Large scale simulation of fluid structure interaction using Lattice Boltzmann methods and the 'physics engine'

J. Götz[1] C. Feichtinger[2] K. Iglberger[3]
S. Donath[4] U. Rüde[5]

(Received 15 August 2008; revised 10 October 2008)

Abstract

We study the methodology behind the simulation of fluid flow with up to 150,000 fully resolved rigid bodies incorporated in the flow. The simulation is performed using a 3D Lattice Boltzmann solver for the fluid flow and a so-called rigid body physics engine for the treatment of the objects. The numerical methods, the necessary extensions and the coupling between both methods are presented in detail. Furthermore, the parallelisation is discussed and performance results are given for different test cases with up to 150,000 rigid bodies on up to 1025 processor cores. The approach enables a detailed simulation of large scale particulate flows, which are relevant for many industrial applications.

http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/1445 gives this article, © Austral. Mathematical Soc. 2008. Published October 29, 2008. ISSN 1446-8735. (Print two pages per sheet of paper.)

Contents

1 Introduction

The transport of solid particles and objects suspended in a fluid is crucial for different physical and industrial processes and a detailed understanding of the transport processes is becoming more important. In previous years many numerical methods used to simulate particulate flows have appeared in the literature, for example the method of Stokesian dynamics [4], Euler–Lagrangian methods [12], distributed Lagrange multiplier methods [22] and the family of discrete element methods [9, e.g.], just to mention a few. Approaches based on the Lattice Boltzmann method have been presented by Ladd [17, 18], Aidun et al. [1] and Qi [27]. Most of these methods do not attempt a fully resolved simulation of the fluid-structure interaction between solid particles and fluid, rather they use approximations to reduce the compu-

tational complexity. Some of them can only be applied in special flow regimes (for example potential or Stokes flow) or they are using approximations of particles as mass points.

In our approach we calculate the fluid flow using a Lattice Boltzmann method (see Section 2.1) while the motion of rigid objects and their interactions are evaluated by a 'physics engine' named pe (see Section 2.2). Since both methods can only handle parts of the physics they need to be extended and coupled together in order to simulate solid objects incorporated in the flow. For this reason the fluid flow of the Lattice Boltzmann method is used as input for the no-slip boundary condition of the objects resulting in hydrodynamic forces from the fluid to the objects, which are send to the physics engine. Due to these forces, the movement and frictional collisions between the rigid bodies are handled by pe. Then the moving objects are influencing the fluid flow through Lattice Boltzmann boundary conditions. The complete numerical method is incorporated in a software framework called waLBerla, which is covered in Section 3. In the simulation the objects are not treated as mere point masses, but are fully resolved as geometric entities within the flow. For the entire simulation the same mesh is used, eliminating the need for remeshing. Compared to other methods, where an automatic generation of a body fitted mesh during the simulation is a difficult problem, this fixed mesh is a great advantage in terms of performance.

2 Numerical methods

In this section we present the numerical algorithms used to simulate a large number of moving objects incorporated in the fluid using an extended Lattice Boltzmann solver for the fluid flow and a physics engine for the collision and movement of the objects. Furthermore, the parallelisation of this approach is shown in detail.

2.1 The Lattice Boltzmann method

The Lattice Boltzmann method (LBM) is an alternative to classical Navier–Stokes solvers for fluid flow and works on an equidistant grid of cells, called lattice cells, which only interact with their direct neighbours. In this study we use the common three dimensional D3Q19 model originally developed by Qian, d'Humiéres and Lallemand [28] with $N = 19$ particle distribution functions (PDFs) $f_\alpha : \Omega \times T \mapsto [0;1)$, where $\Omega \subset \mathbb{R}^3$ and $T \subset \mathbb{R}$ are the physical and time domain, respectively, and the corresponding dimensionless discrete velocity set $\{\mathbf{e}_\alpha \mid \alpha = 0, \ldots, N-1\}$. This model has been shown to be both stable and efficient [19]. For the work presented in this article, we adopt a Lattice Boltzmann Bhatnagar–Gross–Krook (LBGK) collision scheme [2, 28]

$$f_\alpha(\mathbf{x}_i + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}_i, t) - \frac{1}{\tau}\left[f_\alpha(\mathbf{x}_i, t) - f_\alpha^{(eq)}(\mathbf{x}_i, t)\right], \qquad (1)$$

where $\mathbf{x}_i$ is a cell in the discretised simulation domain, τ is the relaxation time, t is the current time step whereas $t + \Delta t$ is the next time step, and $f_\alpha^{(eq)}$ represents the equilibrium distribution, which in an incompressible LBGK scheme reads [10]

$$f_\alpha^{(eq)}(\mathbf{x}_i, t) = w_\alpha \left[\rho(\mathbf{x}_i, t) + \rho_0 \left(3\mathbf{e}_\alpha \mathbf{u}(\mathbf{x}_i, t) + \frac{9}{2}(\mathbf{e}_\alpha \mathbf{u}(\mathbf{x}_i, t))^2 - \frac{3}{2}\mathbf{u}(\mathbf{x}_i, t)^2\right)\right].$$

