# A Lattice Boltzmann Model for Rotationally Invariant Dithering

Kai Hagenburg, Michael Breuß, Oliver Vogel, Joachim Weickert, Martin Welk

Mathematical Image Analysis Group
Faculty of Mathematics and Computer Science
Saarland University, Saarbrücken, Germany
{hagenburg,breuss,vogel,weickert,welk}@mia.uni-saarland.de

**Abstract.** In this paper, we present a novel algorithm for dithering of gray-scale images. Our algorithm is based on the lattice Boltzmann method, a well-established and powerful concept known from computational physics. We describe the method and show the consistency of the new scheme to a partial differential equation. In contrast to widely-used error diffusion methods our lattice Boltzmann model is rotationally invariant by construction. In several experiments on real and synthetic images, we show that our algorithm produces clearly superior results to these methods.

## 1 Introduction

Dithering is the problem of binarising a given gray value image such that its visual appearance remains close to the original image. In this paper, we explore a novel approach to this problem by employing a lattice Boltzmann (LB) framework. LB methods are usually used for the simulation of highly complex fluid dynamics, where a discrete environment, the lattice, is provided to model the propagation of gas or fluid particles [9, 12, 15].

**Previous work.** So far, LB methods have not been used extensively in image processing applications. In 1999, Jawerth et al. [7] proposed an LB method to model non-linear diffusion filtering. To our knowledge this is the only published work on LB methods for image processing.

Standard algorithms for dithering employ the technique of error diffusion. Choosing a starting point and sweeping direction, pixels are locally thresholded. The occurring L1-error is then distributed to unprocessed pixels in the neighbourhood according to a specified distribution stencil. This results in a dithered image with an additional blurring. Prominent examples of such algorithms are the ones by Floyd and Steinberg [4], Jarvis et al. [6], Shiau and Fan [13], Stucki [14] and Ostromoukhov [11]. All algorithms follow the same principle, however they only vary in the choice of the distribution stencil. While error diffusion algorithms are very fast, they share undesirable properties. By construction, these algorithms do not consider rotational invariance, resulting in visible sweep directions (see for example Figure 3). Furthermore, error diffusion methods introduce undesired noisy, worm-like artifacts which are prominent to a greater

or lesser extent depending on the distribution stencil. In one recent paper it is proposed to use simulated annealing for solving an optimisation problem related to dithering. Though this produces visually convincing results, the method is rather slow, depends on several parameters and needs an already dithered image (e.g. by error diffusion methods) as initialisation [10].

**Our contribution.** The goal of our paper is to present a novel dithering algorithm that does not suffer from problems with respect to rotationally invariance, local blurring and directional bias introduced by distribution stencils. Its favorable visual quality results from its edge-enhancing properties. All this can be achieved by choosing lattice Boltzmann strategies in an appropriate way.

**Organization of the paper.** The paper is organized as follows. In Section 2 we describe our LB framework, followed by the definition of the needed reference state in Section 3. In Section 4 we summarize the algorithm. Experiments are presented in Section 5 and the paper is concluded with a summary in Section 6. In the appendix, we provide a proof for the consistency of our method.

## 2 Our Lattice Boltzmann Framework

At the heart of the LB method, one distinguishes a *macroscopic* level and a *microscopic* level. Using this as a framework, the underlying idea behind the scheme is quite intuitive from a physical point of view. The idea is that the state we observe by the gray values in an image is a macroscopic state. The gray value density can be understood as an analogon to the density of a fluid. Knowing that any fluid is naturally a composition of very small molecules, we can explore that analogy by the following idea: If one would zoom close enough into the pixels of an image one would observe that the gray values are represented by an appropriate amount of white particles. These particles constitute the microscopic state. By movement and collision of the particles, the observable macroscopic state may change.

The LB method requires a set of rules for movement and collision of the microscopic particles. After evaluating the microscopic dynamics, the macroscopic gray values are obtained by summation over the discrete particle distribution. In what follows, we explain the corresponding steps in detail.

**The microscopic set-up.** The LB method relies on a discrete grid, or lattice. Each node of the lattice holds the value of a distribution function $u_\alpha$, where $\alpha$ is an index that indicates the neighbourhood relation to the center node. The position of neighbours is identified by a lattice vector $\boldsymbol{e}_\alpha$, where $\boldsymbol{e}_0 = (0,0)^\top$ points to the center node itself. In this paper we employ a $(3\times3)$-stencil giving the set of possible directions $\Lambda_d := \{-1,0,1\} \times \{-1,0,1\}$. For $\alpha \in \{0,\dots,8\} := \Lambda$ indicating all possible directions in $\Lambda_d$, the corresponding directional unit vector is $\boldsymbol{e}_\alpha = (e_{\alpha_1}, e_{\alpha_2})^\top$.

