}_{xx}$ varies on the entire positive interval that is authorized by the *tanh* activation function, *i.e.* $[\tau_s; 2\tau_s]$ (cf. (23)).

The neural collision time relaxing the normal stresses in y-direction greatly differs from $\widetilde{\tau}^c_{yy}$. Firstly, we notice that in the periphery of the vortex the value is about 25% higher than the shear relaxation time τ_s. The contours show two patches with an increased value of about

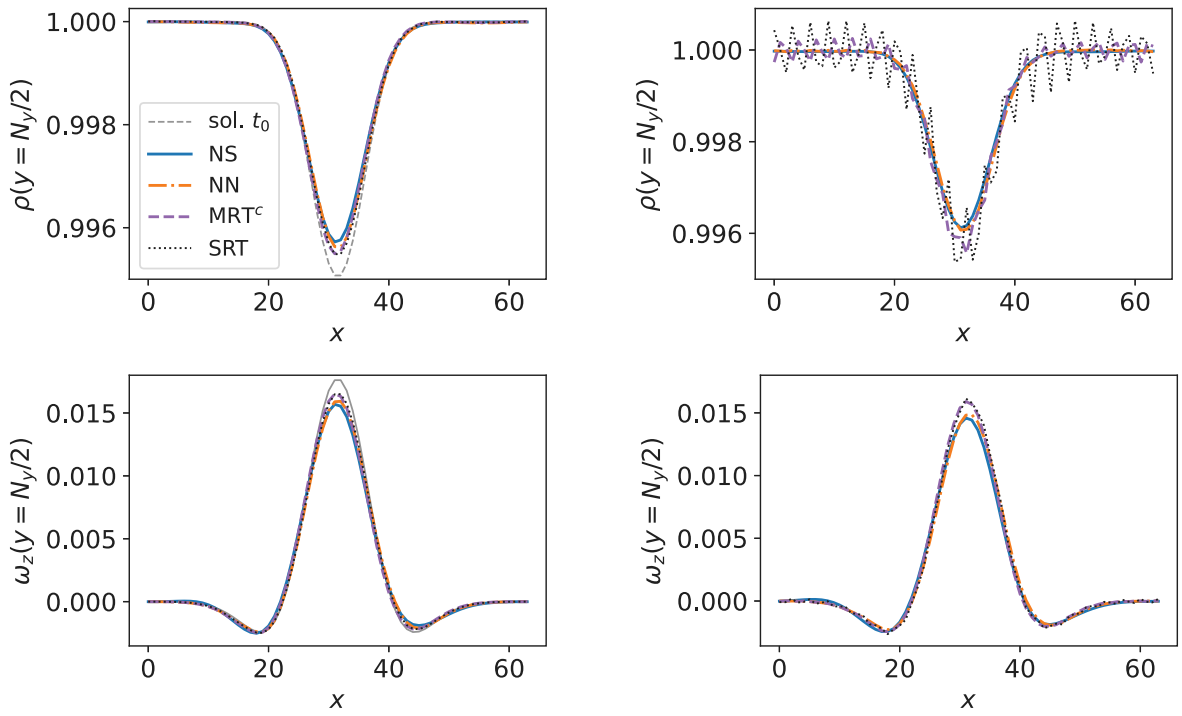
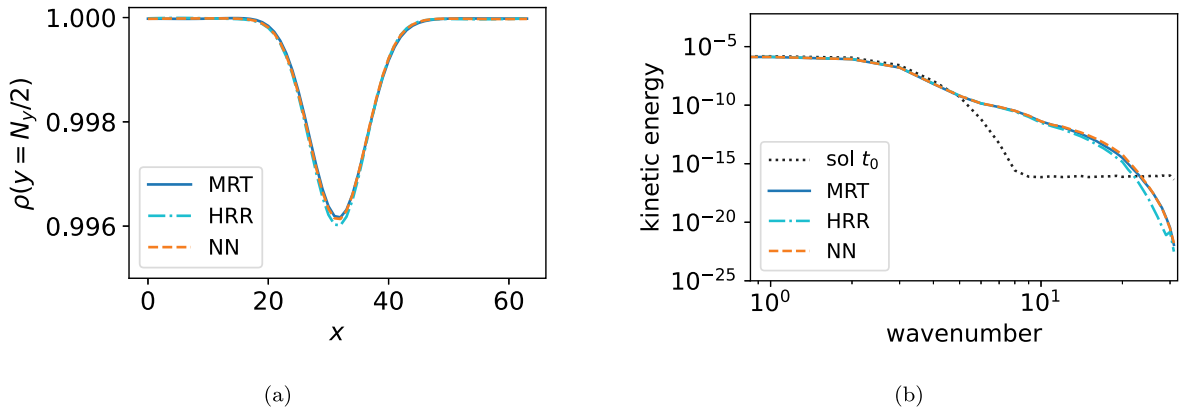