Here, we choose $\rho_0 = 1$. The weighting factors w_α are defined for the D3Q19 discretisation scheme as

$$w_\alpha = \begin{cases} 1/3, & \mathbf{e}_\alpha = (0,0,0) \\ 1/18, & \mathbf{e}_\alpha = (\pm 1,0,0),\ (0,\pm 1,0),\ (0,0,\pm 1) \\ 1/36, & \mathbf{e}_\alpha = (\pm 1,\pm 1,0),\ (\pm 1,0,\pm 1),\ (0,\pm 1,\pm 1)\,. \end{cases}$$

The macroscopic fluid density ρ and macroscopic velocity $\mathbf{u}$ are calculated from the first two moments of the distributions

$$\rho(\mathbf{x}_i, t) = \rho_0 + \delta\rho(\mathbf{x}_i, t) = \sum_\alpha f_\alpha(\mathbf{x}_i, t)\,, \qquad (2)$$

$$\mathbf{u}(\mathbf{x}_i, t) = \frac{1}{\rho_0} \sum_\alpha \mathbf{e}_\alpha f_\alpha(\mathbf{x}_i, t)\,. \tag{3}$$

Equation (1) is separated into two steps, known as the collision step and the streaming step, respectively

$$\tilde{f}_\alpha(\mathbf{x}_i, t) = f_\alpha(\mathbf{x}_i, t) - \frac{1}{\tau}\left[f_\alpha(\mathbf{x}_i, t) - f_\alpha^{(eq)}(\mathbf{x}_i, t)\right]\,, \tag{4}$$

$$f_\alpha(\mathbf{x}_i + \mathbf{e}_\alpha \Delta t, t + \Delta t) = \tilde{f}_\alpha(\mathbf{x}_i, t)\,, \tag{5}$$

where $\tilde{f}_\alpha$ denotes the post-collision state of the distribution function. While the collision step is a local single time relaxation procedure towards equilibrium and is compute intensive, the streaming step advects all PDFs besides f_0 to their neighbouring lattice site depending on the velocity, which is a memory intensive operation.

As a first order no-slip boundary condition often a simple bounce back scheme is used, where distribution functions pointing to a neighbouring wall cell are just reflected such that both normal and tangential velocities vanish

$$f_{\bar{\alpha}}(\mathbf{x_f}, t) = \tilde{f}_\alpha(\mathbf{x_f}, t)\,, \tag{6}$$

with $\bar{\alpha}$ representing the index of the opposite direction of α, $\mathbf{e}_{\bar{\alpha}} = -\mathbf{e}_\alpha$, and $\mathbf{x_f}$ explicitly denoting the fluid cell.

More details on the Lattice Boltzmann algorithm and its derivation are found in the book of Succi [31] and in the article by Chen et al. [5].

2.2 The rigid body physics engine pe

In order to simulate the rigid objects in the flow, we use a rigid body physics engine. The term physics engine is commonly used in virtual reality, robotics and computer games communities and describes a software framework that

simulates the physics related to the movement and interactions of rigid, completely undeformable objects (the rigidity assumption). In contrast to molecular dynamics simulations, where only point masses are used and repulsive potentials replace real collisions, the rigid bodies used in this approach have a physical expansion and both linear movements and rotations of the arbitrarily formed objects are treated. Elastic and/or inelastic collisions between rigid objects occur if the geometries of two objects touch; such collisions have to be handled to keep the objects from penetrating each other.

The decision to use such a framework is easily justified by our requirement to allow arbitrarily shaped objects (and not only spheres, which would be easy to handle) and the complex treatment of collisions between these rigid objects. Most openly available physics engines are programmed for real time simulations due to a focus on virtual reality environments or computer games [20], which results in the use of faster algorithms that unfortunately sometimes only crudely approximate the real physics. Consequently, we decided to implement a new rigid body physics engine primarily focussing on the physically accurate simulation of moving objects.

The pe engine offers everything that well established physics engines offer, but uses appropriate algorithms to solve the movement and especially the collisions of objects as accurately as possible [26, 29]. The movement of objects for instance is treated by a Störmer–Verlet time discretisation of Newton's equations.

