Optimization and Stabilization of LBM Free Surface Flow Simulations using Adaptive Parameterization

Nils Thürey [a] Thomas Pohl [a] Ulrich Rüde [a] Markus Öchsner [b] Carolin Körner [b]

[a] *System Simulation Group, Cauerstr. 6, 91058 Erlangen, Germany*

[b] *Material Science WTM, Martensstr. 5, 91058 Erlangen, Germany*

Corresponding author: nils.thuerey@cs.fau.de

Abstract

We present a method to speed up and stabilize free surface simulations with the lattice Boltzmann method (LBM). This is done by adaptively changing the parameterization of the simulation in a way that corresponds to a different size of the simulation time step. This means that the Mach number changes as well, and requires a rescaling of all distribution functions. Hence we only perform the rescaling when the velocities in the simulation become too large or small. We will demonstrate the effect of this procedure for two and three dimensional test cases. In addition to a reduction of the necessary LBM steps, this method can also be used to stabilize gravity driven simulations, where the maximum velocities are not known a priori.

Key words: lattice Boltzmann method, free surfaces, adaptive time steps

1 Introduction

A variety of applications require the simulation of a liquid with a free surface. One example is the optimization of production processes like casting or the creation of metal foams involving the accurate tracking of a liquid gas interface. Also for graphical applications the animation of liquids is still a challenge [1]. In both cases discretizations of the *Navier-Stokes equations* (NSE) are commonly used to simulate fluids. For the implementation described here we have chosen the *lattice Boltzmann method* (LBM), which was derived from the lattice gas methods and can be regarded as a first order explicit discretization of the Boltzmann equation discretized in phase space. It is able to efficiently

handle complex, dynamic geometries and topologies. However, especially in the field of computer graphics LBM is rarely used to simulate free surfaces. The boundary conditions for the free surface are relatively simple compared to other approaches [4], and thus allow an efficient implementation. The method we will present in the following can be used to speed up and stabilize flows with highly varying velocities, such as gravity driven free surface flows. An overview of the free surface LBM is given in Section 2, while in Section 3 the implementation of adaptive time stepping is explained. Section 4 will demonstrate the effect of the method for various test cases and problem sizes.

2 The Lattice Boltzmann Method

The LBM works on a (in our case equidistant) grid of cells, each of which contains information about a fixed number of discrete velocities. For two-dimensional simulations nine velocity directions are commonly used (D2Q9 model [9]), while for three dimensions, the D3Q19 model with nineteen velocities is the most common, and will also be used for the following examples. The directions of the velocity vectors are shown in Figure 1. For a cell at position $\mathbf{x}$ a *particle distribution function* (DF) $f_i(\mathbf{x}, t)$ has to be stored as a floating-point value for each velocity direction ($i = 0..18$). The direction vectors $\mathbf{e}_i$ are defined as $\mathbf{e}_0 = (0, 0, 0)$, $\mathbf{e}_{1,2} = (\pm 1, 0, 0)$, $\mathbf{e}_{3,4} = (0, \pm 1, 0)$, $\mathbf{e}_{5,6} = (0, 0, \pm 1)$, $\mathbf{e}_{7..10} = (\pm 1, \pm 1, 0)$, $\mathbf{e}_{11..14} = (0, \pm 1, \pm 1)$ and $\mathbf{e}_{15..18} = (\pm 1, 0, \pm 1)$. The conserved moments density ρ and mass flux $\mathbf{j}$ can be computed as

$$\rho = \sum_{i=0}^{18} f_i \quad , \qquad \mathbf{j} = \rho \mathbf{u} = \sum_{i=0}^{18} f_i \mathbf{e}_i \ , \tag{1}$$

where $\mathbf{u}$ is the fluid velocity. The basic algorithm proceeds in two steps – the *stream* step and the *collide* step. As the size of a time step is usually normalized to 1, the streaming step results in a movement of each DF to the adjacent cell along its velocity vector, as shown in Figure 1. The collide step then accounts for the collisions between the particles that occur in a real fluid during their movement, by relaxing the DFs towards an equilibrium state. For the BGK model [9] , the equilibrium DFs f_i^{eq} are defined as