The distribution function $u_\alpha$ models a microscopic state. The macroscopic state, in our case the gray value at position $\boldsymbol{x} = (x_1, x_2)^\top$ at time $t$, is described by summation over the local $(3 \times 3)$-patch:

$$u(\boldsymbol{x}, t) = \textstyle\sum_{\alpha\in\Lambda} u_\alpha(\boldsymbol{x}, t). \tag{1}$$

As indicated, the LB method encodes particle movement and collisions that take place at the microscopic level. The corresponding *fundamental equation* reads as

$$u_\alpha(\boldsymbol{x}+\boldsymbol{e}_\alpha, t+1) = u_\alpha(\boldsymbol{x}, t) + \Omega_\alpha(\boldsymbol{x}, t)\,, \tag{2}$$

where $\Omega_\alpha(\boldsymbol{x}, t)$ is the so-called *collision operator*. The proper modelling of this operator is vital for any LB algorithm as it describes a set of collision rules that can be used to simulate arbitrary fluid models.

In a first step to address this issue, we employ a BGK model named after Bhatnagar, Gross and Krook [1, 12] which has become a standard approach in the LB literature. The specific BGK model we use for $\Omega_\alpha$ reads as

$$\Omega_\alpha = u_\alpha^{\mathrm{ref}} - u_\alpha\,. \tag{3}$$

This model allows to interpret a collision state as the deviation of the current microscopic distribution $u_\alpha$ from a *reference distribution* $u_\alpha^{\mathrm{ref}}$ instead of explicitly defining collision rules. In a second step, we impose as an additional structural property the *conservation of the average gray value* of the image via the pointwise condition

$$\textstyle\sum_{\alpha\in\Lambda} \Omega_\alpha(\boldsymbol{x}, t) = 0\,. \tag{4}$$

We assume now that the lattice parameters $h$ and $\tau$ that denote the spatial and temporal mesh widths are coupled via a relation $\tau/h^2 = \text{constant}$. Employing then a *scaling* in space and time by the scaling parameter $\varepsilon$, one obtains by (2)-(3) the relation

$$u_\alpha(\boldsymbol{x}+\varepsilon\boldsymbol{e}_\alpha, t+\varepsilon^2) = u_\alpha^{\mathrm{ref}}(\boldsymbol{x}, t)\,. \tag{5}$$

In order to define the LB method, we approximate $u_\alpha^{\mathrm{ref}}(\boldsymbol{x}, t)$ by

$$u_\alpha^{\mathrm{ref}}(\boldsymbol{x}, t) \;=\; t_\alpha u(\boldsymbol{x}, t)\,(1+\varepsilon\gamma_\alpha)\;. \tag{6}$$

In case of $\gamma_\alpha = 0$, equation (6) would give a LB description for linear diffusion [15]. By setting $\gamma_\alpha \neq 0$, the reference state can be described as a perturbation of an equilibrium distribution by some function $\gamma_\alpha$ which is crucial to achieve the dithering effect. In the following chapters we will directly give the reference state, as the direct description of $\gamma_\alpha$ can be derived from that. The $t_\alpha$ are normalisation factors depending on the direction [12]:

$$t_\alpha = \begin{cases} 4/9\,, & \boldsymbol{e}_\alpha = (0,0)\;, \\ 1/9\,, & \boldsymbol{e}_\alpha = (0,\pm1),(\pm1,0)\;, \\ 1/36\,, & \boldsymbol{e}_\alpha = (\pm1,\pm1)\;. \end{cases} \tag{7}$$

As usual for normalisation weights, $\sum_{\alpha\in\Lambda} t_\alpha = 1$.

The crucial point about (6) is that it relates the current macroscopic state $u(\boldsymbol{x}, t)$ to $u_\alpha^{\mathrm{ref}}(\boldsymbol{x}, t)$ via the introduced perturbation. The logic behind the scheme definition given as the next step is to model $u_\alpha^{\mathrm{ref}}$ in such a way that it gives the desired steady state – i.e. the dithered image – by evolution in time.

**Macroscopic limit.** The proof of the following assertion is given in the Appendix:

**Theorem 1.** *In the scaling limit $\varepsilon \to 0$, the LB scheme obeying the proposed discrete set-up solves the partial differential equation*

$$\frac{\partial}{\partial t} u(\boldsymbol{x}, t) = \frac{1}{6} \Delta u(\boldsymbol{x}, t) - \operatorname{div} \left( u(\boldsymbol{x}, t) \boldsymbol{\gamma}(\boldsymbol{x}, t) \right) , \tag{8}$$

*where $\Delta := \partial^2/\partial x_1^2 + \partial^2/\partial x_2^2$ is the Laplace operator, and where the divergence operator is denoted by* $\operatorname{div}\left((a_1, a_2)^\top\right) := \partial a_1/\partial x_1 + \partial a_2/\partial x_2$ *and a vector valued function $\boldsymbol{\gamma}(\boldsymbol{x}, t)$ given by*

$$\boldsymbol{\gamma}(\boldsymbol{x}, t) := \textstyle\sum_{\alpha \in \Lambda} \boldsymbol{e}_\alpha t_\alpha \gamma_\alpha . \tag{9}$$

The PDE (8) is a diffusion-advection equation. While the diffusion term $\Delta u$ gives an uniform spreading of the macroscopic variable $u$, this is balanced by the edge-enhancing advection term $\operatorname{div}(u\boldsymbol{\gamma})$.