The major problem for the simulation of rigid objects is the treatment of collisions. One goal of pe is the implementation of an interface that allows for the integration of arbitrary collision treatment algorithms (including algorithms for computer games). The majority of the algorithms for the physically accurate treatment of collisions are handling the following linear

complementarity problem [6]

$$\begin{pmatrix} D^T J^T M^{-1} J D & D^T J^T M^{-1} J N & E_\eta \\ N^T J^T M^{-1} J D & N^T J^T M^{-1} J N & 0 \\ -E_\eta^T & \mu & 0 \end{pmatrix} \cdot \begin{pmatrix} \Delta t \vec{\beta} \\ \Delta t \vec{f_n} \\ \vec{\lambda} \end{pmatrix} + \begin{pmatrix} D^T J^T \left(\vec{v}^t + \Delta t M^{-1} \vec{f}_{ext} \right) \\ N^T J^T \left(\vec{v}^t + \Delta t M^{-1} \vec{f}_{ext} \right) \\ \vec{0} \end{pmatrix} \geq \vec{0}$$

$$\text{complementary to} \begin{pmatrix} \Delta t \vec{\beta} \\ \Delta t \vec{f_n} \\ \vec{\lambda} \end{pmatrix} \geq \vec{0}, \tag{7}$$

where N represents the matrix of all contact normals, D represents the matrix of all frictional tangents of the contact points, J is the Jacobian matrix and M the generalized mass matrix. The two matrices E_η and μ are defined as

$$\mu = \begin{pmatrix} \mu_1 & & 0 \\ & \ddots & \\ 0 & & \mu_K \end{pmatrix} \in \mathbb{R}^{K \times K} \tag{8}$$

$$E_\eta = \begin{pmatrix} \vec{e} & & \vec{0} \\ & \ddots & \\ \vec{0} & & \vec{e} \end{pmatrix} \in \mathbb{R}^{(\eta \cdot K) \times K}, \tag{9}$$

where K is the number of contact points and η the number of frictional components per contact point. The vector $\vec{f_n}$ is the solution vector of all normal components of the acting forces, the vector $\vec{\beta}$ represents all frictional components. $\vec{\lambda}$ is an auxiliary value to ensure the maximum dissipation of the friction, $\vec{v}^t$ are the relative contact velocities and $\vec{f}_{ext}$ the currently acting external forces. The size of a single time step is represented by Δt.

A classical solver for this kind of problem is, for example, the Lemke pivoting algorithm [6]. Unfortunately, the complexity of $O(N^4)$ for this al-

gorithm prohibits its application for a large number of rigid bodies. In order to cope with thousands of rigid bodies, we use the projected Gauss–Seidel algorithm [8], which has a complexity of $O(N^2)$ and is therefore better suited for our purposes. However, since this algorithm can only handle positive definite matrices (and most of the occurring positive semi-definite matrices), the original formulation of the linear complementarity problem has to be changed according to the requirements of this algorithm:

$$\begin{pmatrix} D^T J^T M^{-1} J D & D^T J^T M^{-1} J N \\ N^T J^T M^{-1} J D & N^T J^T M^{-1} J N \end{pmatrix} \cdot \begin{pmatrix} \Delta t \vec{\beta} \\ \Delta t \vec{f}_n \end{pmatrix} + \begin{pmatrix} D^T J^T \left(\vec{v}^t + \Delta t M^{-1} \vec{f}_{ext} \right) \\ N^T J^T \left(\vec{v}^t + \Delta t M^{-1} \vec{f}_{ext} \right) \end{pmatrix} \geq \vec{0}$$
$$\text{complementary to} \begin{pmatrix} \Delta t \vec{\beta} \\ \Delta t \vec{f}_n \end{pmatrix} \geq \vec{0}. \tag{10}$$

This formulation neglects the maximum dissipation requirement, which can result in frictional collision responses that are not directly opposing the causative velocities or accelerations. However, in our current simulations we experienced no restrictions due to this formulation.

2.3 Coupling Lattice Boltzmann to the physics engine

In order to simulate moving objects with the LBM, the basic algorithm for the fluid flow from Section 2.1 has to be extended and coupled to the physics engine described in Section 2.2. The first extension is the coupling of the objects to the fluid through the boundary treatment for objects. In our implementation each lattice node with a cell center inside an object is treated as a moving wall (depicted in Figure 1(a)). This results in the following boundary condition for the moving objects [33]

$$f_{\bar{\alpha}}(\mathbf{x_f}, t) = \tilde{f}_{\alpha}(\mathbf{x_f}, t) + 6 w_{\alpha} \rho_w \mathbf{e}_{\bar{\alpha}} \cdot \mathbf{u_w}\,, \tag{11}$$

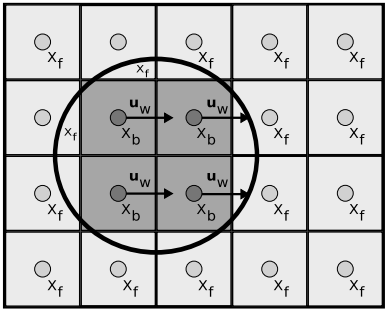
where ρ_w is the fluid density close to the wall and the current velocity $\mathbf{u_w}$ of each object cell corresponds to the velocity of the object at the cell's position. Thus, rotational as well as translational velocities of the objects are taken into account.