$$f_i^{eq}(\rho, \mathbf{u}) = w_i \rho \left[1 - \frac{3}{2} \mathbf{u} \cdot \mathbf{u} + 3 \mathbf{e}_i \cdot \mathbf{u} + \frac{9}{2} (\mathbf{e}_i \cdot \mathbf{u})^2 \right] \ , \tag{2}$$

where $\mathbf{u}$ and ρ are the macroscopic velocity and density, respectively, and w_i are weights that are given by the length of the velocity vector: $w_0 = 1/3$, $w_{1..6} = 1/18$ and $w_{7..18} = 1/36$. The value of the distribution function for the next time step $f_i(\mathbf{x}, t + \Delta t)$ is calculated by:

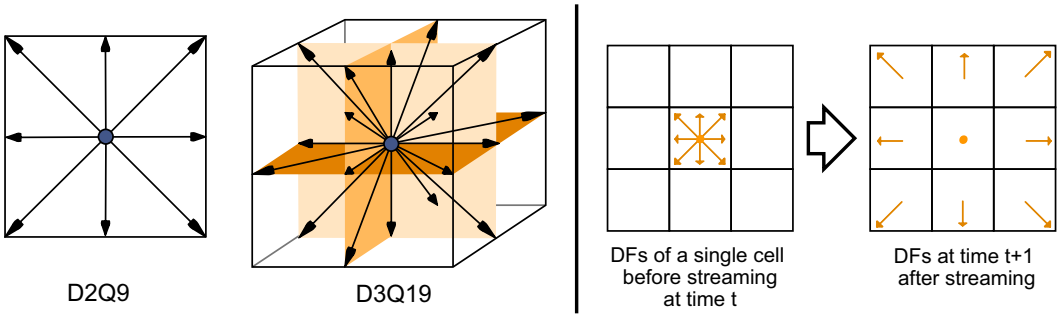

Fig. 1. To the left the velocities of the D2Q9 and D3Q19 LBM models are shown, on the right side the DFs of a LBM cell before and after the stream step can be seen.

$$f_i(\mathbf{x}, t + \Delta t) = (1 - \omega) f_i(\mathbf{x}, t) + \omega f_i^{eq}\Big(\rho(\mathbf{x}, t), \mathbf{u}(\mathbf{x}, t)\Big) \ , \tag{3}$$

with the relaxation factor ω, that is set according to the viscosity of the fluid:

$$\omega = 2/(6\nu + 1) = 1/\tau \ . \tag{4}$$

Here ν is the kinematic viscosity of the fluid in lattice units. Hence, the equation for the time evolution is:

$$f_i(\mathbf{x}+\mathbf{e}_i\Delta t, t+\Delta t) - f_i(\mathbf{x}, t) = -\Delta t \ \omega \Big[f_i(\mathbf{x}, t) - f_i^{eq}\Big(\rho(\mathbf{x}, t), \mathbf{u}(\mathbf{x}, t)\Big)\Big]. \tag{5}$$

This is known as the BGK lattice Boltzmann equation due to its approximation of the particle collisions with a single relaxation parameter. To enhance the stability of the algorithm for higher Reynolds numbers other methods, e.g. multiple relaxation time models [2, 5] could be used. External forces like gravity can be applied by accelerating the fluid in each cell before the calculation of the equilibrium distribution functions.

Thus for simulations with an acceleration force $\mathbf{g}$ Eq. 3 becomes

$$f_i(\mathbf{x}, t + \Delta t) = (1 - \omega) f_i(\mathbf{x}, t) + \omega f_i^{eq}\Big(\rho(\mathbf{x}, t), \mathbf{u}(\mathbf{x}, t) + \mathbf{g}/\omega\Big) \ . \tag{6}$$

We use the bounce-back boundary conditions for obstacles and at the domain boundaries (here, this results in a zero normal and tangential velocity). Hence, if the adjacent cell is an obstacle cell, the DFs are reflected at the cell boundary during streaming.

