Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2022-0048

Vol. **33**, No. 1, pp. 144-173
January 2023

Prediction of Moments in the Particles on Demand Method for LBM

Elizaveta Zipunova[1,*], Anastasia Perepelkina[1], Vadim Levchenko[1] and German Zvezdin[1,2]

[1] *Keldysh Institute of Applied Mathematics RAS, 4 Miusskaya sq., Moscow, Russia, 125047.*
[2] *Moscow State University, 1 Ulitsa Kolmogorova, Moscow, Russia, 119991.*

Received 18 February 2022; Accepted (in revised version) 22 July 2022

Abstract. PonD is a method to extend LBM calculations to arbitrary ranges of Mach number and temperature. The current work was motivated by the issue of mass, momentum and energy conservation in the PonD method for LBM. The collision guarantees their conservation, thus, the study involves all aspects of the streaming step: both coordinate and velocity space discretizations, gauge transfer method, resolution of the scheme implicitness. After obtaining the expressions for the change of moments in the system in a time update of the scheme, it was found that the scheme can be formulated as explicit in some cases. Thus, we found the sufficient conditions to make Pond/RegPonD computations explicit and mass, momentum, and energy conserving. The scheme was implemented in the explicit form, and validated for several test cases.

AMS subject classifications: 76N15, 52B10, 65D18, 68U05, 68U07, 65Z05, 65D99

Key words: LBM, compressible, off-grid.

1 Introduction

Mathematical models of fluid physics are based on the Navier-Stokes-Fourier equations (NSFE) at macroscale and on the Boltzmann equation at mesoscale. The Boltzmann equation defines the evolution of the particle distribution function, and the system of Navier-Stokes-Fourier equations describes the behaviour of its velocity moments. Both models are expressions of the conservation principles. Thus, the property of conservation for the numerical methods in CFD (computational fluid dynamics) is as important as accuracy and stability [1].

*Corresponding author. *Email addresses:* e.zipunova@gmail.com (E. Zipunova), mogmi@narod.ru (A. Perepelkina), lev@keldysh.ru (V. Levchenko)

The equations for the moments are often modeled with finite difference, finite volume, or finite element methods [2]. The Boltzmann equation can be modeled with several kinetic schemes, such as discrete velocity models [3,4] and gas-kinetic schemes [5,6]. Among these, the Lattice Boltzmann method [7,8] (LBM) is a very popular method for simulation of fluid dynamics with a kinetic description. The difficulty of the kinetic description is the fact that particle distribution function (PDF) is a 7-dimensional function of 3 coordinates, 3 velocity components and time. Lattice-Boltzmann handles the integration in the velocity space with a small set of Q discrete velocities, thus, gas kinetics is described through the evolution of Q PDF values, which are functions of time and space.

On the one hand, LBM is a node-based method of particle populations propagation on a lattice, which exhibits fluid behaviour. Just like the Chapman-Enskog analysis [9] gives NSFE from the statistics of particle motion, the similar method can be used to obtain NSFE from the motion of virtual particles in LBM. The moment conservation properties are naturally provided by the method construction. On the other hand, it is a method of discretization of the Boltzmann equation [10]. That is its power, since it expresses physics from the kinetic perspective. And that is its weakness, since the discretization of the PDF in the velocity space relies on its closeness to the equilibrium distribution with fixed fluid velocity and temperature.

This issue leads to the understanding that the LBM area of application is limited to low Mach number flows, and problems with very small temperature variations. The examples of such applications are melting and solidification of metals [11, 12], meteorology [13], biological fluid simulation [14], flows in porous media [15], particulate flows [16], automotive industry [17], computer graphics [18].

At the same time, the LBM is very attractive from the computational point of view. The computing cost is significantly less in comparison with the advanced methods of discretization of NSFE, and the method is easy to implement in a parallel program. The locality of the LBM stencil and the simplicity of the calculation in many widely-used variations of the method makes it an attractive platform for the development of the advanced algorithms [19–21], and HPC codes [22].

The latter property in particular raises the question if the method can be used in all parameter ranges of hydrodynamics problems.

Successful simulation of compressible flows are known from the earliest days of LBM existence [23]. Nowadays, there are two major ways to extend LBM capabilities to the compressible regime. One is to use more points in the velocity space to enable accurate integration of higher PDF moments [24–29]. To support high order moment tensor integration in 3D, the number of required points can be several times larger than that in the original LBM, which leads to the memory and performance limitations of the method. At the same time, the velocity and temperature ranges remain limited [30].

The second popular method family is known as DDF (Double Distribution Function) methods [31,32]. The second distribution function is used for higher order moments, for which the accurate computation would require large velocity sets. This way, smaller velocity sets are used for both distributions, but the data storage is doubled. Alternatively,

the thermal equation may be discretized in a Finite Difference or Finite Volume way in hybrid LBM methods [33,34].

Some compressibility errors may be dealt with by corrective terms in the equations [35–39].

Commercial use of LBM in aeroacoustics has started in 2000-s [40–44] There are known applications to transonic and supersonic aeroacoustics [45–47]. Compressible LBM models are developed enough to be implemented in commercially available software such as Simulia PowerFLOW [45,48], ProLB [33,49].

However, even putting this kind of success into consideration, one may note that the LBM models that are extended to describe compressible regimes are comparable in complexity to the conventional CFD methods, while the Mach number and temperature limitations are still present.

In this context, the introduction of the Particles-on-Demand method [50] has come as a major inspiration. In PonD, the velocity set for the streaming step is scaled and shifted adaptively, thus the streaming vectors adapt to the current flow velocity and temperature. This way, the PDF are streamed from off-grid positions. The conversion of a PDF set from one stencil to another is performed through moment matching. The paper [50] included figures which illustrate that the Mach number can be arbitrarily large in simulations, and the temperature can be varied in a wide range, while the LBM formulation remains simple, that is, the basic collision operator is used with a trivial expression for equilibrium, and the velocity set is the minimal set that supports the required moments.

At the same time, the PonD method in [50] was quite raw, and involved multiple issues to be solved in later works. Let us discuss some issues.

Interpolation of the off grid values leads to the loss of the conservation properties of the numerical scheme. In the Semi-Lagrangian methods [51–54], the populations are streamed from the off-grid positions. In the context of LBM, this kind of issue is resolved by resolving spatial discretization with finite-difference [55–58], finite-volume [59], finite element [60], Taylor expansion and least squares approximation [61] methods. Interpolation may be avoided by restricting the adaptive stencils to fit the grid [62,63], or introducing second propagation step [64,65]. In [50], Lagrangian interpolation was used to find off-grid values and stream them onto the grid nodes. There are works that address the conservation properties of PonD [66]. It was reported that some interpolation methods provide mass conservation in the scheme [67].

While the compressibility issues of LBM are dealt with by change of the reference equilibrium, the real compressibility is impossible before the scheme stability is addressed. PonD [50] inherits the problems on discontinuities, such as shock waves, from the LBM method. The issue is resolved with the use of limiters [67]. The numerical dispersion introduced by the interpolation is yet to be studied [68].

The off-grid positions the PDFs are streamed from are unknown until the streaming itself is conducted, so the scheme streaming step is implicit. The predictor-corrector was used in [50] and in later PonD simulations [52, 67] to resolve the implicitness. It was shown experimentally that predictor-corrector procedure converges in up to 4 steps for many cases. However, this procedure multiplies the computing cost of the simulation.

The LBM streaming step becomes computationally heavy in PonD. To make the PonD method ready for industrial applications it should be made not only adequate in the approximation order and stability, but also viable for large-scale 3D simulations. The large part of the computing cost comes from the moment matching gauge transform. If the analytical expression for the gauge transfer matrix is not found, inversion of a matrix is required in each node at each time step. Furthermore, this is repeated several times due to the predictor-corrector scheme. In [50] the analytical expression in a simple form is found for the velocity sets that are a direct product form of the optimal one-dimensional Gauss-Hermite quadrature. These velocity sets are not optimal in 2D and 3D since the exact expressions for the required hydrodynamical moments can be obtained with fewer points in the velocity space. The optimal multidimensional cubatures are not used in most LBM simulations since their points can't be scaled to match the grid nodes, and interpolation of the off-grid values is often avoided. But, in PonD, there is nothing to loose since the streaming positions are already off-grid, and the search of optimal high-order cubatures has become relevant again [54]. Obtaining the explicit gauge transfer expression even for the smallest 3D velocity sets is a difficult task, and the matrix inversion at each lattice node at each time step is a costly operation. To tackle this issue, an alternative way of gauge transfer has been introduced in the RegPonD method [69, 70].

The current work was motivated by the issue of mass, momentum and energy conservation in PonD. The collision guarantees their conservation, thus, this study involves all aspects of the streaming step: both coordinate and velocity space discretizations, gauge transfer method, resolution of the scheme implicitness. After obtaining the expressions for change of the moments in the system in a time update of the scheme, it was found that the scheme can be formulated as explicit in some cases. Thus, we found sufficient conditions to make Pond/RegPonD computations explicit and mass, momentum, and energy conserving. The scheme was implemented in the explicit form, and validated for several test cases.

2 Shifted and scaled stencils for LBM

The particle DF in the Boltzmann equation [9] in D dimensional space:

$$\frac{\partial f(\boldsymbol{x},\boldsymbol{\xi},t)}{\partial t}+\boldsymbol{\xi}\frac{\partial f(\boldsymbol{x},\boldsymbol{\xi},t)}{\partial \boldsymbol{x}}=\hat{\Omega}f, \tag{2.1}$$

where Ω is a collision operator, can be approximated by its projection onto the space spanned by the first N Hermite polynomials [10,71,72]:

$$f(\boldsymbol{x},\boldsymbol{\xi},t)\approx f^N(\boldsymbol{x},\boldsymbol{\xi},t)=\omega(\boldsymbol{\xi})\sum_{n=0}^{N}\frac{1}{n!}\boldsymbol{a}^{(n)}(\boldsymbol{x},t)\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{\xi}), \tag{2.2}$$

where coefficients $\boldsymbol{a}^{(n)}(\boldsymbol{x},t)$ are the Hermite moments of f [71,73]:

$$\boldsymbol{a}^{(n)}(\boldsymbol{x},t)=\int f(\boldsymbol{x},\boldsymbol{\xi},t)\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{\xi})d\boldsymbol{\xi}, \tag{2.3}$$

and $\omega(\boldsymbol{x})$ is the weight function

$$\omega(\boldsymbol{x})=\frac{1}{(2\pi)^{D/2}}e^{-|\boldsymbol{x}|^2/2}. \tag{2.4}$$

As in [73], $\boldsymbol{x}$, $\boldsymbol{\xi}$ are D-dimensional vectors, and $\boldsymbol{a}^{(n)}$, $\boldsymbol{\mathcal{H}}^{(n)}$ are symmetrical n-rank tensors in D-dimensions. These tensors have $\frac{(D+n-1)!}{(D-1)!n!}$ unique components.

The flow density ρ, velocity $\boldsymbol{u}$ and temperature T are obtained from the raw moments $\boldsymbol{m}^{(n)}$. $m_{\alpha\beta\gamma}$ are the components of the $\boldsymbol{m}^{(n)}$ tensor, $\alpha+\beta+\gamma=n$,

$$\begin{aligned}
&m_{\alpha\beta\gamma}=\int f\xi_x^\alpha\xi_y^\beta\xi_z^\gamma d\boldsymbol{\xi},\\
&\rho=m_{000}=\int f d\boldsymbol{\xi},\quad \rho\boldsymbol{u}=(m_{100},m_{010},m_{001})^{\mathrm{T}}=\int f\boldsymbol{\xi}d\boldsymbol{\xi},\\
&\rho(u^2+T)=m_{200}+m_{020}+m_{002}=\int f\boldsymbol{\xi}^2 d\boldsymbol{\xi}.
\end{aligned} \tag{2.5}$$

The collision operator $\hat{\Omega}$ is taken in the BGK form [74,75]: $-(f-f^{\mathrm{eq}})/\tau$, where f^{eq} is the equilibrium distribution:

$$f^{\mathrm{eq}}(\rho,\boldsymbol{u},T,\boldsymbol{\xi})=\frac{\rho}{(2\pi T)^{D/2}}e^{-(\boldsymbol{\xi}-\boldsymbol{u})^2/(2T)}=\frac{\rho}{T^{D/2}}\omega\left(\frac{\boldsymbol{\xi}-\boldsymbol{u}}{\sqrt{T}}\right). \tag{2.6}$$

We do not use other collision operators in this paper, since the focus is on the PonD-specific scheme features. Many advanced collision operators [39,76–78] may be used together with PonD to improve scheme stability.

