Proceedings of ALGORITMY 2020
pp. 111–120

NUMERICAL INVESTIGATION OF THE DISCRETE SOLUTION OF PHASE-FIELD EQUATION

PAVEL EICHLER*, MICHAL MALÍK*, TOMÁŠ OBERHUBER*, AND RADEK FUČÍK*

Abstract. In this article, we deal with the numerical solution of the phase-field equation. The numerical solution is based on the lattice Boltzmann method compared with the finite difference method. First, a short introduction to the mathematical and numerical model is presented and the implementation of two different methods is briefly investigated. Then, the results of both methods are compared and the experimental order of convergence is determined. One of the significant drawbacks of the finite difference method is that a sufficiently fine computational mesh is crucial for accurate results. The advantage of the lattice Boltzmann method is that it produces accurate results on courser meshes.

Key words. phase-field equation, finite difference method, lattice Boltzmann method, numerical study

AMS subject classifications. 76T99, 68U20, 65Y05

1. Introduction. Multiphase flow is one of the most investigated fields in computation fluid dynamics because it is present in various phenomena in industry. These phenomena include, for example, combustion in fluidized bed reactors, blood flow in vessels, water evaporation from plants, etc.

There are different methods for the solution of multiphase flow [1]. One of these methods uses a phase parameter ϕ $[-]$, which describes the boundary between different phases [2]. This method is mainly used in solidification problems [3], various fluid flow problems [4], etc. There are two different physical theories in the solidification physics. The Gibb's theory [5] assumes sharp interfaces between different phases. On the contrary, the Van der Waals theory [6] assumes boundary with nonzero thickness W $[m]$ and smooth transition of the parameter ϕ on the boundary. It can be shown by the asymptotic analysis that these two methods are asymptotically equivalent as $W \to 0^+$. In this work, we assume that $W > 0$.

There are various numerical methods for the study of the boundary evolution in time, e.g., the volume of fluid method [7], the level-set method [8], the phase-field method [9], the color gradient method [10], the Shan-Chen method [11], the free-energy method [12, 13], etc. In this work, the phase-field method is discussed.

Next, we assume that the evolution of the phase parameter ϕ is determined by the modified Allen-Cahn equation [14]. Two different numerical methods are used for the discrete solution of this equation.

The first numerical method is the lattice Boltzmann method (LBM) [15]. This method was originally derived from cellular automaton for the solution of the Navier-Stokes equation. Contrary to the classical numerical method, LBM uses probability density function as the unknown variables. There are several submethods of LBM such as single relaxation time LBM [16], multi relaxation time LBM [17], cascaded LBM [18], cumulant LBM [19], entropic LBM [20], etc. It was shown, that LBM can be used for the solution of the advection equation [16], the shallow water equation

*Department of Mathematics, Faculty of Nuclears Sciences and Physical Engineering, Czech Technical University in Pragu, Trojanova 13, 120 00 Praha 2.

[21], or the phase-field equation [22, 23]. In this work, LBM form for the phase-field equation [22] is used.

The second numerical method is the finite difference method (FDM), which is used as a reference numerical method.

The main aim of this article is to compare LBM and FDM for the solution of the modified Allen-Cahn equation.

This work is organized as follows. The Section 2 describes the mathematical model. Next, the discrete numerical methods are summarized. In the fourth section, numerical results and discussion is presented. The last section concludes our findings.

2. Mathematical model. The mathematical model is based on the Van der Waals theory, which assumes smooth transition of the phase parameter ϕ within the phase interface of thickness W. It is assumed that ϕ is within the range of $[-0.5, 0.5]$ and the interface is represented as the set $\Gamma = \{\vec{x} \in \Omega \,|\, \phi(\vec{x}) = 0\}$, where Ω is the computational domain.

The computational domain is a square, $\Omega = (0,1) \times (0,1)$ with dimensions of m. Next, the solution is investigated in the time interval $(0, T_{fin})$, T_{fin} $[s]$. The evolution of the phase parameter $\phi(\vec{x}, t)$, $\forall \vec{x} \in \Omega$, $t \in (0, T_{fin})$ is determined by the modified Allen-Cahn equation [22]

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\vec{u}\,\phi) = \nabla \cdot \left[M \left(\nabla \phi - \frac{\nabla \phi}{\|\nabla \phi\|} \frac{1 - 4\phi^2}{W} \right) \right], \tag{2.1}$$

where t [s] is the time, $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right)^T$, M $[\mathrm{m}^2\,\mathrm{s}^{-1}]$ is the mobility and W $[m]$ is the boundary thickness. Eq. (2.1) is supplemented by the periodic boundary conditions on $\partial\Omega$.