## 3 Constructing a Reference State for Dithering

We now model the reference state, see especially (6). Our aim is a dithering strategy that preserves structures and enhances edges by the model.

Structure enhancement can be achieved by enlarging the gradient between two neighbouring pixels. This is done by transporting particles from darker pixels to brighter pixels. In the following, we consider three possible scenarios.

1. We distinguish two cases. If a pixel in $(\boldsymbol{x}, t)$ has a larger gray value than its neighbour in $(\boldsymbol{x} + \mathbf{e}_\alpha, t)$, then we do not want to allow particles to dissipate into direction $\mathbf{e}_\alpha$. In the opposite case, we allow the neighbouring pixel in $(\boldsymbol{x} + \mathbf{e}_\alpha, t)$ to take into account – and take away – the amount $t_\alpha u(\boldsymbol{x}, t)$ particles, as long as the neighbour is not already saturated.
2. For the robustness of the implementation, we also define the following rule. If a pixel in $(\boldsymbol{x}, t)$ has a very low gray value below a minimal threshold $\nu > 0$ close to zero, we always allow a neighbouring pixel in $(\boldsymbol{x}+\mathbf{e}_\alpha, t)$ to take away an amount $t_\alpha u(\boldsymbol{x}, t)$ of particles.
3. If the gray value exceeds 255 we distribute superficial particles to neighbouring pixels.

Summarising these considerations, we obtain the reference state as

$$u_\alpha^{\mathrm{ref}}(\boldsymbol{x} + \boldsymbol{e_\alpha}, t) = \begin{cases} t_\alpha u(\boldsymbol{x}, t) & \text{if } u(\boldsymbol{x} + \boldsymbol{e}_\alpha, t) > u(\boldsymbol{x}, t) \\ & \text{and } u(\boldsymbol{x} + \boldsymbol{e}_\alpha, t) < 255 \\ 0 & \text{if } u(\boldsymbol{x} + \boldsymbol{e}_\alpha, t) < u(\boldsymbol{x}, t) \\ t_\alpha u(\boldsymbol{x}, t) & \text{if } u(\boldsymbol{x}, t) < \nu \text{ and } \alpha \neq (0,0) \\ 0 & \text{if } u(\boldsymbol{x}, t) < \nu \text{ and } \alpha = (0,0) \\ t_\alpha (u(\boldsymbol{x}, t) - 255) & \text{if } u(\boldsymbol{x}, t) > 255 \text{ and } \alpha \neq (0,0) \\ 255 & \text{if } u(\boldsymbol{x}, t) > 255 \text{ and } \alpha = (0,0) \end{cases} \tag{10}$$

with $t_\alpha$ as in (7). Furthermore, we disallow any flow across the image boundaries. This suffices as *boundary conditions* for the reference state.

## 4 The Algorithm

We now show how to code an iterative dithering algorithm making use of the equations (5) and (10). By (5), and setting the scaling parameter to match the grid, we obtain $u_\alpha(\boldsymbol{x}, t+1) = u_\alpha^{\mathrm{ref}}(\boldsymbol{x} - \boldsymbol{e}_\alpha, t)$ and after taking sums:

$$\textstyle\sum_\alpha u_\alpha(\boldsymbol{x}, t+1) = \sum_\alpha u_\alpha^{\mathrm{ref}}(\boldsymbol{x} - \boldsymbol{e}_\alpha, t). \tag{11}$$

By the symmetries incorporated in the directions in $\Lambda_d$ and by (1) follows

$$u(\boldsymbol{x}, t+1) := \textstyle\sum_\alpha u_\alpha^{\mathrm{ref}}(\boldsymbol{x} + \boldsymbol{e}_\alpha, t). \tag{12}$$

With this knowledge, we can describe the algorithm.

**Summary of the algorithm**
**Step 1:** Compute the reference state according to (10)
**Step 2:** Compute $u(\boldsymbol{x}, t+1) := \sum_\alpha u_\alpha^{\mathrm{ref}}(\boldsymbol{x} + \boldsymbol{e}_\alpha, t)$
**Step 3:** If $\|u(\cdot, t+1) - u(\cdot, t)\|_2 < \epsilon$ break, otherwise go back to 1.