Let us perform the LBM construction [8,10] in an arbitrary frame of reference. Let us take a gauge $\lambda=\{\boldsymbol{u}^\lambda,T^\lambda\}$ with arbitrary values $\boldsymbol{u}^\lambda$ and T^λ (which, in general, are not the moments of f but are some constant parameters of the method), and consider another expansion of a PDF:

$$f(\boldsymbol{x},\boldsymbol{\xi},t)=f(\boldsymbol{x},\sqrt{T^\lambda}\boldsymbol{v}+\boldsymbol{u}^\lambda,t)=\omega(\boldsymbol{v})\sum_{n=0}^{\infty}\frac{1}{n!}\boldsymbol{d}^{\lambda(n)}(\boldsymbol{x},t)\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{v}). \tag{2.7}$$

$\boldsymbol{v}=\frac{\boldsymbol{\xi}-\boldsymbol{u}^\lambda}{\sqrt{T^\lambda}}$ is the velocity in a moving reference frame λ. In the classic LBM construction [10], $\lambda=\lambda_0=\{\mathbf{0},1\}$. Hermite moments in a scaled gauge are denoted $\boldsymbol{d}^{\lambda(n)}$. The conversion from raw moments to $\boldsymbol{d}^{\lambda(n)}$ is discussed in Appendix A. $f^{\lambda N}$ is another approximation of $f(\boldsymbol{x},\boldsymbol{\xi},t)$.

$$f^{\lambda N}(\boldsymbol{x},\sqrt{T^\lambda}\boldsymbol{v}+\boldsymbol{u}^\lambda,t)=\frac{\omega(\boldsymbol{v})}{\sqrt{T^\lambda}^D}\sum_{n=0}^{N}\frac{1}{n!}\boldsymbol{d}^{\lambda(n)}(\boldsymbol{x},t)\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{v}), \tag{2.8}$$

$$\begin{aligned}\boldsymbol{d}^{\lambda(n)}(\boldsymbol{x},t)&=\int f^{\lambda N}(\boldsymbol{x},\boldsymbol{\xi},t)\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{v})d\boldsymbol{\xi}=\int\sqrt{T^\lambda}^D f^{\lambda N}(\boldsymbol{x},\sqrt{T^\lambda}\boldsymbol{v}+\boldsymbol{u}^\lambda,t)\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{v})d\boldsymbol{v}\\&=\sum_{q=1}^{Q}\left(\sqrt{T^\lambda}^D w_q\frac{f^{\lambda N}(\boldsymbol{x},\sqrt{T^\lambda}\boldsymbol{c}_q+\boldsymbol{u}^\lambda,t)}{\omega(\boldsymbol{c}_q)}\right)\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{c}_q)=\sum_{q=1}^{Q}f_q^\lambda\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{c}_q),\end{aligned}$$

$$f_q^\lambda=w_q\sum_{n=0}^{N}\frac{1}{n!}\boldsymbol{d}^{\lambda(n)}(\boldsymbol{x},t)\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{c}_q). \tag{2.9}$$

If values $\boldsymbol{u}^\lambda$ and T^λ are the actual macro values at point $\boldsymbol{x}$ and time t then:

$$f_q^{eq,\lambda}=\rho w_q. \tag{2.10}$$

By evaluating the Boltzmann equation at the $\boldsymbol{e}_q=\sqrt{T^\lambda}\boldsymbol{c}_q+\boldsymbol{u}^\lambda$ nodes and multiplying the resulting expressions by $w_q/\omega(\boldsymbol{c}_q)$ we get:

$$\frac{\partial f_q^\lambda}{\partial t}+\boldsymbol{e}_q\frac{\partial f_q^\lambda}{\partial\boldsymbol{x}}=\hat{\Omega}(f_1,f_2,\cdots,f_Q),\quad q=1,\cdots,Q.$$

The raw moments are computed as

$$\boldsymbol{m}^{(n)}=\sum_{i=1}^{Q}f_i^\lambda\boldsymbol{e}_q^{(n)}. \tag{2.11}$$

This way, LBM is formulated in relation to some gauge λ.

In the PonD method, the gauge varies for each node for every time step. To relate the equations for different gauges, the moment matching condition is used

$$\hat{M}^\lambda f_q^\lambda=\hat{M}^{\lambda'}f_q^{\lambda'}, \tag{2.12}$$

where $\hat{M}^\lambda$ is the matrix of transformation to the raw moment space. Thus, when the variables $f_q^{\lambda'}$ are known, the following expression is used to find f_q^λ in another gauge

$$f_q^\lambda=(\hat{M}^\lambda)^{-1}\hat{M}^{\lambda'}f_q^{\lambda'}. \tag{2.13}$$

For the velocity sets in the product form of one-dimensional quadratures the $\hat{M}^\lambda$ matrix is square, and the symbolic expression for the transfer matrix $(\hat{M}^\lambda)^{-1}\hat{M}^{\lambda'}$ is found [50,66].

In the RegPonD method, the gauge transform is implemented through the Hermite expansion of the PDF. The data are stored on the grid as a set of raw moments $\boldsymbol{m}^{(n)}$, which are transformed to $\boldsymbol{d}^{\lambda(n)}$ for the required λ, and PDFs are computed with (2.9).

LBM, PonD and RegPonD contain two sub-steps. In the collision, the PDFs are changed locally in the lattice nodes. Since gauge λ is matched to the current temperature and flow rate of the node, the equilibrium distribution form is trivial:

$$f_q^{\lambda *}(\boldsymbol{x},t) = -\frac{f_q^{\lambda}(\boldsymbol{x},t) - f_q^{\lambda,\mathrm{eq}}(\rho)}{\tau}; \quad f_q^{\lambda \mathrm{eq}} = \rho w_q; \quad \rho = \sum_q f_q^{\lambda}(\boldsymbol{x},t). \tag{2.14}$$

The streaming step becomes implicit, since the streaming vectors are unknown until after the streaming:

$$f_q^{\lambda}(\boldsymbol{x},t) = f_q^{\lambda *}(\boldsymbol{x} - \boldsymbol{e}_q, t), \quad \boldsymbol{e}_q = \sqrt{T^{\lambda}}\boldsymbol{c}_q + \boldsymbol{u}^{\lambda}, \tag{2.15}$$

$$\rho \boldsymbol{u}^{\lambda} = (m_{100}, m_{010}, m_{001})^{\mathrm{T}} = \sum_q \boldsymbol{e}_q f_q^{\lambda}(\boldsymbol{x},t), \tag{2.16}$$

$$\rho(u^{\lambda 2} + T^{\lambda}) = m_{200} + m_{020} + m_{002} = \sum_q f_q^{\lambda}(\boldsymbol{x},t)|\boldsymbol{e}_q|^2.$$

3 Prediction of moments in PonD

3.1 Method concept

The predictor-corrector step is used in the original PonD method since, as can be seen in (2.15), the streaming for each node $\boldsymbol{x}_j$ is performed from the $\boldsymbol{x}_j - \boldsymbol{e}_q$ positions, where $\boldsymbol{e}_q$ vectors depend on $\boldsymbol{u}^{\lambda}$, T^{λ} which can only be computed after the streaming. At the same time, if all calculations of a scheme update for one node are inserted into one another, the moments can be found explicitly in some cases. This method of prediction of moments on the next time step is proposed in [66] for 1D. Let us remind the idea, and extend it to 3D.

Let us consider a streaming into the origin point $\boldsymbol{x} = (0,0,0)$ in 3D: $f_q^{\lambda}(0, t+\delta t) = f_q^{\lambda}(-\boldsymbol{e}_q^{\lambda}, t)$. The moments $\boldsymbol{m}^{(n)}$ are computed as polynomials in $\boldsymbol{e}_q^{\lambda}$. Their components in 3D are:

$$m_{\alpha\beta\gamma}(t+\delta t) = \sum_{q=1}^{Q} f_q^{\lambda}(-\boldsymbol{e}_q, t) e_{qx}^{\alpha} e_{qy}^{\beta} e_{qz}^{\gamma}. \tag{3.1}$$

The values streamed to the origin are approximated in the $-\boldsymbol{e}_q$ position one way or another. Let us take some polynomial approximation

$$f_q^{\lambda}(-\boldsymbol{e}_q^{\lambda}, t) = \sum_i^{L} \mathcal{L}_i(-\boldsymbol{e}_q^{\lambda}) f_q^{\lambda}(\boldsymbol{r}_i, t), \tag{3.2}$$

where $\mathcal{L}_i(-\boldsymbol{e}_q^\lambda)$ is the approximation coefficient, $\boldsymbol{r}_i$ are the points of the spatial reconstruction stencil. $\mathcal{L}_i(-\boldsymbol{e}_q^\lambda)$ may be chosen as Lagrange interpolation coefficients. In that case, (3.2) is the expression for the Lagrange interpolation. If $\mathcal{L}_i(-\boldsymbol{e}_q^\lambda)$ are expressed as any polynomials of the position:

$$\mathcal{L}_i(-\boldsymbol{e}_q^\lambda)=\sum_l A_l e_{qx}^{\alpha_l} e_{qy}^{\beta_l} e_{qz}^{\gamma_l} \tag{3.3}$$

Let $N_l=\max_l(\alpha_l+\beta_l+\gamma_l)$ be the maximum order of a monomial in this sum. By inserting (3.3) into (3.1) and performing the sum in Q we get

$$\begin{aligned} m_{\alpha\beta\gamma}(t+\delta t) &= \sum_{q=0}^{Q-1}\sum_i^L\left(\sum_l A_l e_{qx}^{\alpha_l} e_{qy}^{\beta_l} e_{qz}^{\gamma_l}\right) f_q^\lambda(\boldsymbol{r}_i) e_{qx}^{\alpha} e_{qy}^{\beta} e_{qz}^{\gamma} \\ &= \sum_i^L\sum_l A_l m_{\alpha+\alpha_l,\beta+\beta_l,\gamma+\gamma_l}(\boldsymbol{r}_i). \end{aligned} \tag{3.4}$$

With this expression, some of the moments in the origin point are computed through the values of the moments in the neighboring points as a finite difference expression. The f_q^λ and $\boldsymbol{e}_q^\lambda$ values are defined in the yet unknown gauge λ, however, the moments on the RHS do not depend on λ if moment matching is valid for them.

If the expression is symmetrical and uniform for all nodes, the moments in the LHS are conserved. If velocity and temperature can be predicted through this expression, the $\boldsymbol{e}_q$ vectors are known before streaming, the scheme is explicit, and predictor-corrector iterations are not needed. The conditions of applicability of PoMPonD are detailed in the next section.

Thus, we propose the PoMPonD (Prediction of Moments Particles-on-Demand) method, where the moments are predicted with the (3.4) expression for an iteration-free streaming. It can be used both in PonD and RegPonD formulations of the moment matching.