For the simulation of a free surface, we use the boundary conditions described in [7]. Figure 2 shows an example simulation of breaking dam test case simulated with the algorithm described below. The gas and fluid phase of a free surface simulation require a distinction between cells that do not contain any

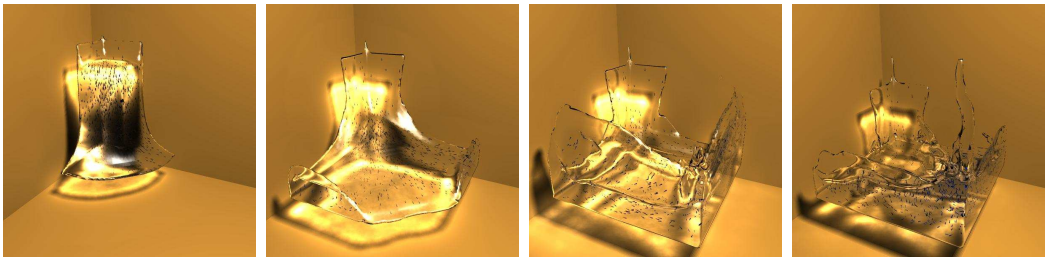

Fig. 2. Several frames of a breaking dam simulation performed with the aglorithm described in Section 2.

fluid (only gas), interface cells which are partially filled, and cells completely filled with fluid. Fluid cells can be treated as described in Section 2, and no computations are necessary for empty cells. However, all DFs which should be streamed from empty cells need to be reconstructed for the interface cells with

$$f_I(\mathbf{x}, t+1) = (f_i^{eq}(\mathbf{u}, \rho_g) + f_I^{eq}(\mathbf{u}, \rho_g)) - f_i(\mathbf{x}, t) \ . \tag{7}$$

Here f_I is related to the anti-parallel lattice vector $\mathbf{e}_I$ with $\mathbf{e}_I = -\mathbf{e}_i$. Note that in this form the boundary conditions do not include any surface tension. The gas density ρ_g prescribes the interface conditions for pressure at leading order, where the atmospheric density is simply the reference density of the LBM simulation. The velocity $\mathbf{u}$ required for the equilibrium DFs in Eq. 7 is that of the fluid at time t, as the gas is assumed to have the same velocity as the fluid close to the interface. Bubbles are treated by calculating their volume in each time step, which is used together with the initial bubble mass to calculate the gas density. A high gas density ($\rho_g > 1$) results in a pressure force at the fluid interface pushing the fluid away, as both equilibrium DFs in Eq. 7 are influenced by ρ_g. A lower pressure ($\rho_g < 1$), on the other hand, applies a force towards the gas phase. To track the surface movement the mass exchange with all neighboring fluid and interface cells needs to be computed by subtracting the outgoing DFs from the incoming ones during streaming

$$m(\mathbf{x}, t+\Delta t) = m(\mathbf{x}, t) + \sum_{l=0}^{18} \Upsilon(\mathbf{x}+\mathbf{e}_i, \mathbf{x}) \ (f_I(\mathbf{x}+\mathbf{e}_i, t) - f_i(\mathbf{x}, t)) \quad , \tag{8}$$

where $m(\mathbf{x}, t)$ denotes the mass of the cell at position $\mathbf{x}$ and time t, and $\epsilon(\mathbf{x}, t)$ is the fraction of the cell that is filled with fluid. The factor $\Upsilon(\mathbf{x}+\mathbf{e}_i, \mathbf{x})$ accounts for the area of fluid between two interface cells. It is calculated as

$$\Upsilon(\mathbf{x}+\mathbf{e}_i, \mathbf{x}) = \frac{\epsilon(\mathbf{x}+\mathbf{e}_i, t) + \epsilon(\mathbf{x}, t)}{2}, \tag{9}$$

if both cells at $\mathbf{x}+\mathbf{e}_i$ and $\mathbf{x}$ are interface cells, while it is equal to one if one of them is a fluid cell. Note that $m(\mathbf{x}, t)$ can become negative. This is