The initial condition for ϕ at $t = 0$ is given by

$$\phi(\vec{x}, 0) = \phi^0(\vec{x}) = \frac{1}{2} \tanh \left[\frac{2\xi_n(\vec{x})}{W} \right], \tag{2.2}$$

where ξ_n is the normal distance to the interface $\Gamma(0)$. The hyperbolic tangent profile is chosen to smoothly approximate the jump of ϕ at the interface.

3. Discrete mathematical model. The computational domain Ω is discretized using regular lattice $\hat{\Omega}$, see Fig. 3.1,

$$\hat{\Omega} = \Big\{ \vec{x}_{i\,j} = \left(\left(i + \frac{1}{2} \right) \Delta_\ell, \left(j + \frac{1}{2} \right) \Delta_\ell \right) \Big| \, i \in \{1, 2, \cdots, N_x - 2\}, \; j \in \{1, 2, \cdots, N_y - 2\} \Big\} \tag{3.1}$$

where N_x and N_y are the numbers of discrete lattice sites in x and y direction, respectively. For simplicity, $N_x = N_y$ and $\Delta_\ell = {}^1\!/_{N_x}$.

The time interval $(0, T_{fin})$ is equidistantly discretized as $\{t_n | n \in \{0, 1, \cdots, N_t\}\}$, where $t_n = \Delta_t n$, $\Delta_t = {}^{T_{fin}}\!/_{N_t}$, and $N_t + 1$ is the number of time steps.

3.1. Lattice Boltzmann method. The lattice Boltzmann method is a numerical method originally designed for the fluid dynamic. The primary unknowns of the method are the non-dimensional particle density functions f_k $[-]$, $k \in \{0, 1, \cdots, q-1\}$, which describes the dynamics in mesoscopic sense. The variable q denotes the number of directions, in which the information is propagating within one time step. In

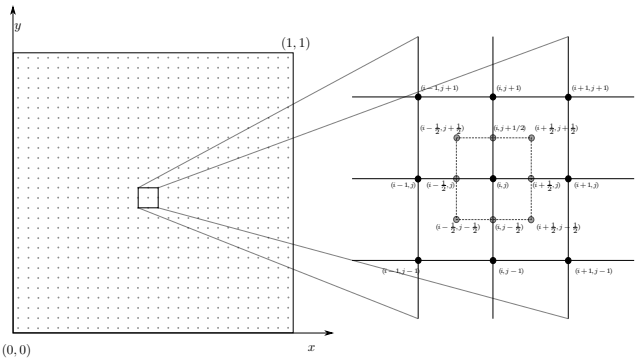


Fig. 3.1: Discretization of the computational domain Ω. Light gray circles correspond to the points $\vec{x}_{i\,j}$.

this work, we use $q = 9$. Next, it is common to work with non-dimensional units in the LBM community. To transform parameters in physical units to non-dimensional units, Δ_ℓ and Δ_t are used as the scaling factors.

The evolution of discrete probability functions is determined $\forall \vec{x}_{i\,j} \in \hat{\Omega}$, $\forall t_n \in \{1, 2, \cdots, N_t\}$ by the equation

$$f_k(\vec{x}_{i\,j} + \delta_t\,\vec{\xi}_k, t_n + \delta_t) - f_k(\vec{x}_{i\,j}, t_n) = C_k(\vec{x}_{i\,j}, t_n), \tag{3.2}$$

where δ_t is the non-dimensional time step, $\vec{\xi}_k$ $[-]$ is the non-dimensional discrete mesoscopic velocity,

$$\vec{\xi}_k = \begin{cases} (0,0)^T & \text{for } k = 0, \\ (1,0)^T, (0,1)^T, (-1,0)^T, (0,-1)^T & \text{for } k = 1,2,3,4, \\ (1,1)^T, (-1,1)^T, (-1,-1)^T, (1,-1)^T & \text{for } k = 5,6,7,8. \end{cases} \tag{3.3}$$