3.2 PoMPonD with RegPonD

As an extension of RegPonD, PoMPomD method is as follows. All mesh data are stored as $\boldsymbol{m}^{(n)}$, $n=\overline{0,N}$. Thus, $\boldsymbol{m}^{(n)}$ are interpolated instead of f_q.

$$\boldsymbol{m}^{(n)}(-\boldsymbol{e}_q^\lambda,t)=\sum_i^L \mathcal{L}_i(-\boldsymbol{e}_q^\lambda)\boldsymbol{m}^{(n)}(\boldsymbol{r}_i,t). \tag{3.5}$$

Since $\boldsymbol{m}^{(n)}$ linearly depend on f_q^λ (see (2.11)), the derivation of PoMPonD above remains valid. Let us show this. f_q^λ are expressed through (2.9) where $\boldsymbol{d}^{\lambda(n)}$ are linearly expressed through the interpolated $\boldsymbol{m}^{(n)}$. The coefficients in the latter linear expression depend on

the yet unknown target gauge λ, but, whatever λ may be equal to, the expression for interpolation is also the same due to linearity

$$\boldsymbol{d}^{\lambda(n)}(-\boldsymbol{e}_q^{\lambda},t)=\sum_i^L \mathcal{L}_i(-\boldsymbol{e}_q^{\lambda})\boldsymbol{d}^{\lambda(n)}(\boldsymbol{r}_i,t). \tag{3.6}$$

Inserting this into (2.9) and rearranging the summation proves that (3.2) is valid in RegPonD.

3.3 Algorithm

According to the PoMPonD in RegPonD formulation, for each node at $\boldsymbol{x}$ one needs to

- Predict moments $\boldsymbol{m}^{(0)}$, $\boldsymbol{m}^{(1)}$, and diagonal components of $\boldsymbol{m}^{(2)}$ at t from moments $\boldsymbol{m}^{(0)},\cdots,\boldsymbol{m}^{(2+N_l)}$ at $t-\Delta_t$, and calculate $\lambda'(\boldsymbol{x},t)$ from these moments.
- For each $\boldsymbol{c}_q$, $q=\overline{1,Q}$:
  - find $\boldsymbol{m}^{(n)}(\boldsymbol{x}-(\sqrt{T}\boldsymbol{c}_q+\boldsymbol{u}))$ with interpolation,
  - calculate $\boldsymbol{d}^{\lambda'(n)}$ from $\boldsymbol{m}^{(n)}$ in the computed gauge $\lambda'(\boldsymbol{x},t)$ (see Appendix A),
  - calculate $f_q^{\lambda'}$ from $\boldsymbol{d}^{\lambda'(n)}$ with (2.9),
  - stream $f_q^{\lambda'}$ to the point $\boldsymbol{x}$,
  - perform collision.
- Calculate new values of $\boldsymbol{m}^{(n)}$ and store them.

With some parameters regular PonD/RegPonD schemes become equivalent to this scheme unintentionally, as was observed in [66].

4 Variations of the numerical methods

The PoMPonD method inherits the flexibility of LBM. Let us study some of the possible parameter variations which are relevant to the introduced idea. In Section 4.4, the relationship between the parameters, for which PoMPonD is valid, is discussed.

4.1 Velocity Sets

The choice of a velocity set is a basic variation of LBM. In PonD and RegPonD, there is no limitation set by the distance between mesh nodes, and more quadratures variations are possible. Optimal quadratures are preferred to get the required order with less quadrature nodes. The optimal 1D quadratures are found from the zeros or Hermite polynomials. A multidimensional quadrature of the same order can be found as the Cartesian

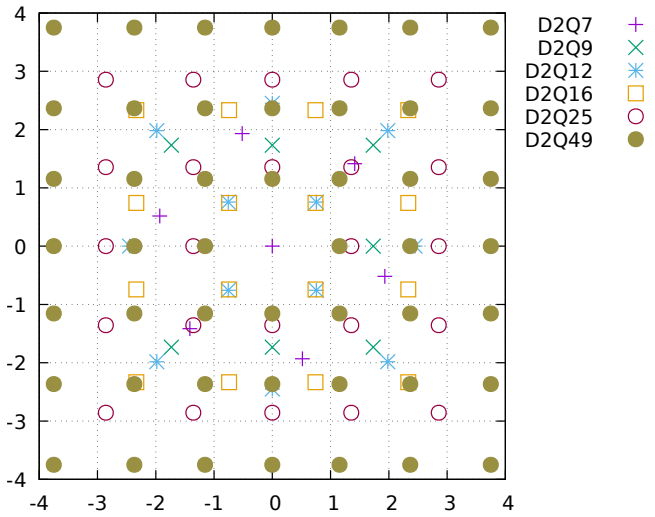

Figure 1: Two dimensional velocity sets used in the simulations.

product of the 1D optimal quadrature. However, a quadrature with the same order and lesser number of nodes exists for relevant cases.

The optimal many-dimensional cubature rules can be found in the reference literature [79, 80], open libraries [81], other LBM works [10, 54, 82].

For the reasons explained in Section 4.4, in this work, we had to use high-order multidimensional velocity sets. Several issues were discovered with the sets that are rarely used in the LBM simulations. The D3Q45 cubature in [82] is presented in the form that is difficult for a human to retype from text, and it is more convenient to obtain from the source [83]. The 7-th order 27-point 3D cubature $E^{27}_{3,7}$ in [10] was retyped from the source [79] with a hard to identify misprint. The incorrect 7-th order 16-point 2D cubature $E^{16}_{2,7}$ in [10] was copied from the source [79], where it is similarly incorrect, as it is in [81] which used the same source. Indeed, the 16-point 7-th order 2D cubature can be obtained by a direct product of the optimal D1Q4 quadrature, thus, other 16-point 7-th order sets are not relevant, and the origin of the mistake in [79] is unknown.

Thus, we had to make a velocity set generator for our purposes, which is described in Appendix B. A similar approach for on-grid LBMs was used before in [84]. For a given set of cubature points, the generator finds weights for the optimal order cubature, checks the order of the cubature, and outputs the solution to be inserted in the program code of the simulation. We take the velocity sets from the sources [79, 83]. It was found that the symmetric sets in the form of regular polygons can be rotated while retaining the order. Thus, we tested an original rotation of the D2Q7 set in the current work (Listing 1).

The cubatures that were used for our simulations are presented in Table 1. The 2D sets are shown in Fig. 1. Note that the distance of the points from the center increases with the order, thus, the reference lattice temperature has to be set sufficiently low.

Table 1: Velocity sets for Gaussian quadrature rules.

Code	Velocities	Weights
D1Q3	$(0,0)$	$2/3$
	$\pm\sqrt{3}$	$1/6$
D1Q4	$\pm\sqrt{3-\sqrt{6}}$	$(3+\sqrt{6})/12$
	$\pm\sqrt{3+\sqrt{6}}$	$(3-\sqrt{6})/12$
D1Q5	$(0,0)$	$8/15$
	$\pm\sqrt{5+\sqrt{10}}$	$7-2\sqrt{10}/60$
	$\pm\sqrt{5-\sqrt{10}}$	$7+2\sqrt{10}/60$
D1Q7	Roots of $H^{(7)}(x)=0$	found numerically
D2Q7	$(0,0)$	$1/2$
	$2(\cos\frac{n\pi}{3},\sin\frac{n\pi}{3}),\quad n=\overline{1,6}$	$1/12$
D2Q9	$(0,0)$	$4/9$
	$(\pm1,0),(0,\pm1)$	$1/9$
	$(\pm1,\pm1)$	$1/36$
D2Q12	$(\pm\sqrt{6},0),(0,\pm\sqrt{6})$	$1/36$
	$(\pm\sqrt{(9-3\sqrt{5})/4},\pm\sqrt{(9-3\sqrt{5})/4})$	$(5+2\sqrt{5})/45$
	$(\pm\sqrt{(9+3\sqrt{5})/4},\pm\sqrt{(9+3\sqrt{5})/4})$	$(5-2\sqrt{5})/45$
D2Q16	$(\pm\sqrt{3+\sqrt{6}},\pm\sqrt{3+\sqrt{6}})$	$(5-2\sqrt{6})/48$
	$(\pm\sqrt{3-\sqrt{6}},\pm\sqrt{3-\sqrt{6}})$	$(5+2\sqrt{6})/48$
	$(\pm\sqrt{3+\sqrt{6}},\pm\sqrt{3-\sqrt{6}})$	$1/48$
	$(\pm\sqrt{3-\sqrt{6}},\pm\sqrt{3+\sqrt{6}})$	$1/48$
D2Q25	Cartesian product formula from D1Q5	
D2Q49	Cartesian product formula from D1Q7	
D3Q13	$(0,0,0)$	$2/5$
	$(\pm\frac{\sqrt{5+\sqrt{5}}}{2},\pm\frac{\sqrt{5-\sqrt{5}}}{2},0),$	$1/20$
	$(0,\pm\frac{\sqrt{5+\sqrt{5}}}{2},\pm\frac{\sqrt{5-\sqrt{5}}}{2}),$	$1/20$
	$(\pm\frac{\sqrt{5-\sqrt{5}}}{2},0,\pm\frac{\sqrt{5+\sqrt{5}}}{2})$	$1/20$
D3Q27	Cartesian product formula from D1Q3	
D3Q27e	$(0,0,0)$	$(720\pm8\sqrt{15})/2205$
	$(\sqrt{(15\pm\sqrt{15})/2},0,0)_{\mathrm{FS}}$	$(270\mp46\sqrt{15})/15435$
	$(\sqrt{6\mp\sqrt{15}},\sqrt{6\mp\sqrt{15}},0)_{\mathrm{FS}}$	$(162\pm41\sqrt{15})/6174$
	$t=\sqrt{9\pm2\sqrt{15}},\ (\pm t,\pm t,\pm t)$	$(783\mp202\sqrt{15})/24696$
D3Q45 [83]	$(0,0,0)$	$28/135$
	icosahedron with $\lvert c_i\rvert=\sqrt{4.5}$	$1/216$
	dodecahedron with $\lvert c_i\rvert=\sqrt{1.5}$	$25/432$
	icosahedron with $\lvert c_i\rvert=\sqrt{7.5}$	$1/2160$
D3Q64	Cartesian product formula from D1Q4	
D3Q125	Cartesian product formula from D1Q5	

4.2 RegPond regularization order

The vectors of moments on both sides of expression (2.12) should be Q values long for the matrix to be reversible. In the original paper [50], the velocity space discretization is taken as a product form of a one-dimensional quadrature. If there are Q discrete velocities in D dimensions and $Q=Q_1^D$ for some integer Q_1, the moment matrix has the form $M_{l_1 l_2 \cdots l_D} = e_{qx_1}^{l_1} e_{qx_2}^{l_2} \cdots e_{qx_D}^{l_D}$ where each index l_{x_α} is $1,\cdots,Q_1$. Evidently, not all of these moments are present in the Chapman-Enskog derived hydrodynamics, and not all of them can be obtained with valid quadrature rules. Nevertheless, the linear transformation is deterministic and preserves correct moments.

In RegPonD, the set of moments that are used in the expansion can be varied: any moments can be included in the sum, whether they are computed correctly with the Gaussian quadrature or not. If we take the same velocity set for PonD and RegPonD, and use the same set of moments in (2.9) and (2.12), the results match due to the uniqueness of linear basis transformation.

On the other hand, in RegPonD, we can omit the moments that are physically irrelevant. The choice of moments in the Hermite expansion is one of the parameters of the method. In this work, we take all moments with the order equal or less than some integer N_{reg}.

The choice of the moments influence the computational complexity both in PonD and RegPonD. In the original moment matching (2.12), the moment set determines the size of the matrix to be inverted. In RegPonD, raw moments $\boldsymbol{m}^{(n)}$ have to be converted to shifted scaled Hermite moments $\boldsymbol{d}^{(n)}$. The number of operations required for the conversion increases with N_{reg}. We use an efficient implementation of this conversion, which is described in Appendix A.

Thus, the minimal Q and N_{reg} values that allow accurate simulation should be found.

4.3 Reconstruction of the off-grid PDF

Let us study some variations of spatial approximation of the off-grid values. We choose the method among polynomial approximations so that the sum in (3.4) is possible.

The approximation is not necessary an interpolation, since we do not require the values of the approximating polynomial to match on-grid PDF values. Smaller order is desirable in the chosen polynomial, and interpolation on the smallest 3D stencil contains high-order terms.