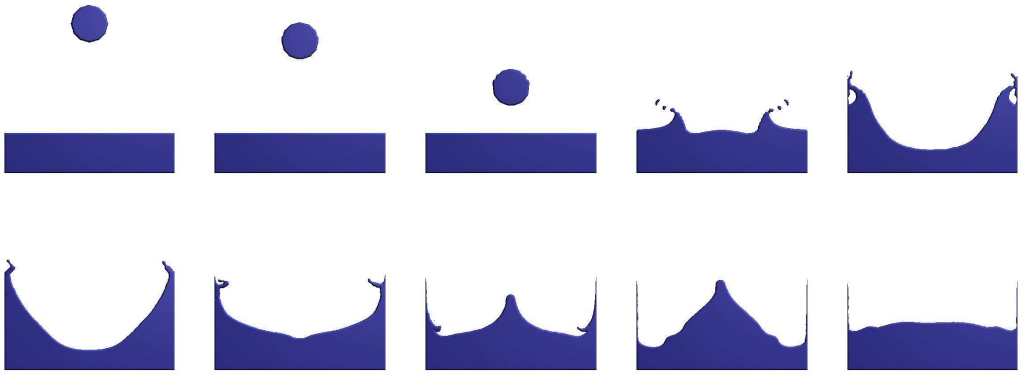

Fig. 3. Here several frames of the simulation setup for the second test case from Section 4 can be seen. A drop of fluid falls into resting fluid in a square computational domain. In this case the grid resolution is 256^2.

handled by the cell conversion and does not cause the algorithm to produce incorrect results – the algorithm still conserves mass up to machine precision. Once interface cells become completely filled or empty – indicated by a mass of zero or equal to the density, respectively – their type has to be changed accordingly. This can also cause neighboring fluid or empty cells to be changed to interface cells, since the layer of interface cells has to be closed. While the higher order analysis of these boundary conditions is a work in progress, the implementation details of the algorithm can be found in our companion paper [6] or in [10].

3 Adaptive Mach Numbers and Time Steps

Simulation configurations like the one shown on the title page exhibit a highly varying range of velocities. As the LBM intrinsically limits the maximum velocity of a cell, a setup like this usually requires a time step that is sufficiently small to assure that the largest occurring velocities remain valid during the course of the simulation. To alleviate this restriction, we have implemented a technique to adaptively modify the parameterization to change the time step size. This also results in a change of the Mach number $Ma = \mathbf{u}/c_s$, as the grid size and thus the speed of the information propagation (determined by c_s) stay fixed, but the velocities $\mathbf{u}$ are rescaled. However, it will be shown in Section 4 that this does not disturb the simulation. Given a fluid viscosity ν and an acceleration force $\mathbf{g}$ a maximum time step size $\Delta t_{\max}$ can be calculated by

$$\Delta t_{\max} \leq \sqrt{\frac{\Delta\rho \cdot \Delta x}{|\mathbf{g}|}} \quad . \tag{10}$$

Here $\Delta\rho$ is a dimensionless constant to limit the maximum density change per time step. For simulation setups similar to those used here, we have chosen $\Delta\rho = 10^{-3}$. As this value controls the compressibility of the fluid, larger fluid volumes may require smaller numbers. On the other end, the minimum size of a time step is limited by the stability of the LBM. For a BGK model we have chosen $\tau_{\min} \geq 0.51$. Given the kinematic viscosity of the fluid ν and $\tau_{\min}$, the lower limit of the time step can be computed as

$$\Delta t_{\min} = \frac{2\tau_{\min} - 1}{6\nu} \quad . \tag{11}$$

During each LBM step, the current maximum velocity $\mathbf{u}_{\max}$ is computed. As a threshold for the velocity we choose $|\mathbf{u}_{\text{thresh}}| = 1/6$ which is half of the speed of sound chosen for D2Q9 and D3Q19. The speed of sound thus represents the upper limit of the lattice velocity to simulate weakly compressible flow. If $\mathbf{u}_{\max}$ exceeds this threshold during the course of the simulation, we perform the following procedure to stabilize the simulation again. If the velocities become too large, this will result in a smaller time step size, while for small $\mathbf{u}_{\max}$ we change the parameters to have the simulation perform larger time steps. The size of the new time step can the be determined by