C_k $[-]$ is the discrete collision operator. In this work we use the single relaxation time collision operator in the form

$$C_k = -\frac{\delta_t}{\tau}\left(f_k - f_k^{(eq)}\right), \tag{3.4}$$

where τ $[-]$ is the relaxation parameter. This parameter is related to the mobility as $M = c_s^2\left(\tau - \frac{\delta_t}{2}\right)$. $f_k^{(eq)}$ $[-]$ is the discrete equilibrium density function. To simulate Eq. (2.1), the equilibrium density function is defined by [22]

$$f_k^{(eq)} = \phi w_k \left[1 + \frac{\vec{\xi}_k \cdot \vec{u}}{c_s^2} + \frac{\left(\vec{\xi}_k \cdot \vec{u}\right)^2}{2c_s^4} - \frac{\vec{u} \cdot \vec{u}}{2c_s^2}\right] + \frac{M}{c_s^2}\frac{(1 - 4\phi^2)}{W}(\vec{\xi}_k \cdot \vec{n}), \tag{3.5}$$

where w_k $[-]$ are weights satisfying

$$w_k = \begin{cases} \frac{4}{9} & \text{for } k = 0, \\ \frac{1}{9} & \text{for } k = 1,2,3,4, \\ \frac{1}{36} & \text{for } k = 5,6,7,8. \end{cases} \tag{3.6}$$

c_s [−] is the non-dimensional speed of sound and $\vec{n}$ [−] is the normal vector to the interface Γ. The normal vector is defined as $\vec{n} = {}^{\nabla\phi}/_{\|\nabla\phi\|}$, it can be approximated using finite difference method as in the Section 3.2 or using central moments of discrete density functions [22]. The second way is local and thus it is more suitable for parallel implementation, nevertheless, it is not straightforward to express the numerical error of this approach. Thus the norm $\|\nabla\phi\|$ is calculated using the finite difference method as shown later.

The unknown phase parameter ϕ is recovered from discrete density functions as

$$\phi = \sum_{k=0}^{8} f_k. \tag{3.7}$$

LBM is implemented using our in-house code based on C++ and CUDA framework enabling computation on NVidia GPU cards.

3.2. Finite difference method. For an arbitrary function $g = g(\vec{x}, t)$, the notation $g^n_{i\,j} = g(\vec{x}_{i\,j}, t_n)$ is introduced.

The advection part is discretized as follows

$$\nabla \cdot (\phi\vec{u})^n_{i\,j} \approx \frac{(\phi\vec{u})^n_{i+1\,j} - (\phi\vec{u})^n_{i-1\,j}}{2\Delta_\ell} + \frac{(\phi\vec{u})^n_{i\,j+1} - (\phi\vec{u})^n_{i\,j-1}}{2\Delta_\ell}. \tag{3.8}$$

Next, if we label

$$\vec{A}^n_{i\,j} = \left(\frac{A}{B}\right)^n_{i\,j} = M\left(\nabla\phi - \frac{\nabla\phi}{\|\nabla\phi\|}\frac{1-4\phi^2}{W}\right)^n_{i\,j} \tag{3.9}$$

then the diffusion part is approximated as

$$(\nabla \cdot \vec{A})^n_{i\,j} \approx \frac{A^n_{i+1/2\,j} - A^n_{i-1/2\,j}}{\Delta_\ell} + \frac{B^n_{i\,j+1/2} - B^n_{i\,j-1/2}}{\Delta_\ell}, \tag{3.10}$$

where

$$A^n_{i+1/2\,j} \approx M\left(\frac{\phi^n_{i+1\,j} - \phi^n_{i\,j}}{\Delta_\ell} - \frac{\frac{\phi^n_{i+1\,j} - \phi^n_{i\,j}}{\Delta_\ell}}{\|\nabla\phi\|^n_{i+1/2\,j}}\frac{1-4(\phi^2)^n_{i+1/2\,j}}{W}\right), \tag{3.11a}$$

$$B^n_{i\,j+1/2} \approx M\left(\frac{\phi^n_{i\,j+1} - \phi^n_{i\,j}}{\Delta_\ell} - \frac{\frac{\phi^n_{i\,j+1} - \phi^n_{i\,j}}{\Delta_\ell}}{\|\nabla\phi\|^n_{i\,j+1/2}}\frac{1-4(\phi^2)^n_{i\,j+1/2}}{W}\right), \tag{3.11b}$$

$$\left(\|\nabla\phi\|^2\right)^n_{i+1/2\,j} \approx \left(\frac{\phi^n_{i+1\,j} - \phi^n_{i\,j}}{\Delta_\ell}\right)^2 + \left(\frac{\phi^n_{i+1/2\,j+1/2} - \phi^n_{i+1/2\,j-1/2}}{\Delta_\ell}\right)^2 + \alpha^2, \tag{3.11c}$$

$$\phi^n_{i+1/2\,j} \approx \frac{1}{2}\left(\phi^n_{i+1\,j} + \phi^n_{i\,j}\right), \tag{3.11d}$$

$$\phi^n_{i+1/2\,j+1/2} \approx \frac{1}{4}\left(\phi^n_{i+1\,j+1} + \phi^n_{i+1\,j} + \phi^n_{i\,j+1} + \phi^n_{i\,j}\right). \tag{3.11e}$$

The remaining terms can be obtained by permuting the indices in Eqs. (3.11). Next α ensures not division by zero. Because the value of α does not have significant impact on the finite difference solution, $\alpha = 10^{-5}$ is used for all simulations using FDM.