**Implementation details.** W.l.o.g. we consider images that already have grey values that sum up to multiples of 255, otherwise we scale the image such that it fulfills this property. In this setting, we can say that our algorithm is grey-value preserving. However, it is possible that at the end of the evolution a few pixels converge to a state neither zero nor 255. On these pixels, we perform a gray-value-preserving threshold to obtain the dithered image.

Furthermore, the $t_\alpha$ as set in (7) are need to be re-normalised, if for some $\alpha$ the microscopic state $u_\alpha$ is zero. The reason is easily seen by considering an example where all particles are concentrated at $\alpha = 0$: Summing up directly with the weights (7) effectively reduces the local gray value by 5/9. Thus, in the algorithm one defines a number $\eta_\alpha$ which is zero for $u_\alpha = 0$ and one otherwise. Then we renormalise the weights $t_\alpha$ via a factor $\eta$ such that $\eta \sum_\alpha t_\alpha \eta_\alpha = 1$; in case the sum is zero we set $\eta$ to some finite number.

## 5 Experiments

In this section we present experiments on both real and synthetic images that show the quality of our approach. Especially, we demonstrate the *edge-enhancing* and *rotationally invariant* properties of our algorithm. We compare the visual quality of the results to the classical standard method of Floyd and Steinberg [4] implemented with serpentine pixel order as this is the essential algorithm mostly identified with error diffusion, as well as to the method of Ostromoukhov [11] which constitutes the state-of-the art error diffusion algorithm in the field. For Ostromoukhov's method we use the original implementation from the author's web page [1].

**The Poker chip experiment.** In the first experiment we deal with a real world image with large contrasts, see Figure 1. While error diffusion methods

[1] http://www.iro.umontreal.ca/ ostrom/varcoeffED/

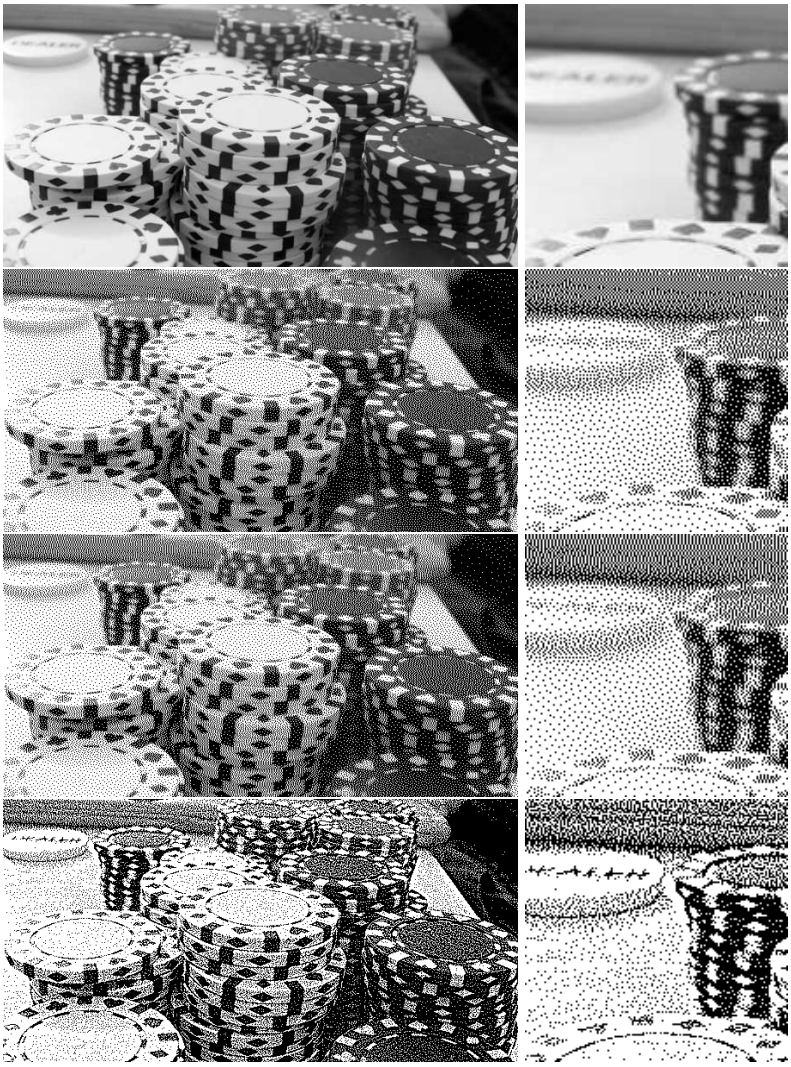

**Fig. 1.** Results of dithering algorithms. **Left**: Result of the algorithms on a real-world image of size $600 \times 305$. **Right**: close-up of the upper left corner with size $150 \times 150$. **First row**: Original image. **Second row**: Floyd-Steinberg with serpentine implementation. **Third row**: Ostromoukhov. **Fourth row**: Lattice Boltzmann dithering.

blur important image structures and introduce noisy patterns, the lattice Boltzmann method preserves edges very well. Our method even recovers prominent structures of blurred objects that are out-of-focus in the original image. Comparing the methods of Floyd-Steinberg and Ostromoukhov, we find no significant visual difference between each other. Let us note that the iteration strategy relying on the pixel ordering is the same in both error diffusion methods.