Furthermore, cell changes due to the movement of the objects must be modified, which is done by adjusting the boundary conditions of the objects in each LBM time step. Here, two cases occur (see Figure 1(b)): fluid cells $\mathbf{x_f}$ can turn into obstacles, which is treated by the conversion of the cell to a moving wall; conversely, when obstacle cells turn into fluid cells the missing distribution functions have to be determined. This is done by setting the missing PDFs to equilibrium distributions $f_\alpha^{(eq)}(\rho, \mathbf{u})$, where the macroscopic velocity $\mathbf{u}$ is set to the velocity $\mathbf{u_w}$ of the former object cell and the density ρ is averaged from the surrounding fluid cells. Iglberger et al. [14] described further details.

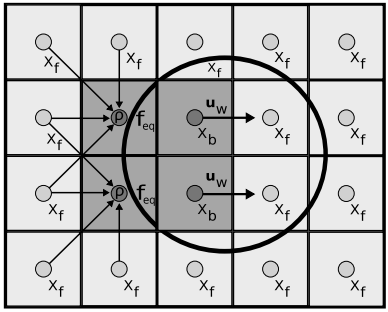
To couple the LBM fluid flow simulation to the objects in the flow, the hydrodynamic forces from the fluid to the objects need to be evaluated. One possible approach to calculate the forces in the LBM is the momentum exchange method [33], resulting in the acting force onto an object of

$$F = \sum_{\mathbf{x_b}} \sum_{\alpha=1}^{19} \mathbf{e}_\alpha \left[2\tilde{f}_\alpha(\mathbf{x_f}, t) + 6 w_\alpha \rho_w \mathbf{e}_{\bar{\alpha}} \cdot \mathbf{u_w} \right] \frac{\Delta x}{\Delta t}, \tag{12}$$

where $\mathbf{x_b}$ are all obstacle cells of the object neighbouring to at least one fluid cell. Iglberger et al. [14] discussed a verification of the method described in this section and Binder et al. [3] used the method to simulate the drag force on object agglomerates.



(a) Initial setup: The velocities $\mathbf{u}$ of the object cells $\mathbf{x_b}$ are set to the velocity $\mathbf{u_w}(\mathbf{x_b})$ of the object. In this example the object only has a translational velocity component. Fluid cells are marked with $\mathbf{x_f}$.



(b) Updated setup: Two fluid cells have to be transformed to object cells and for two object cells the PDFs have to be reconstructed.

FIGURE 1: 2D mapping example.

3 The waLBerla software framework

In computational fluid dynamics many applications of scientific interest share physical and computational aspects. In research environments the usual practice is one software for each application, leading to a reimplementation of the shared physics, the common data structures and also the parallelisation, which often requires a considerable effort. Therefore, waLBerla (which is an acronym for widely applicable Lattice Boltzmann from Erlangen) was developed and serves as a flexible framework, facilitating the introduction of extensions to the method to implement different computational fluid dynamics applications. It offers generic tools for data management, communication, and sequence control. For a 3D LBM using the D3Q19 discretisation model, standard basics are already incorporated such as the most common collision models (LBGK, TRT, MRT), force exertion and handling of boundary conditions.

Since the waLBerla framework aims at high computing performance, all tools are implemented in an optimised fashion suitable for large scale parallelisation. The domain is partitioned into smaller subregions of the fluid domain, called patches (see Section 4), which can support different kind of functionality, grid refinement, architecture optimized memory layouts, load balancing and heterogeneous computations.

Currently, the list of applications under development using waLBerla includes blood flow [30], moving objects [14], charged colloids [13], Brownian motion [21], free surfaces [23, 16], multi-component multi-phase and species transport models. In order to simulate moving objects we incorporated the physics engine pe from Section 2.2 in the waLBerla framework and extended the algorithms appropriately.

4 Parallelisation

The parallelisation concept needed for fluid structure interaction can be split into a LBM and a physics engine part. Although optimising and parallelising the LBM has been studied intensively [15, 25, 32], in particular with respect to cache and memory performance [7, 24, 34], the implementation of a flexible adaptable parallelisation required by a framework supporting several kind of functionality raises new problems.

In the waLBerla framework these problems are solved by our patch concept together with a generic MPI communication. During the domain partitioning the fluid is divided into patches, where several patches are assigned to each process. This procedure can be seen in Figure 2, which depicts two processes, both possessing 16 patches.