$$\Delta t_n = |\mathbf{u}_{\text{thresh}}|/|\mathbf{u}_{\max}| \; . \tag{12}$$

In the following, a subscript of o will denote values before the rescaling procedure, while n will denote values for the new parameterization. Hence, t_o is the size of the time step previously used for the LBM. To account for the new time step size, the hydrodynamic moments as well as the acceleration force have to be rescaled by $s = \Delta t_n / \Delta t_o$:

$$\rho_n = (\rho_o - \rho_{\text{ref}})s + \rho_{\text{ref}} \quad , \quad \mathbf{u}_n = \mathbf{u}_o s \quad , \quad \mathbf{g}_n = \mathbf{g}_o \; s^2 \tag{13}$$

Here ρ_{ref} denotes the current reference density of the simulation, which has to be calculated from the total fluid volume V and the overall mass M as $\rho_{\text{ref}} = V/M$. V is calculated by the sum of all fluid fraction values ϵ (which are equal to one for fluid cells), while M is computed as the sum of all cell masses. Thus for interface cells m and for fluid cells ρ is used. The deviation of the cell densities from the reference density needs to be rescaled, as the changed force $\mathbf{g}_n$ requires an adaption of the density gradient. This rescaling furthermore changes the values of m and ϵ for interface cells:

$$m_n = m_o(\rho_o/\rho_n) \quad , \quad \epsilon_n = m_n/\rho_n \tag{14}$$

Since the relaxation time depends on the size of the time step, the non-equilibrium parts of the distribution functions have to be rescaled, similar

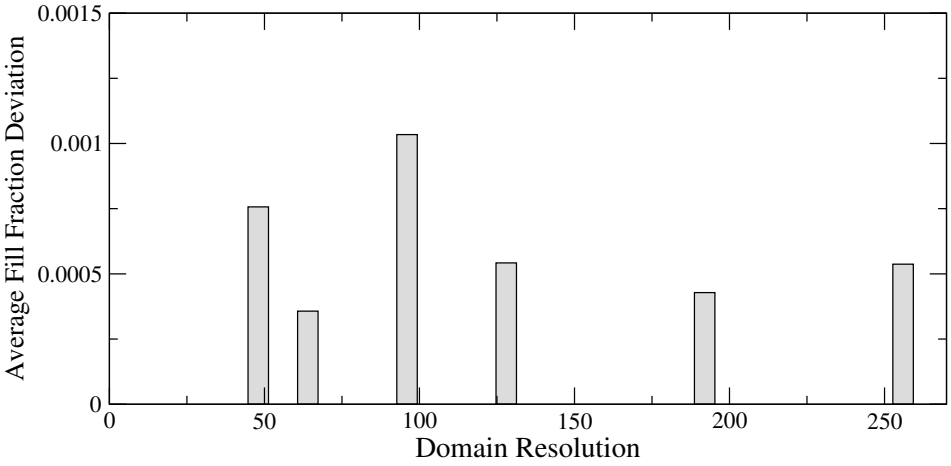

Fig. 4. Effect of the parameter change for a drop falling 2.5 times its radius, compared to a simulation without any parameter changes.

to the rescaling for simulations using differently refined grids as in [3] or [8]:

$$
\begin{aligned}
f_i' &= [f_i^{eq}(\rho_o, \mathbf{u}_o) + (f_i - f_i^{eq}(\rho_o, \mathbf{u}_o))\, s_\tau]\, s_{f_i} \quad , \text{ with} \\
s_{f_i} &= f_i^{eq}(\rho_n, \mathbf{u}_n) \;/\; f_i^{eq}(\rho_o, \mathbf{u}_o) \\
s_\tau &= \Delta t_n \tau_n \;/\; \Delta t_o \tau_o
\end{aligned}
\tag{15}
$$

Here the factor s_τ corresponds to the non-equilibrium scaling factor from [3], while the additional scaling by s_{f_i} is necessary to account for the changes of velocity and density. s_τ is thus equal for all cells and lattice directions i. The factor s_{f_i}, on the other hand, depends on i and can be different for each cell. The scaling by s_{f_i} is required to make the DFs of the cell represent the velocity $\mathbf{u}_n$ and deviation from the median density ρ_n required by the new parameterization. Thus it is assumed that the non-equilibrium part is proportional to the equilibrium part direction by direction.