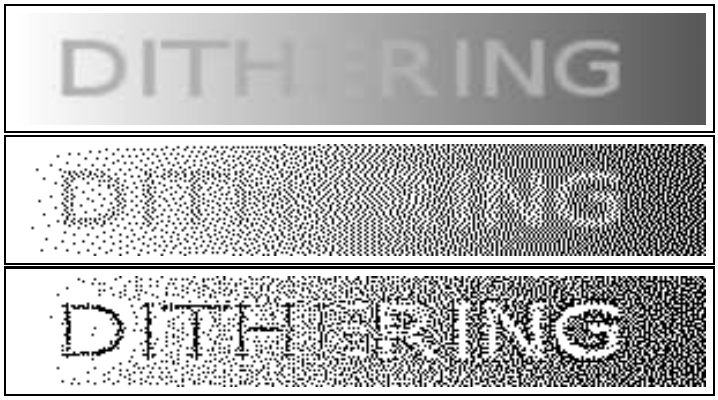

**Fig. 2.** Comparison of dithering algorithms on images with low contrast areas. **Top.** Original image. Gray value ramp gradually with low-contrast text with constant gray value, size $300 \times 50$. **Middle.** Ostromoukhov. **Bottom.** Lattice Boltzmann dithering.

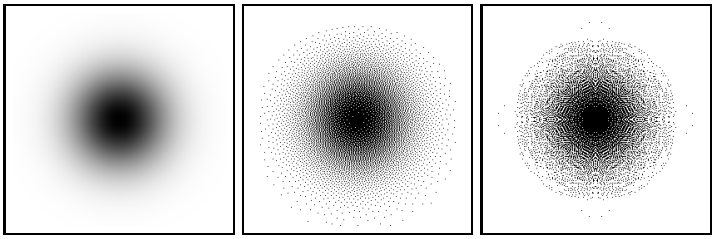

**Fig. 3.** Evaluation w.r.t. rotation invariance. **Left**: Gaussian with size $256 \times 256$. **Middle**: Ostromoukhov. **Right**: Lattice Boltzmann dithering.

**The ramp experiment.** We now consider a synthetic image of low contrast, see Figure 2. The image shows a ramp of gray values decreasing from left to right, together with a text of constant gray value. The latter has been chosen in such a way that parts of the text are indistinguishable from the background ramp. The error diffusion result loses some letters during the dithering process since they tend to smooth out image structures with low contrast. In contrast, our method still produces a readable text.

**The Gaussian test.** In Figure 3 we demonstrate the rotational invariance of our scheme, though some visible directional artifacts remain due to the chosen discretisation. Furthermore, it is observable that the result of a error diffusion method strongly depends on the implementation of the pixel ordering.

**Runtimes.** While the runtime of the error diffusion algorithms lies in the range of milliseconds, our diffusion-advection motivated algorithm takes a couple of seconds to converge on a standard PC with an implementation in C. The inherent parallelisation potential of lattice Boltzmann methods [3] that would allow for a further speedup has not been exploited yet. In its current state, the algorithm is attractive for offline dithering in high quality.

## 6 Conclusion and Future Work

We have derived a novel lattice Boltzmann model for dithering images that is by construction rotationally invariant. The adaptation of the lattice Boltzmann framework to this application has been achieved by specifying an appropriate reference state within the collision operator. We have provided an analysis of the model that shows that its macroscopic equation is a diffusion-advection equation.

For future work, we plan to perform research on efficient algorithms for our LB method and to exploit its excellent parallelisation properties. We also plan to analyse the PDE (8) more thoroughly and eventually extend our algorithm to colour images.