The time derivative

$$\frac{\partial \phi_{ij}^n}{\partial t} = -\nabla \cdot (\phi \vec{u})_{ij}^n + (\nabla \cdot \vec{A})_{ij}^n \tag{3.12}$$

is solved using the fourth order Runge-Kutta method [24] to have sufficient order of discretization in time.

FDM is implemented both in series and in parallel using C++. The parallel version is implemented both on CPU and on GPU using Template Numerical Library (TNL) [25]. TNL is a numerical library based on template meta-programming and offers a flexible code design.

4. Results & Discussion. First, one benchmark problem is selected to investigate the efficiency of both LBM and FDM implementation. Then, the results of LBM and FDM are compared and the experimental order of convergence is determined.

The benchmark problem is the diagonal displacement of a circular disk with diameter R and center S. The parameters for the computation are $M_p = 0.05\ \mathrm{m^2\,s^{-1}}$, $M_0 = 0.001$, $W_p = 0.03$ m, $\vec{u}_p = (100, 100)$ m, $T_{fin} = 0.1$ s, $R_p = 0.5$ m, and $S_p = [0.5, 0.5]$ m, where the subscript p denotes parameters in physical units and 0 in non-dimensional units.

4.1. Implementation efficiency. To investigate the efficiency of our implementation, 5 different meshes are used with $N_x = N_y = 32, 64, 128, 256, 512$. In the case of FDM, serial implementation on CPU, parallel implementation on CPU using TNL and parallel implementation on GPU using TNL are used. In the case of LBM, only parallel implementation using C++ and CUDA is used.

The computational times are given in Tab. 4.1a. Next, the speedup is determined in Tab. 4.1b. From Tab. 4.1a, the implementation of FDM using TNL on GPU provides the results in the shortest time. The second fastest method is the GPU implementation of LBM. LBM is slower because it uses more unknowns and, thus, it involves more accesses to the global memory.

We can conclude from the results in Tab. 4.1b that TNL is an efficient numerical library which allows to speedup numerical solutions.

4.2. Numerical stability of LBM. In this subsection, the numerical stability of LBM is discussed. It was observed in our simulation that the method can be unstable due to the numerical computations of normal vectors. Thus we start with the discussion on the computation of normal vectors in LBM. Two methods for the stabilization are used. The first method is the same as in FDM, see Eq. (3.11c). The value of the parameter α has impact on the numerical solution. If α is huge ($\alpha = 10^{-2}$), the solution contains oscillations in values of ϕ. If α is small ($\alpha = 10^{-9}$) the solution does not contain the oscillations.

Based on the results for different values of the parameter α, similar criterion was suggested. As the magnitude of the phase parameter $|\phi|$ goes to 0.5, the value of $\frac{M}{c_s^2}\frac{(1-4\phi^2)}{W}(\vec{\xi}_k \cdot \vec{n})$ is small because of the expression $(1 - 4\phi^2)$. Next, if $|\phi|$ is close to the value 0.5, the norm $\|\nabla\phi\|$ is small and can cause numerical instabilities. Thus, empirical criterion was introduced. If $|\phi| > H$, the term $\frac{M}{c_s^2}\frac{(1-4\phi^2)}{W}(\vec{\xi}_k \cdot \vec{n})$ is neglected. Nevertheless, the choice of $H = 0.5 - \epsilon$ can have significant influence on the numerical solution ϕ. It can be illustrated in Figs. 4.2, where the contour of ϕ are drawn.

If $\epsilon = 10^{-2}$, then the method is unstable for values of ϕ close to -0.5 and 0.5. Next, on the finer mesh, the high value of H causes higher diffusion. It is illustrated

$N_x \times N_y$	FDM_serial	FDM_CPU_TNL	FDM_GPU_TNL	LBM
32×32	1.30	1.11	0.53	0.32
64×64	20.65	16.32	2.04	2.18
128×128	331.74	266.46	8.34	23.76
256×256	5307.35	4265.93	45.56	251.85
512×521	85013.32	69533.82	451.41	2054.87

(a) Computation times in seconds.