To reduce the startup time overhead, data is accumulated in byte buffers before being sent to neighbouring processors. The buffer is of datatype byte in order to be able to communicate floating point data as well as integer or

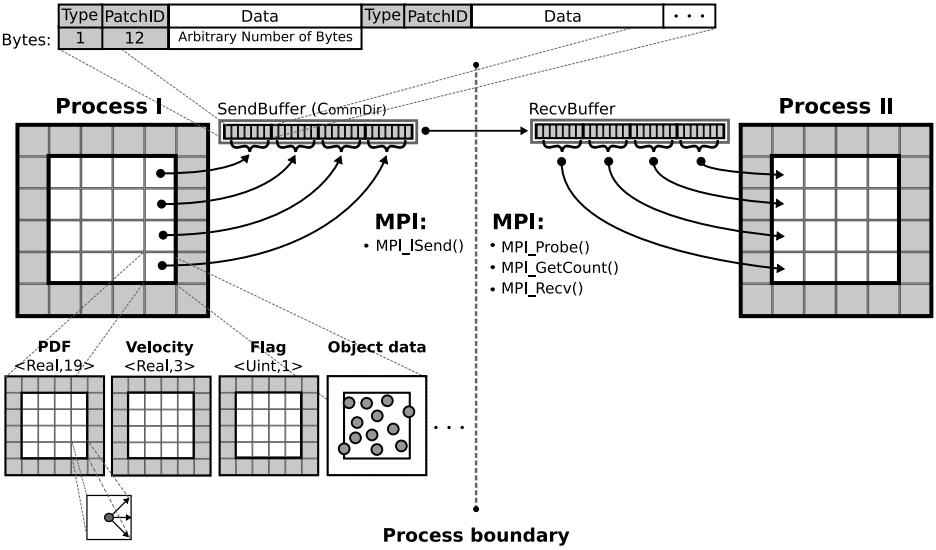


FIGURE 2: Parallelisation concept of the waLBerla project. Two processes are depicted, both having 16 patches. The communication is only shown for one lattice direction and only from process I to II. The message send from process I to process II is constructed from several smaller messages. Each submessage begins with a type information, like structured data or unstructured data and contains the data of one patch.

byte data.

Depending on the functionality, each patch may contain structured data like the data for the PDFs, velocities and cell state data (Flags) as well as unstructured data like the objects. For the structured data a constant amount of data has to be communicated to the neighbouring processes in each time step, but for the unstructured data messages of variable size must be sent. As Figure 2 shows, this is supported by sending the message of process I by an MPI_ISend() and by probing the message for its actual size with a MPI_Probe() on the receiving process II.

While the parallelisation of the LBM is comparatively straight forward, the parallelisation of the physics engine is still an ongoing research area. As a general problem, the rigidity assumption causes problems if rigid objects are in contact across several processes. Our approach for the parallel simulation of the coupled system therefore involves local pe instances for objects that can be handled locally and a global pe instance for the treatment of groups of rigid objects spanning several processes.

In case of a parallel simulation with N processes, $N-1$ processes (called "local workers") are responsible for the simulation of the fluid and the local objects, whereas one process (the global pe instance) deals with the remaining objects. In a parallel simulation the objects are distributed to the processes according to their physical position, which can result in objects that are cut by process interfaces.

As described in Section 2.2, within the physics engine a system of equations is set up for all objects in each time step. In our algorithm, all rigid objects, which can be treated locally (these cannot collide with objects from other processes) are treated by the local workers. Only objects on process interfaces and all objects which can collide with them in the current time step are computed globally by transferring them to the global pe instance. This instance resolves the collisions, computes the movement and sends the updated information back to the local processes. Objects approaching the boundary of the local domain are transferred to the appropriate neighboring process. The complete procedure is shown in Algorithm 1.

5 Performance results

Section 2 presented numerical methods to perform fluid simulations interacting with rigid bodies using an extended Lattice Boltzmann method coupled to a physics engine. During a parallel simulation, only objects cut by process interfaces are treated by one dedicated global pe instance (see Section 4).

Algorithm 1 Coupled LBM-pe solver

```
1:  for each time step do
2:      LBM step
3:      Add forces from fluid to rigid objects
4:      for each object do
5:          if Object near process interface then
6:              Send object to global pe instance
7:          else
8:              Treat object locally
9:          end if
10:     end for
11:     Move and collide objects on global pe instance
12:     Send objects back to local workers
13:     Update local objects
14:     if Object detected near process interface then
15:         Send object to neighboring process
16:     end if
17: end for
```

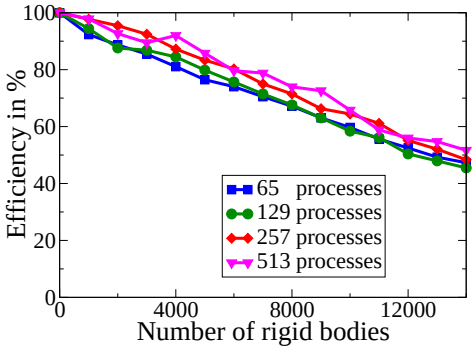
However, since all local workers must wait for the updated values from the global pe instance before proceeding with their work, the performance of this approach strongly depends on the number N_{global} of objects transferred and treated by the central process. This number is a function of the overall number of rigid bodies, their size and the lattice domain size of the local processes.