**Acknowledgements.** The authors gratefully acknowledge the funding given by the Deutsche Forschungsgemeinschaft (DFG).

## References

1. P. L. Bhatnagar, E. P. Gross, M. Krook. A model for collision processes in gases. I. Small amplitude procession charged and neutral one-component systems. Physical Review, 94 (3), 511–525 (1954)
2. S. Chapman, T. G. Cowling. The Mathematical Theory of Non-uniform Gases. University Press, Cambridge (1939)
3. S. P. Dawson, S. Chen, G. D. Doolen. Lattice Boltzmann computations for reaction-diffusion equations. Journal of Chemical Physics, 98 (2), 1514–1523 (1993)
4. R. W. Floyd, L. Steinberg. An adaptive algorithm for spatial gray scale. Proceedings of the Society of Information Display 17, 75–77 (1976)
5. U. Frisch, B. Hasslacher, Y. Pomeau. Lattice-gas automata for Navier-Stokes equations. Phys. Rev. Letters, 56, 1505–1508 (1986)
6. J. F. Jarvis, C. N. Judice, W. H. Ninke. A survey of techniques for the display of continuous tone pictures on bilevel displays. Computer Graphics and Image Processing, 5 (1), 13–40 (1976)
7. B. Jawerth, P. Lin, E. Sinzinger. Lattice Boltzmann models for anisotropic diffusion of images. Journal of Mathematical Imaging and Vision 11, 231–237 (1999)
8. J. F. Lutsko. Chapman-Enskog expansion about nonequilibrium states with application to the sheared granular fluid. Physical Review E, 73, 021302 (2006)
9. G. R. McNamera, G. Zanetti. Use of the lattice Boltzmann equation to simulate lattice-gas automata. Physical Review Letters, 61, 2332–2335 (1988)
10. W.-M. Pang, Y. Qu, T.-T. Wong, D. Cohen-Or, P.-A. Heng. Structure-Aware Halftoning. ACM Transactions on Graphics (SIGGRAPH 2008 issue), 27 (3), 89:1–89:8 (2008)
11. V. Ostromoukhov. A Simple and Efficient Error-Diffusion Algorithm. In: Proceedings of SIGGRAPH 2001, in ACM Computer Graphics, 567–572 (2001)
12. Y. H. Qian, D. D'Humieres, P. Lallemand. Lattice BGK models for Navier-Stokes equation. Europhysics Letters, 17 (6), 479–484 (1992)
13. J. N. Shiau, Z. Fan A set of easily implementable coefficients in error diffusion with reduced worm artifacts Proc. SPIE, Vol. 2658, 222–225 (1996)
14. P. Stucki, MECCA–A Multiple-Error Correction Computation Algorithm for Bi-Level Image Hardcopy Reproduction Research Report RZ-1060 (1981), IBM Research Laboratory, Zurich, Switzerland.
15. D. Wolf-Gladrow. Lattice Gas Cellular Automata and Lattice Boltzmann Models - An Introduction. Springer, Berlin (2000)

## Appendix: Proof of Theorem 1

The proof proceeds in the following way. As we aim at deriving a PDE, we want to obtain expressions of $u(\boldsymbol{x},t)$. In a first step of the proof, we therefore eliminate all dependencies on shifted variables $(\boldsymbol{x}+\varepsilon\boldsymbol{e}_\alpha, t+\varepsilon^2)$. In a second step, we eliminate the reference distribution $u_\alpha^{\mathrm{ref}}$ from the deduced equations. In the final step, we summarise the microscopic variables appropriately to obtain expressions in the macroscopic variable $u(\boldsymbol{x},t)$.

We begin with substituting $u_\alpha(\boldsymbol{x}+\varepsilon\boldsymbol{e}_\alpha, t+\varepsilon^2)$ from the left hand side of equation (5). This is done making use of a Taylor expansion around $(\boldsymbol{x},t)$:

$$u_\alpha(\boldsymbol{x}+\varepsilon\boldsymbol{e}_\alpha, t+\varepsilon^2) \;=\; u_\alpha(\boldsymbol{x},t) + \varepsilon \textstyle\sum_{i=1}^2 e_{\alpha,i}\frac{\partial}{\partial x_i}u_\alpha(\boldsymbol{x},t) + \mathcal{O}(\varepsilon^2)\,. \tag{13}$$

Substituting this expression in (5) and neglecting the second order error gives

$$\varepsilon \textstyle\sum_{i=1}^2 e_{\alpha,i}\frac{\partial}{\partial x_i}u_\alpha(\boldsymbol{x},t) \;=\; u_\alpha^{\mathrm{ref}}(\boldsymbol{x},t) - u_\alpha(\boldsymbol{x},t)\,. \tag{14}$$

For the second step of the proof, we use the *Chapman-Enskog expansion* [2]. This works in analogy to the Taylor expansion, and describes $u_\alpha$ in terms of fluctuations about the reference state $u_\alpha^{\mathrm{ref}}$ that are given by a function $\Phi_\alpha$:

$$u_\alpha \;=\; u_\alpha^{\mathrm{ref}} + \varepsilon\Phi_\alpha + \mathcal{O}(\varepsilon^2)\,. \tag{15}$$