$N_x \times N_y$	FDM_CPU_TNL	FDM_GPU_TNL	LBM
32×32	1.17	2.45	4.06
64×64	1.26	10.12	9.47
128×128	1.24	39.78	13.96
256×256	1.24	116.49	21.07
512×512	1.22	188.32	41.37

(b) Speedup based on times in Tab. 4.1a.

Table 4.1: Comparison of three different parallel implementations of FDM and one parallel implementation of LBM. The abbreviations FDM_serial, FDM_CPU_TNL, and FDM_GPU_TNL represent serial implementation on CPU, parallel implementation on CPU using TNL, and parallel implementation on GPU using TNL, respectively.

in Figs. 4.2a and 4.2c. If we set $\epsilon = 10^{-9}$, then the results are much better compared to the initial condition, see Figs. 4.2b and 4.2d.

In Fig. 4.1, numerical L_1 and L_2 errors are drawn for different values of α and ϵ. The error for mesh with $N \times N$ points is computed according to formulae

$$L_a(N) = \|\phi_N - \phi_N^0\|_{L_a},$$

where $a \in \{1, 2\}$. It illustrates that if $\alpha = \epsilon = 10^{-9}$, similar results are obtained.

From the results in Figs. 4.2 and 4.1, we can conclude that our suggested criterion can stabilize LBM but the choice of H can significantly influence the results of ϕ. However, in the multiphase simulation, the most significant is the set $\Gamma(t) = \{\vec{x} \in \Omega | \phi(\vec{x}, t) = 0\}$ which was determined for both values of H sufficiently, see Figs. 4.2.

4.3. Comparison of LBM and FDM. In this subsection, results of LBM and FDM are compared. In Tabs. 4.2, 4.3, L_1, L_2 errors and corresponding experimental order of convergence are given. The experimental order of convergence is defined as

$$EOC_a = \frac{\ln(L_a(N)) - \ln(L_a(2N))}{\ln(2)}, \tag{4.1}$$

where $a \in \{1, 2\}$.

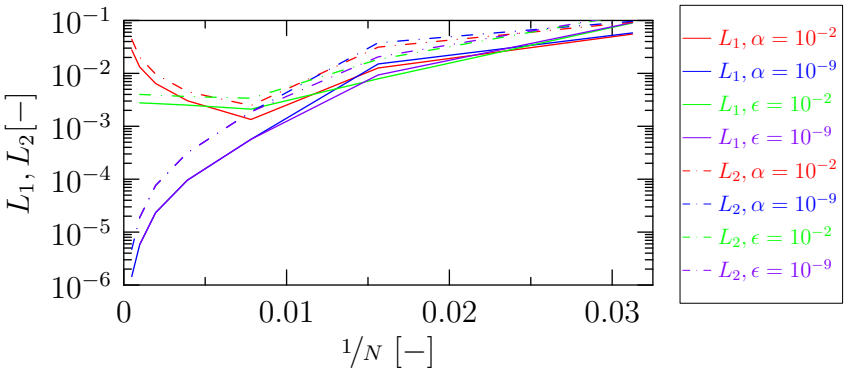


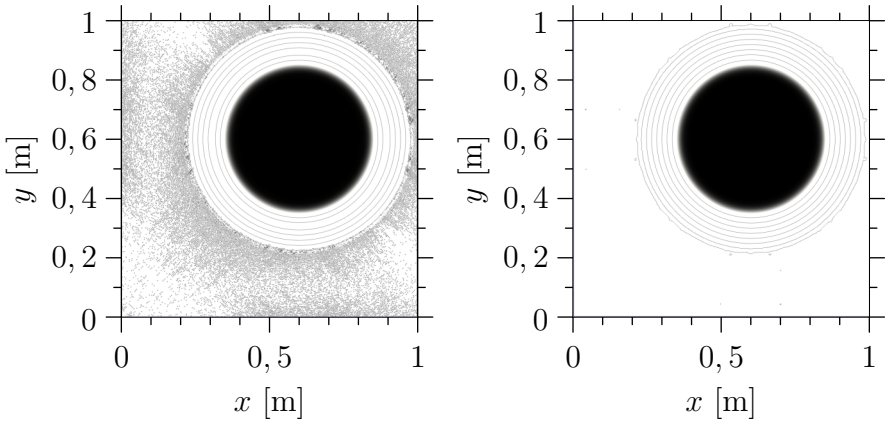
Fig. 4.1: Comparison of L_1 and L_2 numerical errors for LBM with different values of parameters α and ϵ.