To enable summation in (3.4) all Q values that are streamed into one point are to be approximated with one function of the position. B-splines contain segments that are expressed as polynomials with different coefficients. To use PoMPonD, we have to require all Q values to fall into one segment. There are several segments per cell, and the $\boldsymbol{e}_q$ length is of the order of cell size. Thus, B-splines are not considered in this work.

Let us consider streaming into $\boldsymbol{x}=0$ point. Let us denote for convenience

$$f = f_q\big|_{-\boldsymbol{e}_q}, \quad \boldsymbol{e}_q = \boldsymbol{e}, \quad f^{ijk} = f_q\big|_{(ijk)},$$

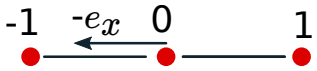

Figure 2: 3-point interpolation in 1D.

and the moments of order $\alpha,\beta,..$ in each component as $m_{xy\cdots}^{\alpha\beta\cdots}$ in the position $(x,y,\cdots)$, and in the position $-\boldsymbol{e}_q$ if the position index is omitted.

Example 1. $D=1$, $L=3$, $N_l=2$, the interpolation stencil contains three points $x_{-1}=-1$, $x_0=0$, $x_1=1$. 9 coefficients should be found: $A_j^{(l)}$, $j=-1,0,1$, $l=0,1,2$.

Lagrange polynomial interpolation provides the following coefficients :

$$\mathcal{L}_i(x)=\prod_{k=1,i\neq k}^{L}\frac{(x-x_k)}{(x_i-x_k)}.$$

In the $-\boldsymbol{e}_q$ position (Fig. 2),

$$\mathcal{L}_0(-e_q)=1-e_q^2,\quad \mathcal{L}_1(-e_q)=\frac{1}{2}(e_q^2-e_q),\quad \mathcal{L}_{-1}(-e_q)=\frac{1}{2}(e_q^2+e_q).$$

The solution is the system found in [66]

$$f=f^0+e_x\left(-\frac{1}{2}f^{+1}+\frac{1}{2}f^{-1}\right)+e_x^2\left(\frac{1}{2}f^{-1}-f^{+0}+\frac{1}{2}f^{+1}\right) \tag{4.1}$$

and the expression for moment prediction is

$$\boldsymbol{m}^l=\boldsymbol{m}_0^{(l+0)}-\frac{1}{2}\boldsymbol{m}_{+1}^{l+1}+\frac{1}{2}\boldsymbol{m}_{-1}^{l+1}-\boldsymbol{m}_0^{l+2}+\frac{1}{2}\boldsymbol{m}_{+1}^{l+2}+\frac{1}{2}\boldsymbol{m}_{-1}^{l+2}. \tag{4.2}$$

Example 2. In 2D, if the direct product Lagrange interpolation is used as in previous works [50,66], RHS of (3.2) would contain monomials of an order higher than 2:

$$\begin{aligned}
f=&f^{000}-e_y\left(\frac{1}{2}f^{010}+\frac{-1}{2}f^{0-10}\right)+e_y^2\left(-f^{000}+\frac{1}{2}f^{010}+\frac{1}{2}f^{0-10}\right)-e_x\left(\frac{1}{2}f^{100}+\frac{-1}{2}f^{-100}\right)\\
&+e_x^2\left(-f^{000}+\frac{1}{2}f^{100}+\frac{1}{2}f^{-100}\right)+e_xe_y\left(\frac{1}{4}f^{110}+\frac{-1}{4}f^{1-10}+\frac{-1}{4}f^{-110}+\frac{1}{4}f^{-1-10}\right)\\
&-e_xe_y^2\left(\frac{-1}{2}f^{100}+\frac{1}{4}f^{110}+\frac{1}{4}f^{1-10}+\frac{1}{2}f^{-100}+\frac{-1}{4}f^{-110}+\frac{-1}{4}f^{-1-10}\right)\\
&-e_x^2e_y\left(\frac{-1}{2}f^{010}+\frac{1}{2}f^{0-10}+\frac{1}{4}f^{110}+\frac{-1}{4}f^{1-10}+\frac{1}{4}f^{-110}+\frac{-1}{4}f^{-1-10}\right)\\
&+e_x^2e_y^2\left(f^{000}+\frac{-1}{2}f^{010}+\frac{-1}{2}f^{0-10}+\frac{-1}{2}f^{100}+\frac{1}{4}f^{110}+\frac{1}{4}f^{1-10}+\right.\\
&\left.\frac{-1}{2}f^{-100}+\frac{1}{4}f^{-110}+\frac{1}{4}f^{-1-10}\right).
\end{aligned} \tag{4.3}$$

This is a fourth order polynomial, but the interpolation is second order.

Example 3. Let us drop the terms of the order $o(e_x^3)$ and smaller. In that case, the spatial dependency of f_q is approximated with a polynomial that is not equal to the PDF values at the lattice nodes.

$$\begin{aligned} f=&f^{00}+e_xe_y\frac{1}{4}(f^{11}-f^{1-1}-f^{-11}+f^{-1-1})-e_y\frac{1}{2}(f^{01}-f^{0-1})\\ &+e_y^2\left(-f^{00}+\frac{1}{2}f^{01}+\frac{1}{2}f^{0-1}\right)-e_x\frac{1}{2}(f^{10}-f^{-10})+e_x^2\left(-f^{00}+\frac{1}{2}f^{10}+\frac{1}{2}f^{-10}\right). \end{aligned} \tag{4.4}$$

Other polynomial forms can be found with the least square minimization [85].

Example 4. Let us take 19-point stencil with $D=3$. Let us require the approximation to take the form

$$f=Af^{000}-Be_{qx}(f_q^{100}-f^{-100})-Be_{qy}(f^{010}-f^{0-10})-Be_{qz}(f^{001}-f^{00-1})+\cdots,$$

where A, B, are the yet unknown coefficients, chosen symmetrically in some cases. All f_q values on the RHS are expanded into Taylor series in relation to the $(-\boldsymbol{e}_q)$ position and the coefficients are matched. The Taylor expansion can be made valid for up to 3-rd order with the following expression:

$$\begin{aligned} f=&f^{000}-\frac{1}{2}e_x(f^{100}-f^{-100})-\frac{1}{2}e_y(f^{010}-f^{0-10})-\frac{1}{2}e_z(f^{001}-f^{00-1})\\ &+\frac{1}{2}e_x^2(f^{100}+f^{-100}-2f^{000})+\frac{1}{2}e_y^2(f^{010}+f^{0-10}-2f^{000})+\frac{1}{2}e_z^2(f^{001}+f^{00-1}-2f^{000})\\ &+\frac{1}{4}e_xe_y(f^{110}+f^{-1-10}-f^{1-10}-f^{-110})+\frac{1}{4}e_ze_y(f^{011}+f^{0-1-1}-f^{01-1}-f^{0-11})\\ &+\frac{1}{4}e_xe_z(f^{101}+f^{-10-1}-f^{10-1}-f^{-101}). \end{aligned} \tag{4.5}$$

And the expression for the moments is:

$$\begin{aligned} m_{\alpha\beta\gamma}=&m_{\alpha\beta\gamma}^{000}-\frac{1}{2}(m_{\alpha+1\beta\gamma}^{100}-m_{\alpha+1\beta\gamma}^{-100})-\frac{1}{2}(m_{\alpha\beta+1\gamma}^{010}-m_{\alpha\beta+1\gamma}^{0-10})-\frac{1}{2}(m_{\alpha\beta\gamma+1}^{001}-m_{\alpha\beta\gamma+1}^{00-1})\\ &+\frac{1}{2}(m_{\alpha+2\beta\gamma}^{100}+m_{\alpha+2\beta\gamma}^{-100}-2m_{\alpha+2\beta\gamma}^{000})+\frac{1}{2}(m_{\alpha\beta+2\gamma}^{010}+m_{\alpha\beta+2\gamma}^{0-10}-2m_{\alpha\beta+2\gamma}^{000})\\ &+\frac{1}{2}(m_{\alpha\beta\gamma+2}^{010}+m_{\alpha\beta\gamma+2}^{0-10}-2m_{\alpha\beta\gamma+2}^{000})+\frac{1}{4}(m_{\alpha+1\beta+1\gamma}^{110}+m_{\alpha+1\beta+1\gamma}^{-1-10}-m_{\alpha+1\beta+1\gamma}^{1-10}-m_{\alpha+1\beta+1\gamma}^{-110})\\ &+\frac{1}{4}(m_{\alpha\beta+1\gamma+1}^{011}+m_{\alpha\beta+1\gamma+1}^{0-1-1}-m_{\alpha\beta+1\gamma+1}^{01-1}-m_{\alpha\beta+1\gamma+1}^{0-11})\\ &+\frac{1}{4}(m_{\alpha+1\beta\gamma+1}^{101}+m_{\alpha+1\beta\gamma+1}^{-10-1}-m_{\alpha+1\beta\gamma+1}^{10-1}-m_{\alpha+1\beta\gamma+1}^{-101}). \end{aligned} \tag{4.6}$$

The moment update scheme is a finite-difference expression, and the right and the left flux terms can be separated if needed. This form allows to see that fluxes of the neighboring points cancel each other on the grid, and the moments, for which the expression is valid, are conserved in streaming.

4.4 Validity conditions

In [66] and [69], it was experimentally found that some simulations conserve mass exactly, and in [66] the reason was shown for 1D simulations. There, PoMPonD was outlined as a suggestion, but the predictor-corrector procedure was still used, even though it was observed to converge to the same value that was predicted explicitly. Here, with the scheme variations reported in the current work, we can finally tell the sufficient conditions under which the PonD/RegPonD scheme become explicit.

Let us consider (3.4) expression once more. It is valid under the assumption that the moments on the RHS are invariant in the gauge transformation. The moments included in (3.4) should be computable both in the original gauge and the final gauge, and the values should match in the two gauges. To predict density, we require the validity of (3.4) up to $\alpha+\beta+\gamma \equiv N_m = 0$, and up to $N_m = 1$ and $N_m = 2$ to predict first and second moments. For this, in RegPonD, the moments on the RHS of (3.4) have to be included in the reconstruction formula (2.9): $N_{reg} \geq N_m + N_l$.

Then, we require the support of this computation in the velocity set. Since the PDF is the polynomial of order $N_{reg} \geq N_m + N_l$ and the moments of the order $N_m + N_l$ have to be computable, the approximation order that has to be supported by the velocity set is $N_Q \geq (N_m + N_l) + N_{reg} \geq 2(N_m + N_l)$. The parameter requirements and the velocity sets that support such schemes are reported in Table 2.

When $\boldsymbol{e}_q = \sqrt{T}\boldsymbol{c}_q + \boldsymbol{u}$ are scaled both with temperature and velocity, the trace of the second moment tensor should be known before the streaming. With (3.4), the second moment can be predicted if the conditions of the third row of Table 2 are satisfied. For athermal simulations, the satisfaction of the conditions in the second row of the table in enough.

There is no N_{reg} parameter in PonD. For the validity of (3.4) in PonD, the moments of the RHS should be included in the moment matrix that is used for moment matching.

Another important requirement is that all interpolation or approximation of f_q streamed to one location are performed with the same stencil. In PonD [50], this may take place unintentionally, and mass can be incidentally conserved. Here we rely on the requirement in the scheme construction, and we have to make sure the lattice temperature is low enough for the velocity stencil to fit into the interpolation stencil. This imposes Courant-like condition on the length of the scheme dependencies. It appears that another mass-conservative scheme DuGKS-PoND [86] has this limitation as well.

Other validity conditions may be satisfied even when the requirements of Table 2 are not met; the conditions are sufficient but not necessary. For example, any finite velocity set in 1D satisfies the condition $\prod_{q'}^{Q}(c_q - c_{q'}) = 0$ for any q. In D1Q3, it reads $c_q^3 - c_q = 0$,

Table 2: Parameter requirements for moment prediction.