Note that the steps described so far require roughly as many computations as a collision step. Thus, we only perform the rescaling when $|\mathbf{u}_{\max}|$ is larger than $|\mathbf{u}_{\text{thresh}}| \cdot \xi$ or smaller than $|\mathbf{u}_{\text{thresh}}|/\xi$, where ξ is used to control the amount of time step changes. We use a value of $\xi = 5/4$. The range of parameterization also depends on the range of valid time steps. If $\Delta t_{\max}$ already requires a τ near 0.51 there will be little room to choose smaller time step sizes. To prevent the time step from being decreased right after it was enlarged, we furthermore delay enlarging of the time step. This, in contrast to decreasing the time step size, is uncritical, as it is ensured that no high velocities currently appear in the simulation. Here a delay of four times the grid resolution in time steps yields good results. As the time step is infrequently changed in comparison with the number of LBM steps, it requires very little computational time (less than 1% for the presented simulations). The method can also be applied to flows without a free surface or without forcing. However, depending on the problem

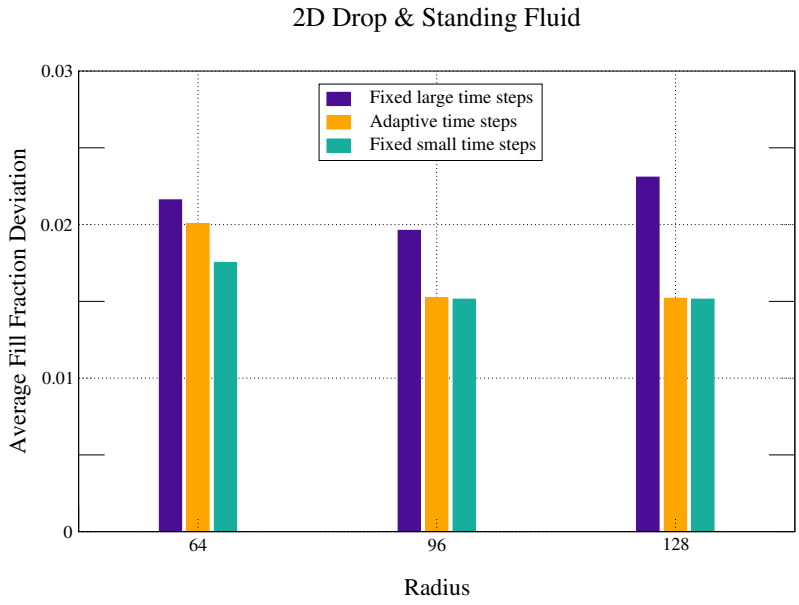

Fig. 5. Results for the test case of Figure 3 in two dimensions.

the advantages might disappear, e.g. due to constantly high velocities. Thus, the following chapter will present results for test cases where the method is applicable.

4 Results

The correctness of the following simulations will be determined by comparing the average deviation of the fluid fraction values ϵ for all cells. This effectively compares the difference of the position of the free surface for two given configurations. The values shown in Figure 4, 5 and 6 are thus computed as

$$\frac{1}{n_{\text{total}}} \sum_{\mathbf{x}\in\Omega} |\epsilon_{\text{ref}}(\mathbf{x}) - \epsilon(\mathbf{x})| \ , \tag{16}$$

where ϵ_{ref} are the fluid fraction values of a reference simulation, Ω is the size of the domain ranging from 0 to 1 in each spatial dimension, and n_{total} is the total number of chosen points where we measure the fluid fraction deviation at. For example Figure 4 was generated using points in 64 intervals along each axis, thus using $n_{\text{total}} = 4096$ points in total. If two configurations are completely different, this deviation will be close to one, while values close to zero indicate a similar shape of the fluid. We take the average value for all measured points to compare simulations of different sizes.