Efficiency measurements of our parallel algorithm to simulate rigid bodies incorporated in the flow, compared to simulations without rigid bodies are shown in Figure 3. These were run on the HLRB-2 SGI Altix system in Munich [11] with up to 1025 processor cores. The efficiency for a moderate number of objects is high (see the first scenario in Figure 3(a)), but drops down to 45% for around 14,000 objects due to the large number of objects treated by the central process. In order to simulate more rigid bodies efficiently, in the second scenario shown in Figure 3(b) the lattice domain size is larger, while the objects are smaller. When the overall number of rigid bodies or their size is increased N_{global} also increases, whereas enlarging the lattice domain size of the local processes leads to a reduction of N_{global}. Additionally, the simulation performs better when the number of rigid bodies in contact is low.

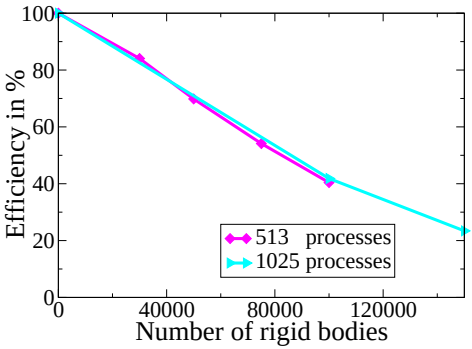
An example for simulating different types of rigid bodies in the flow is shown in Figure 4.

6 Conclusion

We have discussed some aspects of implementing a software framework for a large scale simulation of moving objects that is built from a combination of a Lattice Boltzmann fluid simulation and a rigid body physics engine. Using this coupled simulation system, we are able to simulate 150,000 objects as fully resolved rigid bodies in a fluid flow including rotation and frictional contacts. The scaling experiments show a strong dependency of the parallel



(a) The fluid domain size per process is $75 \times 75 \times 200$ lattice cells, spheres are used with a diameter of ten lattice cells.

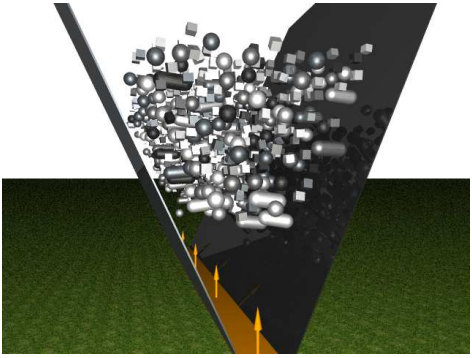


(b) The fluid domain size per process is 180^3 lattice cells, spheres are used with a diameter of six lattice cells.

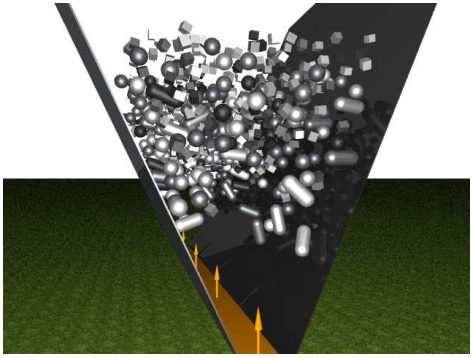
FIGURE 3: Efficiency of simulations with rigid bodies incorporated in the fluid compared to simulations of pure fluid flow.

efficiency on the number of objects that must be treated by a central physics engine process. To further improve the efficiency of the simulation either more objects have to be treated by the local processes, or the solution of the global contact problem must be parallelised in a suitable way.

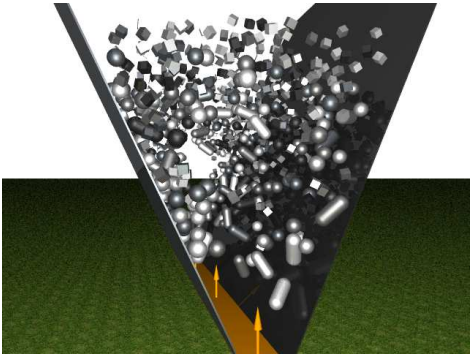
Acknowledgements The work was supported by the Kompetenznetzwerk für Technisch-Wissenschaftliches Hoch- und Höchstleistungsrechnen in Bayern (KONWIHR) under project FreeWIHR II, the European Union (EU) project DECODE (grant 213295) and the Deutsche Forschungsgemeinschaft (DFG) with the DIME project (grant RU422/7-5).