For prediction of	In (2.9) $N_{reg} \geq$	quadrature order $N_Q \geq$	Velocity set for $N_l = 2$		
				direct product	minimal
mass, $N_m = 0$	$0+N_l$	$(0+N_l)+N_{reg}$	1D	D1Q3	
			2D	D2Q9	D2Q7
			3D	D3Q27	D3Q13
momentum, $N_m = 1$	$1+N_l$	$(1+N_l)+N_{reg}$	1D	D1Q4	
			2D	D2Q16	D2Q12
			3D	D3Q64	D3Q27e
energy, $N_m = 2$	$2+N_l$	$(2+N_l)+N_{reg}$	1D	D1Q5	
			2D	D2Q25	
			3D	D3Q125	D3Q45

and the third order moment is equal to the first one. Thus, its value is not lost in the moment conversion that takes account of the moments up to the second order. Additionally, some cubatures may be exact for some components of higher order moment tensors. We expect that more symmetries may be found, and that the optimal parameters may not necessarily satisfy the conditions in Table 2.

The conditions, however, are not satisfied with the minimal schemes found in [69] for RegPonD simulations. With RegPonD, athermal problems can be simulated with minimal cubatures of the 5-th order, such as D2Q7 and D2Q13. To be able to predict moments and make the scheme explicit, the minimal order of the cubature is 7.

4.5 Other

It is interesting to note that the collision and streaming steps are interchangeable in the current scheme. Indeed, BGK collision requires the data from other f_q only in the form of the conserved moments. In PoMPonD, the conserved moments are predicted before reconstruction of f_q, thus, the collision can be performed immediately. It can be useful to remember for the future developments of LBM methods that PDF values do not have to be collected in the same place to collide with each other.

5 Simulation results

5.1 1D Riemann problems

Let us start with the 1D Riemann problem reported in [66]

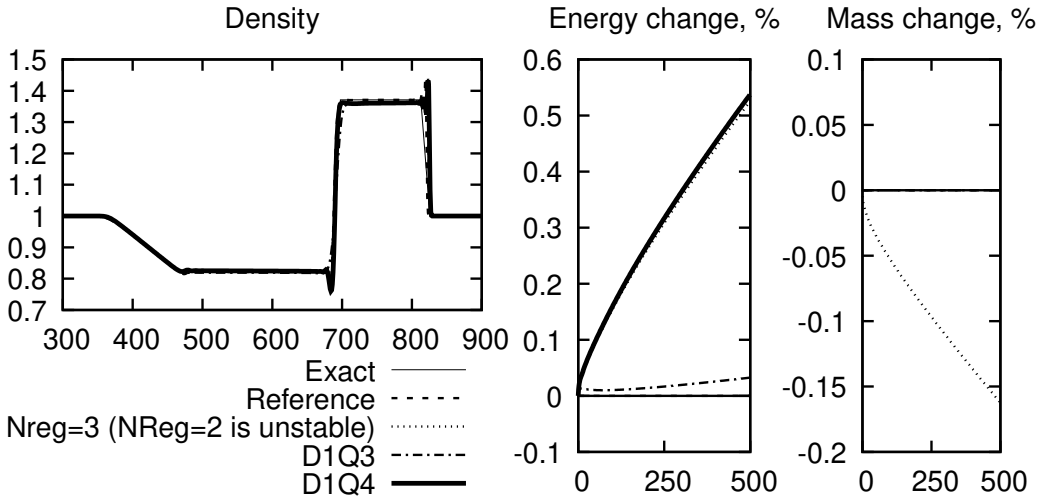

Figure 3: 1D Riemann problem tests.

$$\begin{aligned} &\rho|_{x<0}=1, \quad \rho|_{x\geq 0}=1; \\ &P|_{x<0}=0.1, \quad P|_{x\geq 0}=0.02; \\ &u|_{x<0}=0, \quad u|_{x\geq 0}=0. \end{aligned} \tag{5.1}$$

According to the minimal requirements for validity of PoMPond, the problem is solved with N_{Reg}=4, D1Q5 quadrature, 3 point stencil for finding the off-grid values ((4.1), (4.2)).

The computation is effectively identical to the one in [66]. RegPonD and PonD are identical since there are exactly 5 PDFs for each node, which are converted to 5 moments and back. PoMPonD and PonD are identical since the predictor-corrector iterations should give the same solution as the proposed finite difference scheme for $\boldsymbol{u}$, T.

Let us check the validity requirements that were derived in Section 4.4 by changing one parameter at a time.

First, we decrease N_{Reg}. With $N_{Reg}=3$ the solution changes, and energy is not conserved. With $N_{Reg}=2$ the solution can not be correct since the regularization looses the third order moment which is required in the Chapman-Enskog equation. In the trial simulation, it diverged.

Second, we return $N_{Reg}=4$ and change the quadrature. D1Q5 has approximation order equal to 9, so it is enough to find the fourth moment of the fourth order polynomial, thus, energy is conserved exactly. The order of approximation of D1Q4 is 7, and it is enough to find second order tensor of the fourth order polynomial, thus, mass is conserved.

5.2 Shear wave test

The shear wave test in [50] has proved Galilean invariance of the PonD method in arbitrarily high Mach number flows. In [69], the simulation was recreated with the RegPonD method.

RegPonD allows smaller velocity sets to simulate the same physical phenomena, and, in [69], it was shown that D2Q7 is the smallest velocity set valid for the task. Let us study if the Galilean invariance is satisfied with PoMPonD, and what is the smallest velocity set to support it.

The initial conditions are:

$$u_t(x,y) = 0.05\sin\left(2\pi\frac{x+y}{S}\right), \quad u_\ell = Ma\sqrt{T}, \tag{5.2}$$
$$u_x = (u_t - u_\ell)/\sqrt{2}, \quad u_y = (u_t + u_\ell)/\sqrt{2}, \quad \rho = 1$$

in an $S\times S$ domain in 2D. According to the Navier-Stokes solution, the decay rate of the velocity amplitude is $\kappa = (\tau - 0.5)|\boldsymbol{k}^2|T$, where $\boldsymbol{k} = \{2\pi/S, 2\pi/S, 0\}$.

For an athermal problem, only the velocity of the new gauge should be predicted. According to Table 2, if the interpolation order is at least 2, $N_{reg} \geq 3$ and $N_Q \geq 6$ are required. Thus, at least 12 discrete velocities should be used.

Another issue is that the fixed interpolation stencil introduces the limitation on the maximum streaming distance. Whatever is the chosen stencil for spacial approximation, we require all $\boldsymbol{e}_q$ vectors to fall into the convex hull of this stencil. The more rigorous theoretical stability analysis is left for future studies, and here we test stability with the simulation runs (Fig. 4).

The longest discrete velocity among the studied cubatures is from the D2Q49 set, and its value is close to 4. To be sure it fits the stencil, $T = 0.25/4^2$ is set for all simulations in this section.

We start with $Ma = 0$, $S = 128$, $\tau = 0.6$ parameters to study the validity of the scheme. After 3000 time steps, the mass gain per cell, the difference between predicted $\boldsymbol{u}$ and its value after streaming $\boldsymbol{u}'$ at one randomly chosen reference node, and the error in the fitted decay rate are studied (Table 3).

Experiment #1 satisfies the outlined conditions, and it is proven with the computation result: the velocity is predicted with the error of the order of the machine zero. In experiments #2–#5, we tested some deviations in the cubature rule and N_{reg}. As expected, the error in the prediction of the reference velocity is observed when the validity conditions are not satisfied. Mass gain, however, remained small, which is probably caused by the symmetry of the problem.

In the experiment #6, we show that the 2D Lagrange interpolation is not suited for explicit simulations in PoMPonD. To satisfy the presented conditions, the order of the cubature should be at least 10, and it is proven in the experiment #7, where the prediction of velocity is valid. Such high order of approximation for athermal problems does not

Table 3: Shear wave decay computation results. The 'valid' column shows if the validity conditions are satisfied.

#	valid	stencil	cubature	N_{reg}	mass gain	$\|\boldsymbol{u}-\boldsymbol{u}'\|/\|\boldsymbol{u}\|$	$\|\kappa-\kappa'\|/\kappa'$
1	yes	(4.4)	D2Q12	3	-2.9e-13	2.8e-16	7.8e-03
2	no	(4.4)	D2Q7	3	9.0e-13	4.0e-11	7.8e-03
3	no	(4.4)	D2Q12	2	-4.3e-13	2.1e-10	7.7e-03
4	no	(4.4)	D2Q7	2	9.2e-13	4.2e-11	7.8e-03
5	yes	(4.4)	D2Q25	4	-3.4e-12	1.4e-15	7.8e-03
6	no	(4.3)	D2Q12	3	7.3e-08	2.6e-08	5.0e-02
7	yes	(4.3)	D2Q49	5	2.4e-12	8.3e-16	7.3e-03

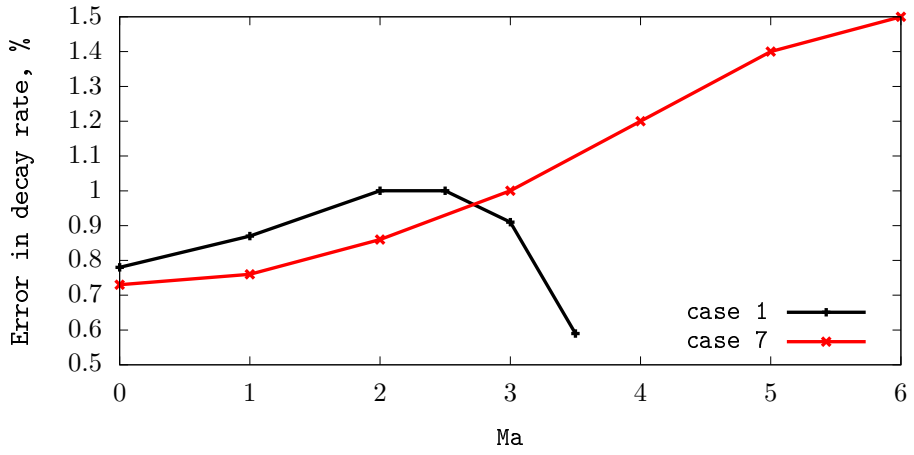

Figure 4: Error in decay rate vs Ma number.

seem adequate in practice. Experiments have also shown that the error in the decay rate depends more on the spatial interpolation method than on any other scheme parameter.

Let us take cases 1 and 7 from Table 3 and test the Galilean invariance of the method by increasing the background flow rate. The results are presented in Fig. 4. The critical Mach number when some e_q becomes greater than the mesh space is 7.4 for case 1 and 5.6 for case 7. However, in the case 1, the simulation becomes unstable at $Ma=4$. It appears that the fact that the approximating polynomial does not match on-grid PDF values for case 1 is crucial for the diagonal propagation, and the limit on streaming distances is more strict.

Finally, the spatial approximation introduces numerical dispersion. With $Ma=0$ the dependency of the fitted decay rate error on the ppw (points per wavelength) is plotted in Fig. 5.

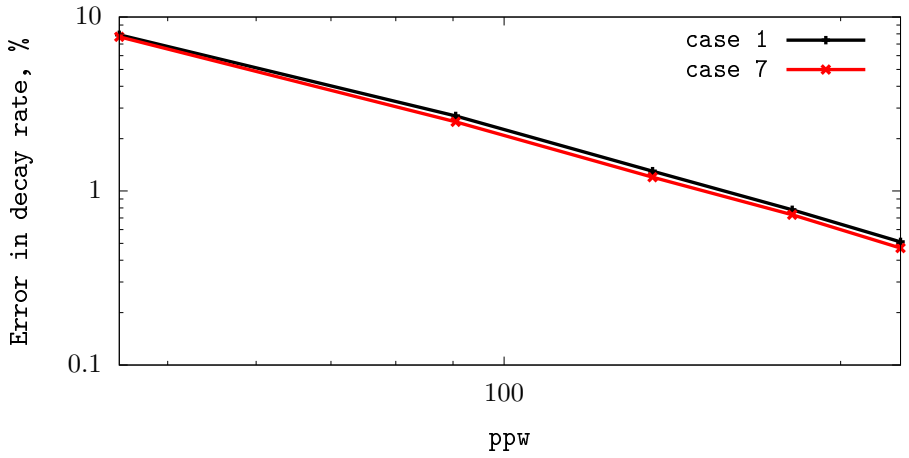

Figure 5: Error in decay rate vs points per wavelength.