$N_x \times N_y$	L_1	EOC_1	L_2	EOC_2
32×32	$3.26 \cdot 10^{-1}$		$4.72 \cdot 10^{-1}$	
		0.68		0.44
64×64	$2.04 \cdot 10^{-1}$		$3.47 \cdot 10^{-1}$	
		0.95		0.63
128×128	$1.05 \cdot 10^{-1}$		$2.24 \cdot 10^{-1}$	
		2.93		2.36
256×256	$1.37 \cdot 10^{-2}$		$4.35 \cdot 10^{-2}$	
		3.89		3.70
515×512	$9.23 \cdot 10^{-4}$		$3.33 \cdot 10^{-3}$	
		2.00		2.00
1024×1024	$2.30 \cdot 10^{-4}$		$8.32 \cdot 10^{-4}$	
		2.00		2.00
2048×2048	$5.75 \cdot 10^{-5}$		$2.08 \cdot 10^{-4}$	

Table 4.2: L_1 and L_2 errors and experimental order of convergence EOC for FDM.

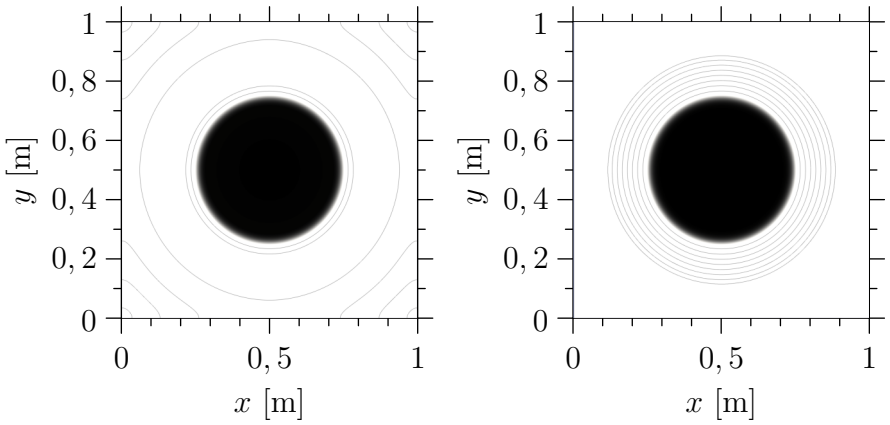
$N_x \times N_y$	L_1	EOC_1	L_2	EOC_2
32×32	$7.55 \cdot 10^{-2}$		$1.23 \cdot 10^{-1}$	
		3.37		2.81
64×64	$7.29 \cdot 10^{-3}$		$1.76 \cdot 10^{-2}$	
		3.52		2.97
128×128	$6.36 \cdot 10^{-4}$		$2.23 \cdot 10^{-3}$	
		2.11		2.13
256×256	$1.47 \cdot 10^{-4}$		$5.10 \cdot 10^{-4}$	
		1.97		1.98
515×512	$3.75 \cdot 10^{-5}$		$1.29 \cdot 10^{-4}$	
		1.99		1.99
1024×1024	$9.42 \cdot 10^{-6}$		$3.23 \cdot 10^{-5}$	
		1.99		1.99
2048×2048	$2.36 \cdot 10^{-6}$		$8.09 \cdot 10^{-6}$	

Table 4.3: L_1 and L_2 errors and experimental order of convergence EOC for LBM.



(a) Mesh 512×512, $\epsilon = 10^{-2}$.

(b) Mesh 512×512, $\epsilon = 10^{-9}$.



(c) Mesh 2048×2048, $\epsilon = 10^{-2}$.