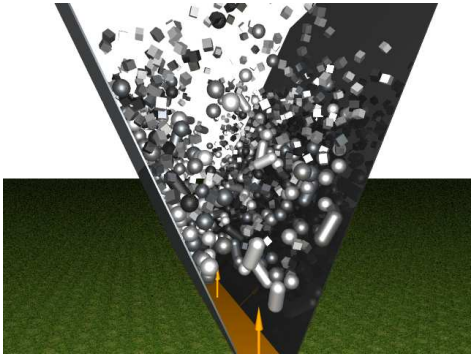
(a) At $t = 0\,s$.



(b) At $t = 1.5\,s$.



(c) At $t = 3\,s$.



(d) At $t = 4.5\,s$.

FIGURE 4: Example of a simulation with 1000 different objects in a horn using 31 processes and a fluid stream from bottom to top.

References

[1] C. K. Aidun, Y. Lu, and E.-J. Ding. Direct analysis of particulate suspensions with inertia using the discrete Boltzmann equation. *J. Fluid Mech.*, 373:287–311, October 1998. http://adsabs.harvard.edu/abs/1998JFM...373..287A. C167

[2] P. L. Bhatnagar, E. P. Gross, and M. Krook. A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Phys. Rev.*, 94(3):511–525, 1954. doi:10.1103/PhysRev.94.511. C169

[3] C. Binder, C. Feichtinger, H.J. Schmid, N. Thürey, W. Peukert, and U. Rüde. Simulation of the Hydrodynamic Drag of Aggregated Particles. *Journal of Colloid and Interface Science*, 301:155–167, Jan 2006. doi:10.1016/j.jcis.2006.04.045. C174

[4] J. F. Brady and G. Bossis. Stokesian Dynamics. *Annu. Rev. Fluid Mech.*, 20(1):111–157, 1988. doi:10.1146/annurev.fl.20.010188.000551. C167

[5] S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annu. Rev. Fluid Mech.*, 30:329–364, 1998. doi:10.1146/annurev.fluid.30.1.329. C170

[6] R. W. Cottle, J. S. Pang, and R. E. Stone. *The Linear Complementarity Problem.* Academic Press, 1992. C172

[7] S. Donath, K. Iglberger, G. Wellein, T. Zeiser, and A. Nitsure. Performance Comparison of Different Parallel Lattice Boltzmann Implementations on Multi-core Multi-socket Systems. *International Journal of Computational Science and Engineering (IJCSE)*, accepted for publication 2008. C176

[8] K. Erleben. *Stable, Robust, and Versatile Multibody Dynamics Animation*. PhD thesis, University of Copenhagen (DIKU), 2005. ftp://ftp.diku.dk/diku/image/publications/erleben.050401.pdf. C173

[9] M. Griebel, S. Knapek, and G. Zumbusch. *Numerical Simulation in Molecular Dynamics. Numerics, Algorithms, Parallelization, Applications*, volume 5 of *Texts in Computational Science and Engineering*. Springer Verlag, 2007. doi:10.1007/978-3-540-68095-6. C167

[10] X. He and L.-S. Luo. Lattice Boltzmann model for the incompressible Navier-Stokes equation. *J. Stat. Phys.*, 88:927–944, 1997. doi:10.1023/B:JOSS.0000015179.12689.e4. C169

[11] Information on the HLRB 2. http://www.lrz-muenchen.de/services/compute/hlrb/, Aug. 2008. C180

[12] K. Höfler, M. Müller, S. Schwarzer, and B. Wachmann. Interacting Particle-Liquid Systems. In E. Krause and W. Jäger, editors, *High Performance Computing in Science and Engineering '98*. Springer Verlag, 1998. C167

[13] J. Horbach and D. Frenkel. Lattice-Boltzmann method for the simulation of transport phenomena in charged colloids. *Phys. Rev. E*, 64(6):061507, 2001. doi:10.1103/PhysRevE.64.061507. C176

[14] K. Iglberger, N. Thürey, and U. Rüde. Simulation of moving particles in 3D with the Lattice Boltzmann method. *Comp. Math. Appl.*, 55(7):1461–1468, 2008. doi:10.1016/j.camwa.2007.08.022. C174, C176

[15] C. Körner, T. Pohl, U. Rüde, N. Thürey, and T. Zeiser. Parallel Lattice Boltzmann Methods for CFD Applications. In A.M. Bruaset and A. Tveito, editors, *Numerical Solution of Partial Differential*

Equations on Parallel Computers, volume 51 of *Lecture Notes for Computational Science and Engineering*, chapter 5, pages 439–465. Springer Verlag, 2005. doi:10.1007/3-540-31619-1¸3. C176