A first two-dimensional test to validate our rescaling procedure is a falling drop of radius 0.1 (the size of the computational domain being 1.0) falling for

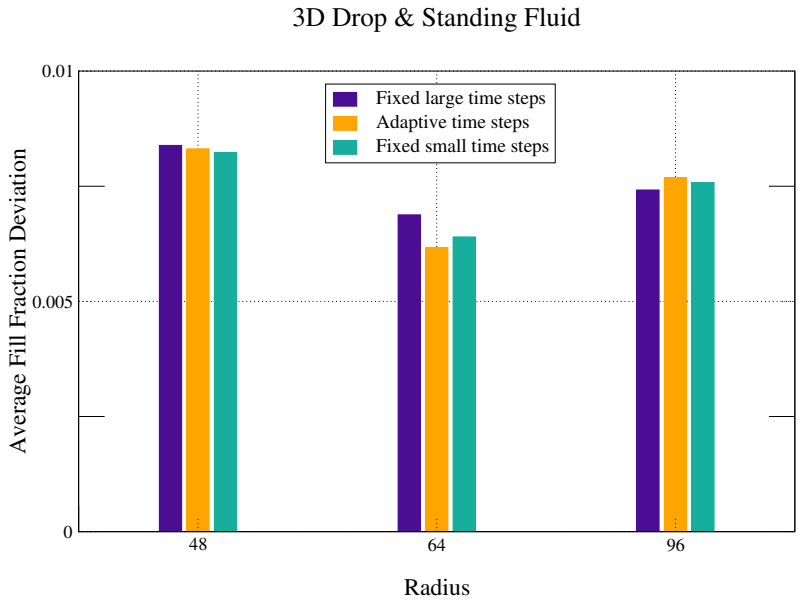

Fig. 6. Results for the test case of Figure 3 in three dimensions.

2.5 times its radius. During the course of this movement the parameterization is changed 6 times (for a resolution of 48^2) to 9 times (for a resolution of 256^2). These simulations use a $\mathbf{u}_{\text{thresh}} = 0.05$, which causes more parameterization changes than really necessary, and were compared to a simulation using the same resolution with a fixed parameterization. In Figure 4 the results of these experiments are shown. It can be seen that the results are almost independent of the resolution. Furthermore, the shape of the drop is not disturbed by the rescaling which is evident from the very small deviations, being usually less than 0.001.

A more realistic test case is shown in Figure 3. Here a drop with radius 0.1 is falling into a resting fluid of height 0.25. The results for experiments with various grid sizes in 2D can be seen on the left side of Figure 5. Here three simulations with the grid size shown in the graph were compared to a simulation with twice this resolution parameterized to run with a small time step. One of the three simulations used for comparison is run with a large time step (left bars), the next with adaptive time stepping (middle bars), while the last simulation (right bars) uses the smallest time step used in the corresponding simulation with adaptive time steps. Thus the left- and rightmost bars for each test case use parameterizations according to the largest and smallest time step sizes used during the course of the simulation for the middle bars. The simulations with lattice resolution 64^2 were initially parameterized with $\omega = 1.87$, $\mathbf{g} = (0\ ,\ -5.34 \cdot 10^{-4})$ running for approximately 1000 steps. For the larger grid resolutions the parameters were scaled to keep the Reynolds number constant (hence for the test cases with a 128^2 resolution $\omega = 1.678$). As the graph shows, the parameterization changes for time step and Mach number adaption yield the expected results lying close to the both extremes

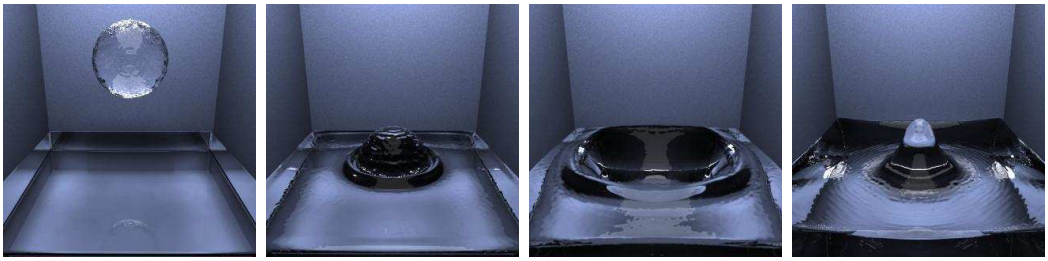

Fig. 7. A simulation of a falling drop with a grid resolution of 240^3 requiring less than 6800 steps, in contrast to more than 16000 that would have been necessary to run the simulation with the smallest time step.

of parameterization.