Fig. 9. Density and vorticity profiles after 3 (left) and 5 (right) domain passages. The neural collision operator was trained with the NS data using the QUICK scheme. The learning rate was set to $\beta = 0.005$.



(a) (b)

Fig. 10. Density profile and energy spectrum after 10 domain passages.

$1.45\tau_s$, which are located below and to the right of the vortex center. In addition, there is a region above the vortex center where $\widetilde{\tau}_{yy}^{\mathrm{NN}}$ is lower compared to the value in the far field. This is not the case for $\widetilde{\tau}_{yy}^{c}$.

Similar to the corrected relaxation times, the neural relaxation times are influenced by the velocity field. The specific correction nevertheless was not transferred to the neural LBM scheme. This can be explained by the fact that the two schemes (LBM and RK4-QUICK) also differ in terms of numerical dissipation and that this difference is more important. The neural values are maximized within the limits of the activation function in order to get closer to the dissipation rate of the NS solver.

4. Conclusion

This work is part of a series of (independent) studies, where neural networks are employed either as surrogate collision models or as part of the MRT collision operator in LBM. In a previous study [20], it was demonstrated that a neural network can learn a mapping between the local velocity moments and the non-hydrodynamic relaxation times of an MRT model, which improves simulation results. In this study we adopt a similar approach by learning a mapping for the relaxation times associated with the bulk viscosity. Another novelty here is that the training data stems from a Galilean invariant, isothermal Navier–Stokes solver. Demonstrated through the example of a 2D barotropic vortex, several interesting conclusions can be drawn. The neural relaxation times $\tau_{xx}^{\mathrm{NN}}(\boldsymbol{x}, t)$ and $\tau_{yy}^{\mathrm{NN}}(\boldsymbol{x}, t)$, which vary locally, depend mainly on the velocity field and show non-linear behavior. A learning of the specific correction of the cubic defect is not observed as other numerical differences (dissipation rate) dominate. In terms of stability, the neural collision operator performs better than an MRT model that only corrects the cubic defect. Furthermore, on the selected test case the novel collision operator performs slightly better in terms of accuracy, when compared to a naive increase of bulk viscosity, which is often done to stabilize simulations. The use of a neural network in order to determine certain relaxation times thus constitutes an interesting alternative to conventional stability enhancing schemes. It is shown that the neural collision operator allows to endow LBM with certain numerical properties that are inherent to the donor scheme (NS). One could therefore imagine to use LBM as an efficient base algorithm that