The actual choice of the reference state is not crucial, cf. [8] where arbitrary reference states are used. Substituting $u_\alpha(\boldsymbol{x},t)$ in (14) by (15) gives

$$\Phi_\alpha = -\textstyle\sum_{i=1}^2 e_{\alpha,i}\frac{\partial}{\partial x_i}u_\alpha(\boldsymbol{x},t) + \mathcal{O}(\varepsilon)\,. \tag{16}$$

Having thus computed an expression for the fluctuation $\Phi_\alpha$, we plug this into the Chapman-Enskog expansion (15):

$$u_\alpha \;=\; u_\alpha^{\mathrm{ref}} - \varepsilon \textstyle\sum_{i=1}^2 e_{\alpha,i}\frac{\partial}{\partial x_i}u_\alpha(\boldsymbol{x},t) + \mathcal{O}(\varepsilon^2)\,. \tag{17}$$

We proceed by considering the collision rule (4). Using (3) one obtains

$$\textstyle\sum_{\alpha\in\Lambda}\big(u_\alpha(\boldsymbol{x}+\varepsilon\boldsymbol{e}_\alpha, t+\varepsilon^2) - u_\alpha(\boldsymbol{x},t)\big) \;=\; 0\,. \tag{18}$$

The Taylor approximation of $u_\alpha(\boldsymbol{x}+\varepsilon\boldsymbol{e}_\alpha, t+1)$ reads as

$$\begin{aligned} u_\alpha(\boldsymbol{x}+\varepsilon\boldsymbol{e}_\alpha, t+1) = u_\alpha(\boldsymbol{x},t) + \textstyle\sum_{\alpha\in\Lambda}\Big[\varepsilon^2\frac{\partial}{\partial t}u_\alpha(\boldsymbol{x},t) + \sum_{i=1}^2 \varepsilon e_{\alpha,i}\frac{\partial}{\partial x_i}u_\alpha(\boldsymbol{x},t) \\ + \frac{\varepsilon^2}{2}\textstyle\sum_{i,j=1}^2 e_{\alpha,i}e_{\alpha,j}\frac{\partial^2}{\partial x_i\partial x_j}u_\alpha(\boldsymbol{x},t)\Big]\,. \end{aligned} \tag{19}$$

Inserting this expression for $u_\alpha(\boldsymbol{x}+\varepsilon\boldsymbol{e}_\alpha, t+\varepsilon^2)$ in (18) gives

$$\begin{aligned} 0 \;=\; \underbrace{\varepsilon^2\textstyle\sum_{\alpha\in\Lambda}\frac{\partial}{\partial t}u_\alpha(\boldsymbol{x},t)}_{=:A} + \underbrace{\varepsilon\textstyle\sum_{\alpha\in\Lambda}\sum_{i=1}^2 e_{\alpha,i}\frac{\partial}{\partial x_i}u_\alpha(\boldsymbol{x},t)}_{=:B} \\ + \underbrace{\tfrac{\varepsilon^2}{2}\textstyle\sum_{\alpha\in\Lambda}\sum_{i,j=1}^2 e_{\alpha,i}e_{\alpha,j}\frac{\partial^2}{\partial x_i\partial x_j}u_\alpha(\boldsymbol{x},t)}_{=:C}\,. \end{aligned} \tag{20}$$

We now rewrite the terms $A$, $B$ and $C$ individually.
*Term A.*

$$\varepsilon^2 \sum_{\alpha\in\Lambda} \tfrac{\partial}{\partial t} u_\alpha(\boldsymbol{x},t) \;=\; \varepsilon^2 \tfrac{\partial}{\partial t} \sum_{\alpha\in\Lambda} u_\alpha(\boldsymbol{x},t) \overset{(1)}{=} \varepsilon^2 \tfrac{\partial}{\partial t} u(\boldsymbol{x},t)\,. \tag{21}$$

*Term B.*

$$\begin{aligned}\varepsilon \sum_{\alpha\in\Lambda} \sum_{i=1}^{2} e_{\alpha,i} \tfrac{\partial}{\partial x_i} u_\alpha(\boldsymbol{x},t) &\overset{(17)}{=} \varepsilon \sum_{\alpha\in\Lambda} \sum_{i=1}^{2} e_{\alpha,i} \tfrac{\partial}{\partial x_i} u_\alpha^{\mathrm{ref}} \\ &\quad - \;\varepsilon^2 \sum_{\alpha\in\Lambda} \sum_{i=1}^{2} e_{\alpha,i} \tfrac{\partial}{\partial x_i} \Big[\sum_{j=1}^{2} e_{\alpha,j} \tfrac{\partial}{\partial x_j} u_\alpha(\boldsymbol{x},t)\Big]\end{aligned} \tag{22}$$