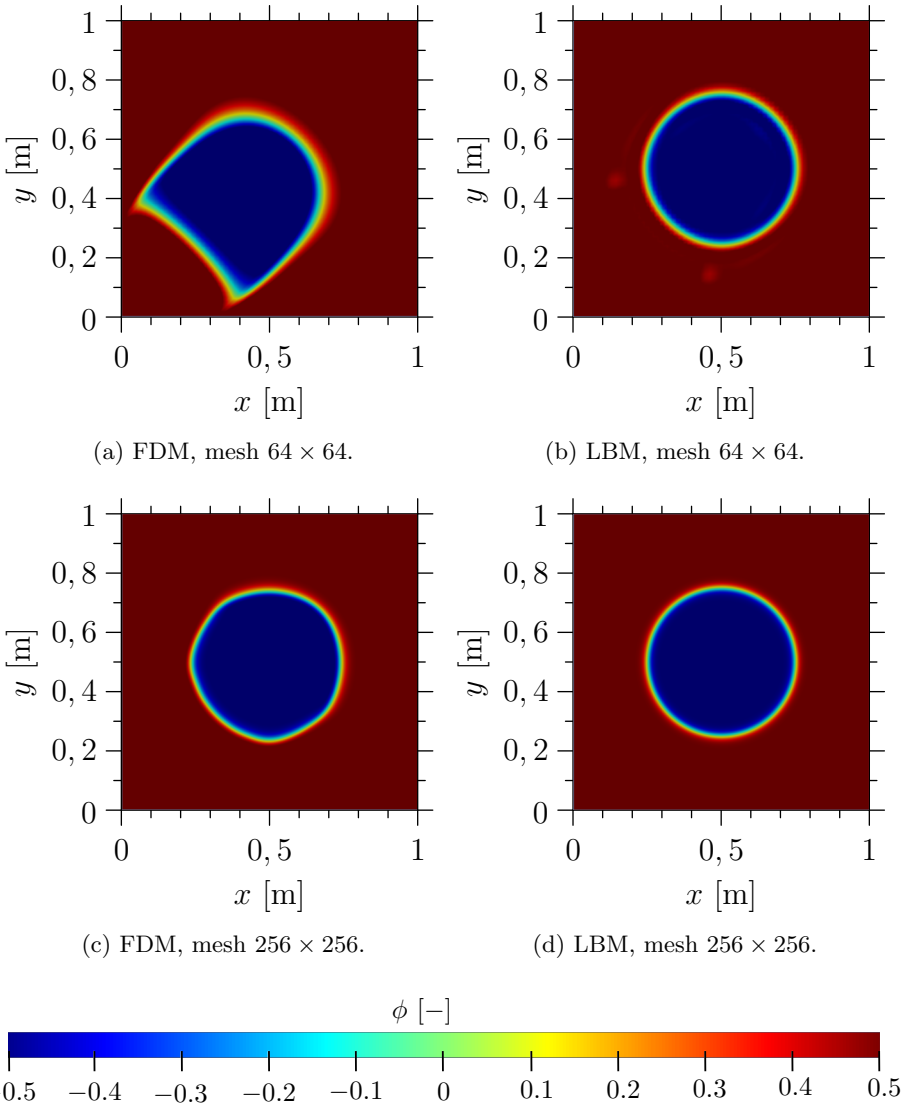
(d) Mesh 2048×2048, $\epsilon = 10^{-9}$.

Fig. 4.2: Contour lines of ϕ for different values of the parameter ϵ and different mesh sizes. The black circle represents the area where $\phi < 0$. The results are computed using LBM and are given at the time $t = \Delta_t$.

It can be seen in Tabs. 4.2, 4.3 that for coarse computational meshes, the experimental order of convergence is lower than for finer meshes. The reason for this observation is that the initial profile of the hyperbolic tangent could not be approximated sufficiently and thus the results are impaired, see Figs. 4.3a, 4.3b. Both methods have experimental order of convergence near 2, nevertheless, LBM has lower L_1 and L_2 error for all meshes. Next, Figs. 4.3c, 4.3d illustrate that LBM preserve the initial shape of the boundary Γ much better on coarser computational meshes than FDM.

5. Conclusion. The first computational study was to measure the time efficiency of our numerical solvers. It was observed that the implementation of the finite difference method on GPU using Template Numerical Library is very efficient compared to the other implementation discussed in this article.

Next, the finite difference method and the lattice Boltzmann method were com-



(a) FDM, mesh 64 × 64.

(b) LBM, mesh 64 × 64.

(c) FDM, mesh 256 × 256.

(d) LBM, mesh 256 × 256.

Fig. 4.3: Comparison of the FDM and LBM on two different computational meshes. The results are given at the final time $t = T_{fin}$.

pared to the problem of the diagonal motion of a circle. We have observed the experimental order of convergence was near 2 for both methods. However, LBM has lower errors and produce better results on courser meshes.

These observations demonstrate that in the multiphase flow problems, the solution of the phase-field equation using the lattice Boltzmann method is more suitable than the finite difference method, which is commonly used in literature.

Acknowledgments. The work was supported by the Czech Science Foundation project No. 18-09539S , by the project No. NV19-08-00071 of the Ministry of Health of the Czech Republic,by the Ministry of Education, Youth and Sports of the Czech Republic under the OP RDE grant number CZ.02.1.01/0.0 /0.0/16 019/0000753 “Re-

search centre for low-carbon energy technologies", and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS20/184/OHK4/3T/14.