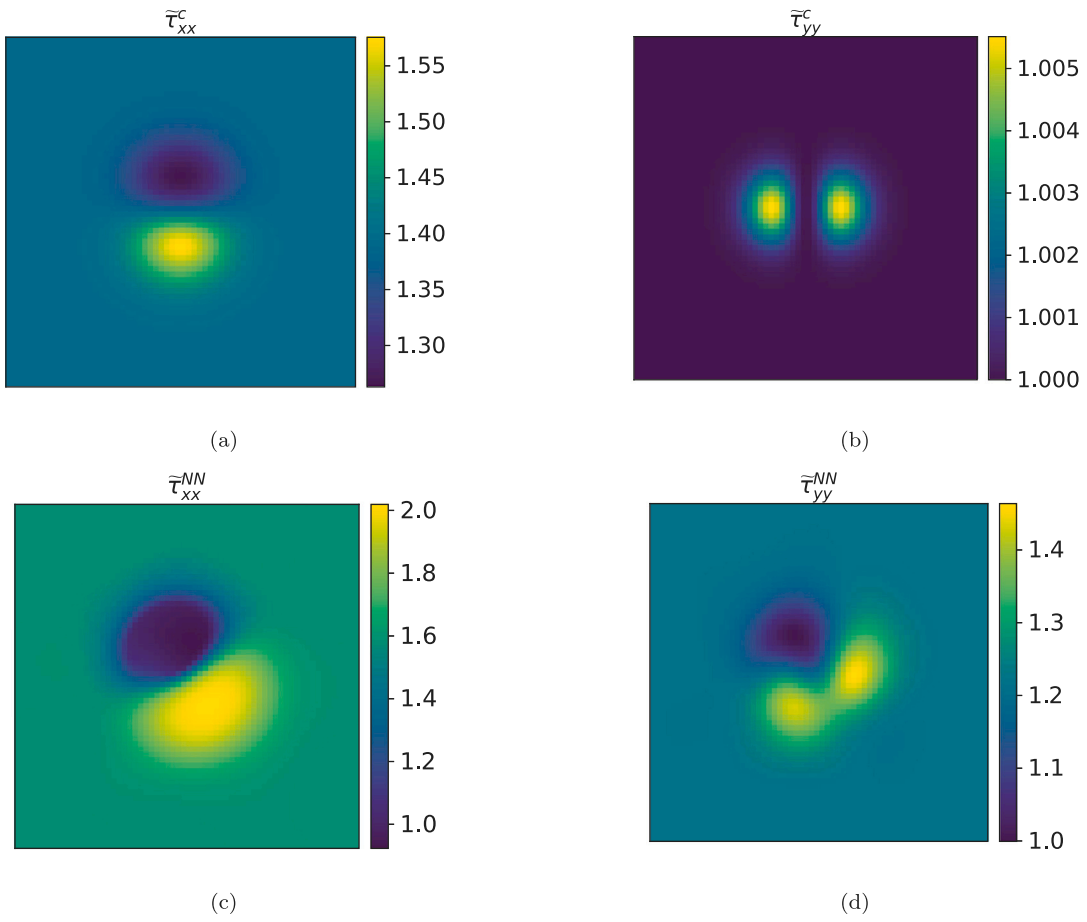


Fig. 11. Contour maps of the normalized relaxation times $\widetilde{\tau}_{xx}$ and $\widetilde{\tau}_{yy}$ after 960 iterations (5 domain passages) of the barotropic vortex case at Ma = 0.433. The upper row shows the relaxation field that is necessary to correct the cubic defect of standard LBM. The bottom row are the neural collision times that where trained to match the solution of an isothermal Navier–Stokes solver. Both, the training and the testing were carried out on a 64 × 64 grid at Re = 25 000.

can be customized for specific flows. Another area of application are flows with conflicting numerical requirements. A current workaround are sensors that allow to locally change the numerical behavior of a scheme based on a predefined threshold. A properly trained neural collision operator would automatically take care of this. In LES simulations the subgrid-scale turbulence is accounted for by an artificial shear viscosity. In this case one could also use a neural net to compute $\tau^{\mathrm{NN}}_{xy}(\boldsymbol{x}, t)$. Since the input layer contains information about the velocity gradients, such a neural turbulence model would automatically adapt to regions of increased shear, *i.e.* in proximity to a wall. Many more ideas exist, including the use of a convolution layer paired with a central moment collision operator to build more efficient, equivariant neural nets. This will be tested in future studies.