5.3 Performance considerations

The use of explicit moment prediction allows to update a node in one iteration. In [50] and [69] the simulations converged in 1-3 predictor-corrector iterations. These figures are provided for problems with smooth variations of flow variables, more iterations are observed near shock waves. Therefor, we can estimate that the number of required calculations per time step is at least 1-3 times less in PoMPonD.

On the other hand, implicit schemes are often preferred since they allow larger time steps, thus, less time steps total are required for the same results. The same is true here, since we had to introduce Courant-like condition: the length of $\boldsymbol{e}_q$ should be less than the stencil size for spatial approximation.

In [69], the shear wave test was performed with $T_L = 1/3$. If we estimate the highest temperature for which $\boldsymbol{e}_q$ falls inside the approximation stencil, we require ≈ 1.5 more time steps than was used there. Additionally, D2Q12 set was used here and D2Q7 was used in [69].

In total, the effective performance gain in the current case is marginal. Therefore, the main advantage of PoMPonD is the guaranteed moment conservation.

6 Conclusion

In the course of restoring conservation properties to the PonD method, we have discovered its interesting property. It appears that the scheme that is formulated as implicit can become explicit with some variation of the method. The explicit formulation of the RegPonD is proposed in this work, and it is called PoMPonD. This variation interconnects various aspects of the scheme: spatial discretization, velocity discretization, and

reconstruction of the PDF values. Thus, we studied the variations and found the validity conditions of the newly introduced PoMPonD.

The explicit formulation, as well as the low order polynomials for the spatial interpolation that are introduced here, reduce the computation cost of the method. About 1-4 predictor-corrector iterations were required in PonD and RegPonD. Thus, without predictor-corrector iteration, the amount of operations is reduced several times.

We have shown that lower order polynomials are preferred in the spatial approximation. The influence of the choice on the numerical dispersion and isotropy remains to be studied.

Unfortunately, the minimal order of the velocity set that is required for the new PoMPonD method has a higher number of points than was reported sufficient for the same problem previously [70].

Acknowledgments

The work is supported by Russian Science Foundation, grant # 18-71-10004.

Appendices

A Moment space conversion

Multiple times in the simulation a set of $M=\frac{(D+n)!}{D!n!}$ moments $\boldsymbol{m}^{(n)}$ has to be converted to a set of $\boldsymbol{d}^{\lambda(n)}$ moments. The conversion has to be implemented with a minimal number of operations. The set of moments $\boldsymbol{m}^{(n)}$ is discarded after that, so the conversion can potentially be in-place, without creation of a second data array. We have found an efficient procedure for the moment conversion. The procedure can be useful in other aspects of LBM methods, such as conversion to central moment space [87], temperature scaled moments [76], and obtaining high-order moments of the equilibrium distribution for given $\boldsymbol{u}$, T.

The $\boldsymbol{d}^{\lambda(n)}$ moments can also be obtained through the recursive expressions from the raw moments $\boldsymbol{m}^{(n)}$. Let us recall their definition:

$$\boldsymbol{d}^{(n)}(\boldsymbol{x},t)=\int\sqrt{T}^{D}f(\boldsymbol{x},\sqrt{T}\boldsymbol{v}+\boldsymbol{u},t)\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{v})d\boldsymbol{v};\quad \boldsymbol{m}^{(n)}(\boldsymbol{x},t)=\int f(\boldsymbol{x},\boldsymbol{\xi},t)\boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{\xi})d\boldsymbol{\xi},$$

and $\boldsymbol{v}=\frac{\boldsymbol{\xi}-\boldsymbol{u}}{\sqrt{T}}$. The recursive expression for Hermite polynomials reads [73]

$$\mathcal{H}^{(n+1)}_{\alpha+1\beta\gamma}(\boldsymbol{v})=v_x\mathcal{H}^{(n)}_{\alpha\beta\gamma}(\boldsymbol{v})-\alpha\mathcal{H}^{(n-1)}_{\alpha-1\beta\gamma}(\boldsymbol{v}).$$

After multiplication by $f(\boldsymbol{\xi})$ and integrating both sides of the equation with respect to $\boldsymbol{\xi}$, we obtain:

$$\int\mathcal{H}^{(n+1)}_{\alpha+1\beta\gamma}(\boldsymbol{v})f(\boldsymbol{\xi})d\boldsymbol{\xi}=\int\frac{(\xi_x-u_x)}{\sqrt{T}}\mathcal{H}^{(n)}_{\alpha\beta\gamma}(\boldsymbol{v})f(\boldsymbol{\xi})d\boldsymbol{\xi}-\int\alpha\mathcal{H}^{(n-1)}_{\alpha-1\beta\gamma}(\boldsymbol{v})f(\boldsymbol{\xi})d\boldsymbol{\xi}.$$

Let λ_l be the coefficients of the polynomial

$$\mathcal{H}^{(n)}_{\alpha\beta\gamma}(\boldsymbol{v}) = \mathcal{H}^{000}_{\alpha\beta\gamma} = \sum_l \lambda^{(n)}_l \xi_x^{\alpha_l} \xi_y^{\beta_l} \xi_z^{\gamma_l}.$$

Let us define $\mathcal{H}^{ijk}_{\alpha\beta\gamma}$ polynomials as

$$\mathcal{H}^{ijk}_{\alpha\beta\gamma} = \sum_l \lambda^{(n)}_l \xi_x^{\alpha_l+i} \xi_y^{\beta_l+j} \xi_z^{\gamma_l+k}; \quad \mathcal{H}^{(n)}_{\alpha\beta\gamma}(\boldsymbol{v}) = \mathcal{H}^{000}_{\alpha\beta\gamma}(\boldsymbol{v}).$$

That is, the coefficient before $\xi_x^{\alpha_l+i} \xi_y^{\beta_l+j} \xi_z^{\gamma_l+k}$ in $\mathcal{H}^{ijk}_{\alpha\beta\gamma}$ is equal to the coefficient before $\xi_x^{\alpha_l} \xi_y^{\beta_l} \xi_z^{\gamma_l}$ in the Hermite polynomial $\mathcal{H}^{(n)}_{\alpha\beta\gamma}(\boldsymbol{v})$ for each l. Similarly, let us define $d^{ijk}_{\alpha\beta\gamma}$ so that

$$d^{(n)}_{\alpha\beta\gamma} = d^{000}_{\alpha\beta\gamma} = \sum_l \lambda^{(n)}_l m_{\alpha_l\beta_l\gamma_l}, \quad d^{ijk}_{\alpha\beta\gamma} = \sum_l \lambda^{(n)}_l m_{\alpha_l+i,\beta_l+j,\gamma_l+k}.$$

Then

$$\begin{aligned}
&\int \mathcal{H}^{000}_{\alpha+1\beta\gamma} f(\boldsymbol{\xi}) d\boldsymbol{\xi} = \frac{1}{\sqrt{T}} \int \left(\mathcal{H}^{100}_{\alpha+1\beta\gamma} - u_\alpha \mathcal{H}^{000}_{\alpha\beta\gamma} \right) f(\boldsymbol{\xi}) d\boldsymbol{\xi} - \int \alpha \mathcal{H}^{000}_{\alpha-1\beta\gamma} f(\boldsymbol{\xi}) d\boldsymbol{\xi}, \\
&d^{000}_{\alpha+1\beta\gamma} = \frac{1}{\sqrt{T}} \left(d^{100}_{\alpha+1\beta\gamma} - u_\alpha d^{000}_{\alpha\beta\gamma} \right) - \alpha d^{000}_{\alpha-1\beta\gamma}.
\end{aligned} \tag{A.1}$$

The same expression can be written for other axes. Let $\theta = \frac{1}{\sqrt{T}}$. Even though a new set of variables $d^{ijk}_{\alpha\beta\gamma}$ is introduced, the conversion can be implemented in such a way that it is not necessary to allocate any storage for them. Here is the complete system that is used for the conversion from $m_{\alpha\beta\gamma}$ to $d_{\alpha\beta\gamma}$:

$$\begin{aligned}
&d^{\alpha\beta\gamma}_{000} = m_{\alpha\beta\gamma}, \\
&d^{ijk}_{\alpha+1\beta\gamma} = \theta \left(d^{i+1jk}_{\alpha\beta\gamma} - u_x d^{ijk}_{\alpha\beta\gamma} \right) - \alpha d^{ijk}_{\alpha-1\beta\gamma}, \\
&d^{ijk}_{\alpha\beta+1\gamma} = \theta \left(d^{ij+1k}_{\alpha\beta\gamma} - u_y d^{ijk}_{\alpha\beta\gamma} \right) - \beta d^{ijk}_{\alpha\beta-1\gamma}, \\
&d^{ijk}_{\alpha\beta\gamma+1} = \theta \left(d^{ijk+1}_{\alpha\beta\gamma} - u_z d^{ijk}_{\alpha\beta\gamma} \right) - \gamma d^{ijk}_{\alpha\beta\gamma-1}, \\
&d_{\alpha\beta\gamma} = d^{000}_{\alpha\beta\gamma}.
\end{aligned} \tag{A.2}$$

It can be observed that this system is stencil scheme in the α–β–γ space (Fig. 6). The dependencies in the graph show the proper order of computation, and even some asynchronous portions can be observed. It is evident from the dependencies that one can convert moments in-place: one column of the dependency graph illustration corresponds to a code variable, and no more than M variables are required at any time. Since the data copy is not needed, even for large sets of moments the conversion can be localized in GPU registers (for GPU codes) or L1 cache (for CPU codes).

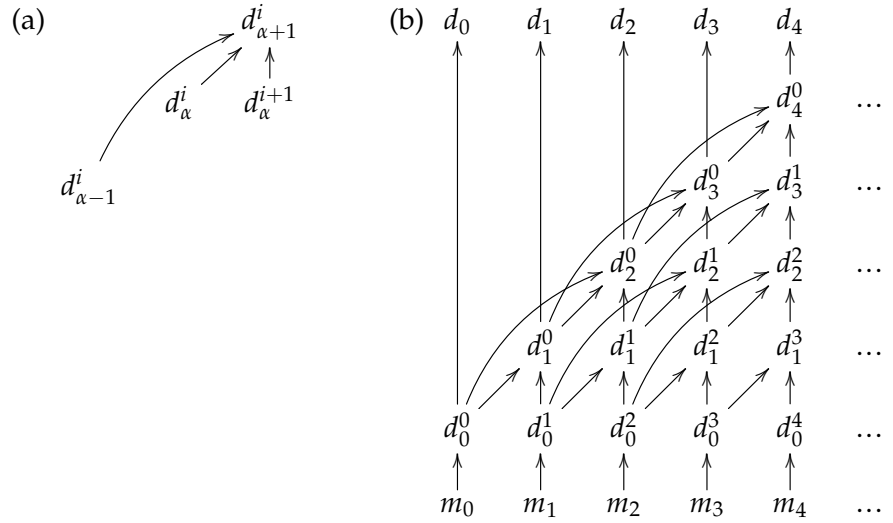

Figure 6: A 1D stencil projection (a) and a sample dependency graph (b) for the problem of moment space conversion (A.2).

Thus, the system (A.2) should be implemented for the moment conversion. We made a Python script that generates code for all equations in the system in the target programming language. The input parameters are the number of dimensions and the maximal order of moments. The use of such code is very convenient, since the conversion of 3D tensors of the orders 4 and higher contains many terms. The expressions are not hard to find with some computer algebra system, but they become large and hard to debug. Additionally, simple expressions such as (A.2) are more suitable for compiler optimizations.