The graph in Figure 6 shows results for running the previous test case extended to 3D. The only difference here is, that due to memory limitations the reference simulation uses 1.5 times the shown grid resolution, instead of the factor 2 for the 2D cases. Again the results with adaptive time steps lie in the expected range. Note that these test cases were chosen to ensure that even the initial large time step size remains stable. The method presented here also allows the stabilization of simulations were the occurring velocities cannot be determined from the start. In these cases the parameterization will be automatically changed in order to stabilize the simulation, if the new value for ω does not itself cause stability problems. The method can then save significant amounts of computational time by reducing the number of necessary LBM steps. As an example, the simulation shown in Figure 7 requires 6757 LBM steps with adaptive parameterization while running the simulation with the smallest time step size would have required 16200 steps. Thus, using the method presented here, this test case runs 2.4 times faster than without it. While not all simulations will benefit this much, they will nut run slower using the adaptive parameterization, as the computational cost for checking whether to perform a parameterization change or not is negligible in comparison to the cost of each LBM step.

5 Conclusions

We have presented a method to adaptively change the Mach number and parameterization of a LBM simulation to account for a change of the time step size. The maximum velocity occurring in the simulation is used to trigger a rescaling procedure similar to the rescaling of the non-equilibrium components for grid refinement, additionally scaling the cell velocities and density deviations. It was shown that this process does not disturb the free surface flow and can yield significant speedups for simulations with varying velocities, such as gravity driven free surface flows. It furthermore stabilizes and simplifies the

parameterization for flows where maximum velocities cannot be determined in advance.

6 Acknowledgements

This research is funded by the DFG Graduate College GRK-244 *3-D Image Analysis and Synthesis*. Furthermore, we want to thank Torsten Hofmann and Thomas Zeiser (Regionales RechenZentrum Erlangen).

References

[1] D. Enright, S. Marschner, and R. Fedkiw, "*Animation and Rendering of Complex Water Surfaces*", ACM Transactions on Graphics 22, pp. 736-744, 2002

[2] D. d'Humières, "*Generalized lattice-Boltzmann Equations*", AIAA Rarefied Gas Dynamics: Theory and Simulations, Progress in Astronautics and Aeronautics 59, pp. 450–548, 1992

[3] O. Filippova and D. Hänel, "*Grid Refinement for Lattice-BGK models*", J. Comp. Phys. 147, 1998

[4] I. Ginzburg and K. Steiner, "*Lattice Boltzmann model for free-surface flow and its application to filling process in casting*", Journal of Comp. Phys. 185, pp. 61-99, 2003

[5] P. Lallemand and Li-Shi Luo, "*Theory of the Lattice Boltzmann Method: Dispersion, Dissipation, Isotropy, Galilean Invariance, and Stability*", Physical Review E 61, pp. 6546-6562, 2000

[6] C. Körner et. al, "*Lattice Boltzmann Model for Free Surface Flow Including Gas Diffusion*", to appear for ICMMES 2004

[7] C. Körner and R.F. Singer, "*Numerical Simulation of Foam Formation and Evolution with Modified Cellular Automata*", Metal Foams and Porous Metal Structures, MIT Publishing, pp. 91-96, 1999.

[8] B. Crouse, E. Rank, M. Krafczyk, and J. Tölke, "*A LB-Based Approach for Adaptive Flow Simulations*", Int. Journal of Modern Physics B, Vol. 17, pp. 109-112, 2003

[9] Y. H. Qian, D. d' Humires, and P. Lallemand, "*Lattice BGK models for Navier-Stokes equation*", Europhysics Letters, Vol. 17(6), 1992

[10] Nils Thürey, "A single-phase free-surface Lattice-Boltzmann Method", *Masters thesis*, IMMD10 University of Erlangen-Nuremberg, 2003.