CRediT authorship contribution statement

Jan Tobias Horstmann: Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Mario Christopher Bedrunka:** Software, Writing – review & editing. **Holger Foysi:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] 2023. https://www.dlr.de/en/research-and-transfer/research-infrastructure/hpc-cluster/caro. [Accessed 12 December 2023].
[2] Malik MR, Bushnell DM. Role of computational fluid dynamics and wind tunnels in aeronautics R&D. 2012, p. 1–62, NASA/TP-2012-217602 (September).
[3] Spalart PR, Venkatakrishnan V. On the role and challenges of CFD in the aerospace industry. Aeronaut J 2016;120(1223):209–32.
[4] Choi H, Moin P. Grid-point requirements for large eddy simulation: Chapman's estimates revisited. Phys Fluids 2012;24(1):1–6.
[5] Probst A, Knopp T, Grabe C, Jägersküpper J. HPC requirements of high-fidelity flow simulations for aerodynamic applications. In: Euro-par 2019: Parallel processing workshops, vol. 11997. 2020, p. 375–87.
[6] Slotnick J, Khodadoust A, Alonso J, Darmofal D. CFD vision 2030 study: A path to revolutionary computational aerosciences. Tech. rep. NASA/CR-2014-218178 (March), 2014.
[7] Manoha E, Caruelle B. Summary of the LAGOON Solutions from the Benchmark problems for Airframe Noise Computations-III Workshop. AIAA J 2015. Proceedings of the 21st AIAA/CEAS Aeroacoustics Conference.
[8] Boudet J, Léveque E, Touil H. Unsteady Lattice Boltzmann simulations of corner separation in a compressor cascade. J Turbomach 2022;144(1):1–12.
[9] Comparison of a finite volume and two Lattice Boltzmann solvers for swirled confined flows. Comput Fluids 241.
[10] Vinuesa R, Brunton SL. Enhancing computational fluid dynamics with machine learning. Nat Comput Sci 2022;2(6):358–66.
[11] Duraisamy K, Iaccarino G, Xiao H. Turbulence modeling in the age of data. Annu Rev Fluid Mech 2019;51:357–77.
[12] Beck A, Kurz M. A perspective on machine learning methods in turbulence modeling. GAMM-Mitt 2021;44(1).
[13] Rabault J, Kuchta M, Jensen A, Réglade U, Cerardi N. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. J Fluid Mech 2019;865:281–302.
[14] Raissi M, Perdikaris P, Karniadakis G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.