B Automatic generation of velocity sets

B.1 Definitions

Consider the general form of the expression (2.3)

$$\int_{-\infty}^{+\infty} \omega(\boldsymbol{x}) f(\boldsymbol{x})\, dx \approx \sum_{q=1}^{Q} w_q f(\boldsymbol{c}_q). \tag{B.1}$$

Here w_q are the cubature weights and $\boldsymbol{c}_q$, $q=1,\cdots,Q$ is the set of the cubature points. The curature formula B.1 has the order of approximation of N_Q if the equality holds exactly, provided that $f(\boldsymbol{x})$, $\boldsymbol{x}\in\mathbf{R}^D$ is a polynomial of the total degree no higher than N_Q. By definition, Hermite polynomials in 3D are given by the following expression [73]

$$\boldsymbol{\mathcal{H}}^{(n)}_{\alpha\beta\gamma} = \frac{(-1)^n}{\omega(\boldsymbol{x})} \frac{d^\alpha}{dx^\alpha} \frac{d^\beta}{dy^\beta} \frac{d^\gamma}{dz^\gamma} \omega(\boldsymbol{x}), \tag{B.2}$$

where $\omega(\boldsymbol{x})$ is defined in (2.4).

B.2 System of equations

Let us use the orthogonality of Hermite polynomials, $\mathcal{H}^{(0)}=1$:

$$\int_{-\infty}^{\infty}\mathcal{H}^{(n)}(\boldsymbol{x})\omega(\boldsymbol{x})dx=\delta_{n0}=\sum_{q=1}^{Q}\mathcal{H}^{(n)}(\boldsymbol{c}_q)w_q. \tag{B.3}$$

It is a system of linear algebraic equations with unknown weights w_q. We require the system to be satisfied up to the maximal possible order N_Q, and the maximum N_Q considered in this work is 11. Let us calculate the size of the resulting system of equations. The number of polynomials of the order n is $\frac{\prod_{i=1}^{D-1}(n+i)}{(D-1)!}$ (Fig. 7).

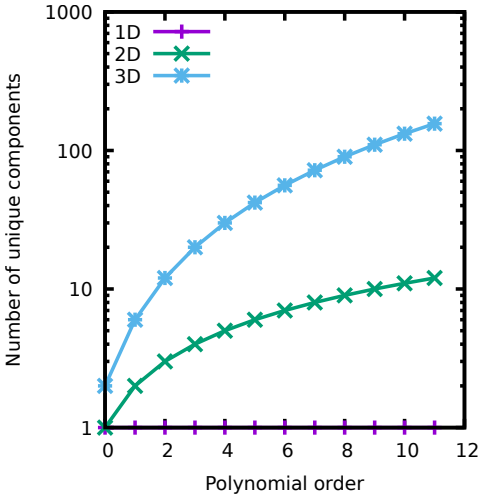

Figure 7: The number of unique tensor components vs the order of polynomial.

B.3 Optimization

The size of the system can be significantly reduced. We define a shell S_i, $i=0,1,\cdots$ as a set of points equidistant from the center, with some kind of a symmetry. When the points of the shell have equal weights, the equations of the system at the odd orders are satisfied trivially. For example, regular polygons and Platonic solids define sets of points with the same weights.

Let's use this to optimize calculations. We break the original system into shells, we

get:

$$w_0 \sum_{\boldsymbol{c}_q \in S_0} \boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{c}_q) + w_1 \sum_{\boldsymbol{c}_q \in S_1} \boldsymbol{\mathcal{H}}^{(n)}(\boldsymbol{c}_q) + \cdots = \delta_{n0}. \tag{B.4}$$

The system obtained in this way is significantly smaller.

The number of equations in the system is found in the following manner. First, we build a matrix by appending rows in the order of increasing n. The order of rows with same n which represent different components of the tensor of a given order n is arbitrary. If, after appending a row, the rank of the matrix is not increased, there is a linear dependence of the current row on some of the previous ones. In this case, the row is erased and the next component is tested. This is repeated until the rank of the matrix is equal to the number of shells. Then the matrix is inverted and the weights are found.

B.4 Program interface

The generator is written in C++17. The input is a set of user-defined shells. Since many cubatures consist of several regular shapes, basic classes describing regular polygons and polyhedra were implemented in the program interface. Some shells can be set manually by specifying the points.

The output is an array of weights w_q for each shell, and the order N_Q of the cubature rule that was obtained. With this information, the output can be configured to generate code for any programming language. The sample output in Listing 1 corresponds to D2Q7. It was generated for the simulations in the current paper, which are implemented with CUDA.

Listing 1: Code example of CUDA code generator

```
//ORDER-5
#define DIM 2
const int Nq = 7, NQ = Nq;
__device__ const ftype3 ciq[] = {
{1.414213562373095368, 1.414213562373095145, 0.000000000000000000},
{-0.517638090205041368, 1.931851652578137069, 0.000000000000000000},
{-1.931851652578136846, 0.517638090205042145, 0.000000000000000000},
{-1.414213562373095590, -1.414213562373095145, 0.000000000000000000},
{0.517638090205040702, -1.931851652578137291, 0.000000000000000000},
{1.931851652578137069, -0.517638090205041479, 0.000000000000000000},
{0.000000000000000000, 0.000000000000000000, 0.000000000000000000},
};
const ftype w0 = 0.083333333333333315;
const ftype w1 = 0.500000000000000222;
__device__ const ftype wiq[] = {w0, w0, w0, w0, w0, w0, w1};
```

A more detailed description of the software can be found in [88].

References

[1] Yu P Popov and Aleksander Andreevich Samarskii. Completely conservative difference schemes. *USSR Computational Mathematics and Mathematical Physics*, 9(4):296–305, 1969.

[2] M P Galanin and E B Savenkov. *Methods of Numerical Analysis of Mathematical Models*. Bauman Moscow State Technical University, Moscow, 2010. In Russian.

[3] BT Nadiga and DI Pullin. A method for near-equilibrium discrete-velocity gas flows. *Journal of Computational Physics*, 112(1):162–172, 1994.

[4] C.Baranger, J.Claudel, N.Hérouard, and L.Mieussens. Locally refined discrete velocity grids for stationary rarefied flow simulations. *Journal of Computational Physics*, 257:572–593, 2014.

[5] Zhaoli Guo, Kun Xu, and Ruijie Wang. Discrete unified gas kinetic scheme for all Knudsen number flows: Low-speed isothermal case. *Physical Review E*, 88(3):033305, 2013.

[6] Zhaoli Guo, Ruijie Wang, and Kun Xu. Discrete unified gas kinetic scheme for all Knudsen number flows. II. Thermal compressible case. *Physical Review E*, 91(3):033313, 2015.

[7] Sauro Succi. *The lattice Boltzmann equation: for complex states of flowing matter*. Oxford University Press, 2018.

[8] Timm Krüger, Halim Kusumaatmaja, Alexandr Kuzmin, Orest Shardt, Goncalo Silva, and Erlend Magnus Viggen. The lattice Boltzmann method. *Springer International Publishing*, 10(978-3):4–15, 2017.

[9] Sydney Chapman and Thomas George Cowling. *The Mathematical Theory of Non-Uniform Gases: An Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases*. Cambridge University Press, 1990.

[10] S. Xiaowen, Y. Xue-feng, and H. Chen. Kinetic theory representation of hydrodynamics: A way beyond the Navier–Stokes equation. *J. Fluid Mech.*, 550:413–441, 2006.

[11] Andrey Zakirov, Sergei Belousov, Maria Bogdanova, Boris Korneev, Andrey Stepanov, Anastasia Perepelkina, Vadim Levchenko, Andrey Meshkov, and Boris Potapkin. Predictive modeling of laser and electron beam powder bed fusion additive manufacturing of metals at the mesoscale. *Additive Manufacturing*, 35:101236, 2020.

[12] Regina Ammer, Matthias Markl, Ulric Ljungblad, Carolin Körner, and Ulrich Rüde. Simulating fast electron beam melting with a parallel thermal free surface lattice Boltzmann method. *Computers & Mathematics with Applications*, 67(2):318–330, 2014.

[13] Takashi Shimokawabe, Toshio Endo, Naoyuki Onodera, and Takayuki Aoki. A stencil framework to realize large-scale computations beyond device memory capacity on GPU supercomputers. In *Cluster Computing (CLUSTER)*, pages 525–529. IEEE, 2017.

[14] Rafik Ouared and Bastien Chopard. Lattice Boltzmann simulations of blood flow: Non-Newtonian rheology and clotting processes. *Journal of Statistical Physics*, 121(1-2):209–221, 2005.

[15] Benjamin Ahrenholz, Jonas Tölke, and Manfred Krafczyk. Lattice-Boltzmann simulations in reconstructed parametrized porous media. *International Journal of Computational Fluid Dynamics*, 20(6):369–377, 2006.

[16] Christoph Rettinger and Ulrich Rüde. A comparative study of fluid-particle coupling methods for fully resolved lattice Boltzmann simulations. *Computers & Fluids*, 154:74–89, 2017.

[17] Christoph A Niedermeier, Christian F Janßen, and Thomas Indinger. Massively-parallel multi-GPU simulations for fast and accurate automotive aerodynamics. In *7th European Conference on Computational Fluid Dynamics*, 2018.

[18] Nils Thurey. *Physically based animation of free surface flows with the lattice Boltzmann method*. PhD thesis, University of Erlangen, 2007.

[19] Gerhard Wellein, Georg Hager, Thomas Zeiser, Markus Wittmann, and Holger Fehske. Efficient temporal blocking for stencil computations by multicore-aware wavefront parallelization. In *2009 33rd Annual IEEE International Computer Software and Applications Conference*, volume 1, pages 579–586. IEEE, 2009.

[20] Anthony Nguyen, Nadathur Satish, Jatin Chhugani, Changkyu Kim, and Pradeep Dubey. 3.5-D blocking optimization for stencil computations on modern CPUs and GPUs. In *SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–13. IEEE, 2010.

[21] Vadim Levchenko, Anastasia Perepelkina, and Andrey Zakirov. New compact streaming in LBM with ConeFold LRnLA algorithms. In Sergey Sobolev Vladimir Voevodin, editor, *Supercomputing. RuSCDays 2020. Communications in Computer and Information Science*, volume 1331, pages 50–62, 2020.

[22] Christoph Riesinger, Arash Bakhtiari, Martin Schreiber, Philipp Neumann, and Hans-Joachim Bungartz. A holistic scalable implementation approach of the lattice Boltzmann method for CPU/GPU heterogeneous clusters. *Computation*, 5(4):48, 2017.

[23] Frank J Alexander, Hudong Chen, Shiyi Chen, and GD Doolen. Lattice Boltzmann model for compressible fluids. *Physical Review A*, 46(4):1967, 1992.

[24] IV Karlin SS Chikatamarla. Lattices for the lattice Boltzmann method. *Physical Review E*, 79(4):046701, 2009.

[25] IV Karlin SS Chikatamarla. Entropy and galilean invariance of lattice Boltzmann theories. *Physical Review Letters*, 97(19):190601, 2006.

[26] X He X Shan. Discretization of the velocity space in the solution of the Boltzmann equation. *Physical Review Letters*, 80(1):65, 1998.

[27] H Chen X Shan, XF Yuan. Kinetic theory representation of hydrodynamics: A way beyond the Navier–Stokes equation. *Journal of Fluid Mechanics*, 550:413–441, 2006.

[28] Nicolò Frapolli, Shyam S Chikatamarla, and Iliya V Karlin. Entropic lattice Boltzmann model for compressible flows. *Physical Review E*, 92(6):061301, 2015.

[29] Nicolò Frapolli, Shyam S Chikatamarla, and Ilya V Karlin. Entropic lattice Boltzmann model for gas dynamics: Theory, boundary conditions, and implementation. *Physical Review E*, 93(6):063302, 2016.

[30] Shyam Chikatamarla Nicolò Frapolli and Ilya Karlin. Theory, analysis, and applications of the entropic lattice Boltzmann model for compressible flows. *Entropy*, 22:370, 2020.

[31] G Doolen X He, S Chen. A novel thermal model for the lattice Boltzmann method in incompressible limit. *J. Comp. Phys.*, 146:282–300, 1998.