REFERENCES

[1] H. Huang, M. Sukop, and X. Lu, *Multiphase lattice Boltzmann methods: Theory and application,* John Wiley & Sons, 2015.
[2] Y. Sun and C. Beckermann, *Sharp interface tracking using the phase-field equation,* Journal of Computational Physics, 2007, 220.2: 626-653.
[3] J. W, Boettinger, J. A. Beckermann, and A. Karma, *Phase-field simulation of solidification,* Annual review of materials research, 2002, 32.1: 163-194.
[4] D. Jacqmin, *Calculation of two-phase Navier–Stokes flows using phase-field modeling,* Journal of Computational Physics, 1999, 155.1: 96-127.
[5] S. Luckhaus, *Solutions for the two-phase stefan problem with the Gibbs—Thomson law for the melting temperature. In: Fundamental contributions to the continuum theory of evolving phase interfaces in solids,* Springer, Berlin, Heidelberg, 1999. p. 317-327.
[6] H. Greberg, G. V. Paolini, J. Satherley, J. Penfold, and S. Nordholm, *Generalized van der Waals theory of interfaces in simple fluid mixtures,* Journal of colloid and interface science, 2001, 235.2: 334-343.
[7] M. E. Gurtin, *On the Two-Phase Stefan Problem with Interfacial Energy and Entropy,* Technical Summary Report, 1985.
[8] J. A. Sethian, *Level set methods: Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science,* Cambridge: Cambridge University Press, 1996.
[9] P.-H. Chiu and Y.-T. Lin, *A conservative phase field method for solving incompressible two-phase flows,* Journal of Computational Physics, 2011, 230.1: 185-204.
[10] A. K. Gunstensen, D. H. Rothman, S. Zaleski and G. Zanetti, *Lattice Boltzmann model of immis- cible fluids,* Physical Review A, 1991, 43.8: 4320.
[11] X. W. Shan and H. D. Chen, *Lattice Boltzmann model for simulating flows with multiple phases and components,* Physical review E, 1993, 47.3: 1815.
[12] M. R. Swift, E. Orlandi, W. R. Osborn, and J. M. Yeomans, *Lattice Boltzmann simulations of liquid-gas and binary fluid systems,* Physical Review E, 1996, 54.5: 5041.
[13] M. R. Swift, W. R. Osborn, and J. M. Yeomans, *Lattice Boltzmann simulation of nonideal fluids,* Physical review letters, 1995, 75.5: 830.
[14] M. S. Allen and J. W. Cahn, *Mechanisms of phase transformations within the miscibility gap of Fe-rich Fe-Al alloys,* Acta Metallurgica, 1976, 24.5: 425-437.
[15] Z. Guo and C. Shu, *Lattice Boltzmann method and its applications in engineering,* World Scientific, 2013.
[16] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Orest and E. Viggen, *The lattice Boltzmann method,* Springer International Publishing, 2017, 10: 978-3.
[17] D. d'Humi'ere, *Generalized lattice-Boltzmann equations,* Rarefied gas dynamics, 1992.
[18] M. Geier, A. Greiner and J. G. Korvink, *Cascaded digital lattice Boltzmann automata for high Reynolds number flow,* Physical Review E, 2006, 73.6: 066705.
[19] M. Geier, M. Schönherr, A. Pasquili and M. Krafczyk, *The cumulant lattice Boltzmann equation in three dimensions: Theory and validation,* Computers & Mathematics with Applications, 2015, 70.4: 507-547.
[20] S. Chikatamarla, S. Ansumali and I. Karlin, *Entropic lattice Boltzmann models for hydrodynamics in three dimensions,* Physical review letters, 2006, 97.1: 010201.
[21] J. G. Zhou, *Lattice Boltzmann methods for shallow water flows,*Berlin: Springer, 2004.
[22] M. Geier, A. Fakhari, and T. Lee, *Conservative phase-field lattice Boltzmann model for interface tracking equation,* Physical Review E, 2015, 91.6: 063309.
[23] A. Fakhari, M. Geier, and D. Bolster, *A simple phase-field model for interface tracking in three dimensions,* Computers & Mathematics with Applications, 2019, 78.4: 1154-1165.
[24] J. C. Butcher, *Numerical methods for ordinary differential equations,* John Wiley & Sons, 2016.
[25] *Template Numerical Library,* `https://tnl-project.org/`.