[15] Cai S, Mao Z, Wang Z, Yin M, Karniadakis GE. Physics-informed neural networks (PINNs) for fluid mechanics: A review. 2021, arXiv:2105.09506.
[16] Lou Q, Meng X, Karniadakis GE. Physics-informed neural networks for solving forward and inverse flow problems via the Boltzmann-BGK formulation. J Comput Phys 2021;447:110676.
[17] Hennigh O. Lat-Net: Compressing Lattice Boltzmann flow simulations using deep neural networks. 2017, arXiv:1705.09036.
[18] Corbetta A, Gabbana A, Gyrya V, Livescu D, Prins J, Toschi F. Toward learning Lattice Boltzmann collision operators. Eur Phys J E 2023;46(3).
[19] Prins JHM. Lattice Boltzmann method with a neural network collision operator. Eindhoven University of Technology; 2022.
[20] Bedrunka MC, Wilde D, Kliemank M, Reith D, Foysi H, Krämer A. Lettuce: Pytorch-based lattice Boltzmann framework. Lecture Notes in Comput Sci 2021;12761 LNCS:40–55.
[21] Jinhua L, Lei H, Shu C, Dai C. The more actual macroscopic equations recovered from the lattice Boltzmann equation and their applications. J Comput Phys 2020;415(109546).
[22] Kawai S, Shankar SK, Lele SK. Assessment of localized artificial diffusivity scheme for large-eddy simulation of compressible turbulent flows. J Comput Phys 2010;229(5):1739–62.
[23] Dellar PJ. Lattice Boltzmann algorithms without cubic defects in Galilean invariance on standard lattices. J Comput Phys 2014;259.
[24] Krüger T, Kusumaatmaja H, Kuzmin A, Shardt O, Goncalo S, Viggen EM. The Lattice Boltzmann method, first ed. Springer; 2017, p. 694.
[25] Lycett-Brown D, Luo KH. Multiphase cascaded lattice Boltzmann method. Comput Math Appl 2014;67(2):350–62.
[26] Geier M, Pasquali A, Schönherr M. Parametrization of the cumulant lattice Boltzmann method for fourth order accurate diffusion part I: Derivation and validation. J Comput Phys 2017;348:862–88.
[27] Ricot D, Marié S, Sagaut P, Bailly C. Lattice Boltzmann method with selective viscosity filter. J Comput Phys 2009;228(12):4478–90.
[28] Marié S, Gloerfelt X. Adaptive filtering for the lattice Boltzmann method. J Comput Phys 2017;333:212–26.
[29] Guo Z, Zheng C, Zhao T. A lattice BGK scheme with general propagation. J Sci Comput 2001;16:569–85.
[30] Guo Z, Zhao TS. Explicit finite-difference lattice Boltzmann method for curvilinear coordinates. Phys Rev E 2003;67(6):066709.
[31] Rao PR, Schaefer LA. Numerical stability of explicit off-lattice Boltzmann schemes: A comparative study. J Comput Phys 2015;285:251–64.
[32] Shrestha K, Mompean G, Calzavarini E. Finite-volume versus streaming-based lattice Boltzmann algorithm for fluid-dynamics simulations: A one-to-one accuracy and performance study. Phys Rev E 2016;93(2):1–14.
[33] Horstmann JT. Hybrid numerical method based on the lattice Boltzmann approach with application to non-uniform grids [Ph.D. thesis], Ecole Centrale Lyon; 2018, p. 143.
[34] Latt J, Chopard B. Lattice Boltzmann method with regularized non-equilibrium distribution functions. 2005, arXiv:physics/0506157.
[35] Malaspinas O. Increasing stability and accuracy of the lattice Boltzmann scheme: recursivity and regularization. 2015, arXiv.
[36] Jacob J, Malaspinas O, Sagaut P. A new hybrid recursive regularised Bhatnagar–Gross–Krook collision model for Lattice Boltzmann method-based large eddy simulation. J Turbul 2019;19(11):1051–76.
[37] Spinelli GG, Horstmann T, Masilamani K, Soni MM, Klimach H, Stück A, et al. HPC performance study of different collision models using the Lattice Boltzmann solver Musubi. Comput & Fluids 2023;255.
[38] Coreixas C, Chopard B, Latt J. Comprehensive comparison of collision models in the lattice Boltzmann framework: Theoretical investigations. Phys Rev E 2019;100.
[39] Wissocq G, Sagaut P, Boussuge J-F. An extended spectral analysis of the lattice Boltzmann method: modal interactions and stability issues. J Comput Phys 2019;380:311–33.
[40] De Rosis A. Non-orthogonal central moments relaxing to a discrete equilibrium: A D2Q9 lattice Boltzmann model. Europhys Lett 2016;116(4).
[41] Gendre F. Développement de méthodes de Boltzmann sur réseau en maillages non-uniformes pour l' aéroacoustique automobile [Ph.D. thesis], Aix-Marseille Université; 2018.
[42] Dellar PJ. Bulk and shear viscosities in lattice Boltzmann equations. Phys Rev E 2001.
[43] Marié S, Ricot D, Sagaut P. Comparison between lattice Boltzmann method and Navier-Stokes high order schemes for computational aeroacoustics. J Comput Phys 2009;228(4):1056–70.
[44] Suss A, Mary I, Le Garrec Thomas MS. Comprehensive comparison between the Lattice Boltzmann and Navier-Stokes methods for aerodynamic and aeroacoustic applications. J Comput Fluids 2023;257.
[45] Leonard BP. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. Comput Methods Appl Mech Engrg 1979;19(1):59–98.
[46] Samtaney R, Pullin DI, Kosović B. Direct numerical simulation of decaying compressible turbulence and shocklet statistics. Phys Fluids 2001;13(5):1415–30.
[47] Moin P. Fundamentals of engineering numerical analysis. 2nd ed. Cambridge University Press; 2010.
[48] Bedrunka MC, Horstmann JT, Wilde D, Reith D, Foysi H, Krämer A. Machine learning enhanced collision operator for the Lattice Boltzmann method based on equivariant networks. 2023, unpublished.
[49] Coreixas C, Wissocq G, Puigt G, Boussuge J-F, Sagaut P. Recursive regularization step for high-order Lattice Boltzmann methods. Phys Rev E 2017;96(3).
[50] Wissocq G, Boussuge JF, Sagaut P. Consistent vortex initialization for the athermal Lattice Boltzmann method. Phys Rev E 2020;101(4):1–11.