[32] Succi S D'Orazio A. Simulating two-dimensional thermal channel flows by means of a lattice Boltzmann method with new boundary conditions. *Future Generation Computer Systems*, 20(6):935–944, 2004.

[33] S. Guo, Yongliang Feng, Jérôme Jacob, F. Renard, and Pierre Sagaut. An efficient lattice Boltzmann method for compressible aerodynamics on D3Q19 lattice. *Journal of Computational Physics*, 418:109570, 2020.

[34] Florian Renard, Yongliang Feng, Jean-François Boussuge, and Pierre Sagaut. Improved compressible hybrid lattice Boltzmann method on standard lattice for subsonic and supersonic flows. *Computers & Fluids*, 219:104867, 2021.

[35] MH Saadat, SA Hosseini, B Dorschner, and IV Karlin. Extended lattice Boltzmann model for gas dynamics. *Physics of Fluids*, 33(4):046104, 2021.

[36] Mohammad Hossein Saadat, Fabian Bösch, and Ilya V Karlin. Lattice Boltzmann model for compressible flows on standard lattices: Variable Prandtl number and adiabatic exponent.

Physical Review E, 99(1):013306, 2019.
[37] Paul J. Dellar. Lattice Boltzmann algorithms without cubic defects in Galilean invariance on standard lattices. *Journal of Computational Physics*, 259:270–283, 2014.
[38] Nikolaos I. Prasianakis and Iliya V. Karlin. Lattice Boltzmann method for simulation of compressible flows on standard lattices. *Phys. Rev. E*, 78:016704, 2008.
[39] Martin Geier, Martin Schönherr, Andrea Pasquali, and Manfred Krafczyk. The cumulant lattice Boltzmann equation in three dimensions: Theory and validation. *Computers & Mathematics with Applications*, 70:507–547, 2015.
[40] Simon Marie and Denis Ricot. Accuracy of lattice Boltzmann method for aeroacoustic simulations. *Conference: 13th AIAA/CEAS Aeroacoustics Conference (28th AIAA Aeroacoustics Conference)*, 2007.
[41] Jean-Luc Adam, Denis Ricot, Flavien Dubief, and Christine Guy. Aeroacoustic simulation of automotive ventilation outlets. *Journal of the Acoustical Society of America*, 123(5):3250, 2008.
[42] Alois Sengissen, Jean-Christophe Giret, Christophe Coreixas, and Jean-François Boussuge. Simulations of lagoon landing-gear noise using lattice Boltzmann solver. *Conference: AIAA Aviation 21st AIAA/CEAS Aeroacoustics Conference At: Dallas*, 2015.
[43] Sebastien Bocquet, Denis Ricot, Alois Sengissen, Cyril Vincent-Viry, Bruno Demory, Manuel Henner, and Fabrice Ailloud. Evaluation of the lattice Boltzmann method for aero-acoustic simulations of industrial air systems. *Conference: 25th AIAA/CEAS Aeroacoustics Conference*, 2019.
[44] Thomas Astoul. *Towards improved lattice Boltzmann aeroacoustic simulations with non-uniform grids: Application to landing gears noise prediction.* PhD thesis, 6 2021.
[45] Ehab Fares, Michael Wessels, Raoyang Zhang, Chenghai Sun, Nath Gopalaswamy, Peter Roberts, Jamie Hoch, and Hufong Chen. Validation of a lattice-Boltzmann approach for transonic and supersonic flow simulations. *Conference: 52nd Aerospace Sciences Meeting*, 2014.
[46] D. Casalino, A. Hazir, and A. Mann. Turbofan broadband noise prediction using the lattice Boltzmann method. *Conference: 22nd AIAA/CEAS Aeroacoustics Conference*, 2017.
[47] Avinash Jammalamadaka, Gregory M. Laskowski, Yanbing Lia, James Kopriva, Pradeep Gopalakrishnan, Raoyang Zhang, and Hudong Chen. Lattice-Boltzmann very large eddy simulations of fluidic thrust vectoring in a converging/diverging nozzle. *Conference: AIAA Aviation 2020 Forum*, 2020.
[48] Ignacio Gonzalez-Martino and Damiano Casalino. Fan tonal and broadband noise simulations at transonic operating conditions using lattice-Boltzmann methods. *Conference: 2018 AIAA/CEAS Aeroacoustics Conference*, 2018.
[49] Y. Feng, S. Guo, J. Jacob, and P. Sagaut. Grid refinement in the three-dimensional hybrid recursive regularized lattice Boltzmann method for compressible aerodynamics. *Phys. Rev. E*, 101:063302, 2020.
[50] B. Dorschner, F. Bosch, and I.V. Karlin. Particles-on-demand for kinetic theory. *Physical Review Letters*, 12(13):130602, 2019.
[51] Andreas Krämer, Knut Küllmer, Dirk Reith, Wolfgang Joppich, and Holger Foysi. Semi-Lagrangian off-lattice Boltzmann method for weakly compressible flows. *Physical Review E*, 95(2):023305, 2017.
[52] Nikolaos G Kallikounis, Benedikt Dorschner, and IV Karlin. Multiscale semi-Lagrangian lattice Boltzmann method. *Physical Review E*, 103(6):063305, 2021.
[53] Dominik Wilde, Andreas Krämer, Dirk Reith, and Holger Foysi. Semi-Lagrangian lattice Boltzmann method for compressible flows. *Phys. Rev. E*, 101:053306, May 2020.
[54] Dominik Wilde, Andreas Krämer, Dirk Reith, and Holger Foysi. High-order semi-

Lagrangian kinetic scheme for compressible turbulence, 2020.
[55] Takeshi Kataoka and Michihisa Tsutahara. Lattice Boltzmann method for the compressible euler equations. *Phys. Rev. E*, 69:056702, May 2004.
[56] Nianzheng Cao, Shiyi Chen, Shi Jin, and Daniel Martinez. Physical symmetry and lattice symmetry in the lattice Boltzmann method. *Physical Review E*, 55(1):R21, 1997.
[57] Weiping Shi, Wei Shyy, and Renwei Mei. Finite-difference-based lattice Boltzmann method for inviscid compressible flows. *Numerical Heat Transfer: Part B: Fundamentals*, 40(1):1–21, 2001.
[58] Abbas Fakhari and Taehun Lee. Finite-difference lattice Boltzmann method with a block-structured adaptive-mesh-refinement technique. *Physical Review E*, 89(3):033310, 2014.
[59] Haowen Xi, Gongwen Peng, and So-Hsiang Chou. Finite-volume lattice Boltzmann method. *Physical Review E*, 59(5):6202, 1999.
[60] Mohammad Hossein Saadat and Ilya V Karlin. Arbitrary Lagrangian–Eulerian formulation of lattice Boltzmann model for compressible flows on unstructured moving meshes. *Physics of Fluids*, 32(4):046105, 2020.
[61] C Shu, XD Niu, and YT Chew. Taylor-series expansion and least-squares-based lattice Boltzmann method: Two-dimensional formulation and its applications. *Physical Review E*, 65(3):036708, 2002.
[62] Nicolò Frapolli, Shyam S Chikatamarla, and Iliya V Karlin. Lattice kinetic theory in a co-moving Galilean reference frame. *Physical Review Letters*, 117(1):010604, 2016.
[63] Christophe Coreixas and Jonas Latt. Compressible lattice Boltzmann methods with adaptive velocity stencils: An interpolation-free formulation. *Physics of Fluids*, 32(11):116102, 2020.
[64] Chenghai Sun. Lattice-Boltzmann models for high speed flows. *Physical Review E*, 58(6):7283, 1998.
[65] Chenghai Sun. Adaptive lattice Boltzmann model for compressible flows: Viscous and conductive properties. *Physical Review E*, 61(3):2645, 2000.
[66] Vadim Levchenko, Anastasia Perepelkina, Andrey Zakirov, and Boris Korneev. On the conservativity of the particles-on-Demand method for the solution of the discrete Boltzmann equation. *Keldysh Institute Preprints*, 35:19, 2019.
[67] Karlin I.V. Ehsan R., Dorschner B. Particles-on-demand for high Mach number flows. *Presented at DSFD 30*, 2021. https://youtu.be/QKayqbJkKM0.
[68] Florian Renard, Gauthier Wissocq, Jean-François Boussuge, and Pierre Sagaut. A linear stability analysis of compressible hybrid lattice Boltzmann methods. *Journal of Computational Physics*, 446:110649, 2021.
[69] E.Zipunova, A.Perepelkina, A.Zakirov, and S.Khilkov. Regularization and the particles-on-demand method for the solution of the discrete Boltzmann equation. *Journal of Computational Science*, 53:101376, 2021.
[70] E Zipunova, A Perepelkina, and A Zakirov. Applicability of regularized particles-on-demand method to solve Riemann problem. *Journal of Physics: Conference Series*, 1740(1):012024, jan 2021.
[71] Harold Grad. On the kinetic theory of rarefied gases. *Communications on Pure and Applied Mathematics*, 2(4):331–407, 1949.
[72] Xiaowen Shan and Xiaoyi He. Discretization of the velocity space in the solution of the Boltzmann equation. *Phys. Rev. Lett.*, 80:65–68, Jan 1998.
[73] Harold Grad. Note on N-dimensional Hermite polynomials. *Communications on Pure and Applied Mathematics*, 2(4):325–330, 1949.
[74] Prabhu Lal Bhatnagar, Eugene P Gross, and Max Krook. A model for collision processes in

gases. I. Small amplitude processes in charged and neutral one-component systems. *Physical Review*, 94(3):511, 1954.
[75] Yue-Hong Qian, Dominique d'Humières, and Pierre Lallemand. Lattice BGK models for Navier-Stokes equation. *EPL (Europhysics Letters)*, 17(6):479, 1992.
[76] Xuhui Li, Yangyang Shi, and Xiaowen Shan. Temperature-scaled collision process for the high-order lattice Boltzmann model. *Physical Review E*, 100:013301, 2019.
[77] Iliya V. Karlin, Alexander N. Gorban, S. Succi, and V. Boffi. Maximum entropy principle for lattice kinetic equations. *Phys. Rev. Lett.*, 81:6–9, Jul 1998.
[78] Christophe Coreixas, Bastien Chopard, and Jonas Latt. Comprehensive comparison of collision models in the lattice Boltzmann framework: Theoretical investigations. *Physical Review E*, 100(3):033305, 2019.
[79] Arthur H Stroud. *Approximate Calculation of Multiple Integrals*. Prentice-Hall, 1971.
[80] I.P. Mysovskikh. *Interpolating Cubature Formulae*. Nauka, Moscow, 1981. In Russian.
[81] Nico Schlömer. quadPy library. 2021. https://github.com/nschloe/quadpy v0.16.10.
[82] Dominik Wilde, Andreas Krämer, Mario Bedrunka, Dirk Reith, and Holger Foysi. Cubature rules for weakly and fully compressible off-lattice Boltzmann methods. *Journal of Computational Science*, 51:101355, 2021.
[83] SI Konyaev. Ninth-order quadrature formulas invariant with respect to the icosahedral group. In *Doklady Akademii Nauk*, volume 233, pages 784–787. Russian Academy of Sciences, 1977.
[84] Dominic Spiller and Burkhard Dünweg. Semiautomatic construction of lattice Boltzmann models. *Physical Review E*, 101(4):043310, 2020.
[85] B Mond. Multivariate polynomial approximation for equidistant data. *Mathematics of Computation*, 18(86):298–301, 1964.
[86] Karlin I. V. Kallikounis N.G., Dorschner B. DuGKS-in-pond: A finite volume implementation of particles on demand method. *Presented at DSFD 30*, 2021. https://youtu.be/QKayqbJkKM0.
[87] Xiaowen Shan et al. Central-moment-based Galilean-invariant multiple-relaxation-time collision model. *Physical Review E*, 100(4):043308, 2019.
[88] German Zvezdin. ermPoints. 2022. https://github.com/GermanZvezdin/ermPoints v1.0.0.