We consider the first summand in (23). For replacing $u_\alpha^{\mathrm{ref}}$ in this term, we make use of assumption (6), yielding

$$\begin{aligned}&\varepsilon \sum_{\alpha\in\Lambda} \sum_{i=1}^{2} e_{\alpha,i} \tfrac{\partial}{\partial x_i} \left[t_\alpha u(\boldsymbol{x},t)\,(1+\varepsilon\gamma_\alpha)\right] \\ &= \varepsilon \sum_{i=1}^{2} \tfrac{\partial}{\partial x_i} \left[u(\boldsymbol{x},t) \sum_{\alpha\in\Lambda} e_{\alpha,i} t_\alpha\right] + \varepsilon^2 \sum_{i=1}^{2} \tfrac{\partial}{\partial x_i} \left[u(\boldsymbol{x},t) \sum_{\alpha\in\Lambda} e_{\alpha,i} t_\alpha \gamma_\alpha\right]\end{aligned} \tag{23}$$

By $\sum_{\alpha\in\Lambda} e_{\alpha,i} t_\alpha = 0$ and (9), the result is

$$\varepsilon \sum_{\alpha\in\Lambda} \sum_{i=1}^{2} e_{\alpha,i} \tfrac{\partial}{\partial x_i} \left[t_\alpha u(\boldsymbol{x},t)\,(1+\varepsilon\gamma_\alpha)\right] \;=\; \varepsilon^2 \sum_{i=1}^{2} \tfrac{\partial}{\partial x_i} \left[u(\boldsymbol{x},t)\boldsymbol{\gamma}(\boldsymbol{x},t)\right]\,. \tag{24}$$

We now employ

$$u_\alpha \overset{(15)}{=} u_\alpha^{\mathrm{ref}} + \mathcal{O}(\varepsilon) \overset{(6)}{=} t_\alpha u(\boldsymbol{x},t) + \mathcal{O}(\varepsilon)\,, \tag{25}$$

and by plugging it into the second summand of (23) gives

$$\begin{aligned}&\varepsilon^2 \sum_{\alpha\in\Lambda} \sum_{i=1}^{2} e_{\alpha,i} \tfrac{\partial}{\partial x_i} \Big[\sum_{j=1}^{2} e_{\alpha,j} \tfrac{\partial}{\partial x_j} t_\alpha u(\boldsymbol{x},t)\Big] \\ &= \varepsilon^2 \sum_{i,j=1}^{2} \underbrace{\sum_{\alpha\in\Lambda} e_{\alpha,i} e_{\alpha,j} t_\alpha}_{=1/3\cdot\delta_{ij}} \tfrac{\partial^2}{\partial x_i \partial x_j} u(\boldsymbol{x},t) \;=\; \tfrac{\varepsilon^2}{3} \sum_{i=1}^{2} \tfrac{\partial^2}{\partial x_i^2} \underbrace{\sum_{\alpha\in\Lambda} u_\alpha(\boldsymbol{x},t)}_{=u(\boldsymbol{x},t) \text{ by } (1)}\,.\end{aligned} \tag{26}$$

In summary, Term $B$ (23) results in

$$\varepsilon \sum_{\alpha\in\Lambda} \sum_{i=1}^{2} e_{\alpha,i} \tfrac{\partial}{\partial x_i} u_\alpha(\boldsymbol{x},t) \;=\; \varepsilon^2 \mathrm{div}\big(u(\boldsymbol{x},t)\boldsymbol{\gamma}(\boldsymbol{x},t)\big) - \tfrac{\varepsilon^2}{3} \Delta u(\boldsymbol{x},t)\,. \tag{27}$$

*Term C.* In a first step, we substitute $u_\alpha$ as in (25), neglecting higher order terms in $\varepsilon$, giving us a first-order approximation of $u_\alpha$. Using this gives

$$\begin{aligned}&\frac{\varepsilon^2}{2} \sum_{\alpha\in\Lambda} \sum_{i,j=1}^{2} e_{\alpha,i} e_{\alpha,j} \tfrac{\partial^2}{\partial x_i \partial x_j} t_\alpha u(\boldsymbol{x},t)) \\ &= \frac{\varepsilon^2}{2} \sum_{i,j=1}^{2} \tfrac{\partial^2}{\partial x_i \partial x_j} u(\boldsymbol{x},t) \underbrace{\sum_{\alpha\in\Lambda} e_{\alpha,i} e_{\alpha,j} t_\alpha}_{=1/3\cdot\delta_{ij}} \;=\; \tfrac{\varepsilon^2}{6} \Delta u(\boldsymbol{x},t)\,.\end{aligned} \tag{28}$$

Plugging all the three terms $A$, $B$, and $C$ together, dividing by $\varepsilon^2$ and taking the limit $\varepsilon \to 0$ results in the diffusion-advection equation (8) which concludes our proof.