[16] C. Körner, M. Thies, T. Hofmann, N. Thürey, and U. Rüde. Lattice Boltzmann Model for Free Surface Flow for Modeling Foaming. *Journal of Statistical Physics*, 121:179–196, 2005. doi:10.1007/s10955-005-8879-8. C176

[17] A. J. C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. *J. Fluid Mech.*, 271:285–309, 1994. doi:10.1017/S0022112094001771. C167

[18] A. J. C. Ladd. Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 2. Numerical results. *J. Fluid Mech.*, 271:311–339, 1994. doi:10.1017/S0022112094001783. C167

[19] R. Mei, W. Shyy, D. Yu, and L.-S. Luo. Lattice Boltzmann Method for 3-D Flows with Curved Boundary. *J. Comp. Phys.*, 161:680–699, 2002. doi:10.1006/jcph.2000.6522. C169

[20] I. Millington. *Game Physics Engine Development*. Series in Interactive 3D Technology. Morgan Kaufmann, 2007. CD included. C171

[21] P. Neumann. Numerical simulation of nanoparticles in Brownian motion using the lattice Boltzmann method. Master's thesis, University of Erlangen-Nuremberg, Computer Science 10 – Systemsimulation, 2008. C176

[22] T.-W. Pan, D. D. Joseph, R. Bai, R. Glowinski, and V. Sarin. Fluidization of 1204 spheres: simulation and experiment. *J. Fluid Mech.*, 451:169–191, 2002. doi:10.1017/S0022112001006474. C167

[23] T. Pohl. *High Performance Simulation of Free Surface Flows Using the Lattice Boltzmann Method*. PhD thesis, University of Erlangen-Nuremberg, Computer Science 10 — Systemsimulation, July

2008. http://www10.informatik.uni-erlangen.de/Publications/Dissertations/Pohl_080713.pdf. C176

[24] T. Pohl, M. Kowarschik, J. Wilke, K. Iglberger, and U. Rüde. Optimization and Profiling of the Cache Performance of Parallel Lattice Boltzmann Codes. *Parallel Processing Letters*, 13(4):549–560, 2003. doi:10.1142/S0129626403001501. C176

[25] T. Pohl, N. Thürey, F. Deserno, U. Rüde, P. Lammers, G. Wellein, and T. Zeiser. Performance Evaluation of Parallel Large-Scale Lattice Boltzmann Applications on Three Supercomputing Architectures. Nov 2004. Supercomputing Conference 04. doi:10.1109/SC.2004.37. C176

[26] T. Preclik. Frictional Rigid Body Dynamics. Master's thesis, University of Erlangen-Nuremberg, Computer Science 10 — Systemsimulation, 2007. Computer Science Department 10 (System Simulation), University of Erlangen-Nuermberg. C171

[27] D. Qi. Lattice-Boltzmann simulations of particles in non-zero-Reynolds-number flows. *J. Fluid Mech.*, 385:41–62, April 1999. http://adsabs.harvard.edu/abs/1999JFM...385...41Q. C167

[28] Y. H. Qian, D. D'Humières, and P. Lallemand. Lattice BGK Models for Navier–Stokes Equation. *Europhysics Letters (EPL)*, 17(6):479–484, 1992. doi:10.1209/0295-5075/17/6/001. C169

[29] D. E. Stewart. *Impact and Friction of Solids, Structures and Machines*, chapter Time-stepping methods and the mathematics of rigid body dynamics. Birkhäuser, 2000. C171

[30] M. Stürmer, J. Götz, G. Richter, A. Dörfler, and U. Rüde. Fluid flow simulation on the Cell Broadband Engine using the lattice Boltzmann method. *Comp. Math. App.*, submitted 2007, accepted 2008, to be published. C176

[31] S. Succi. *The Lattice Boltzmann Equation—For Fluid Dynamics and Beyond*. Clarendon Press, 2001. C170

[32] G. Wellein, T. Zeiser, S. Donath, and G. Hager. On the single processor performance of simple lattice Boltzmann kernels. *Computer & Fluids*, 35(8–9):910–919, 2005. doi:10.1016/j.compfluid.2005.02.008. C176

[33] D. Yu, R. Mei, L.-S. Luo, and W. Shyy. Viscous flow computations with the method of lattice Boltzmann equation. *Prog. Aero. Sci.*, 39(5):329–367, 2003. doi:10.1016/S0376-0421(03)00003-4. C173, C174

[34] T. Zeiser, G. Wellein, A. Nitsure, K. Iglberger, U. Rüde, and G. Hager. Introducing a parallel cache oblivious blocking approach for the lattice Boltzmann method. *Progress in Computational Fluid Dynamics*, 8(1-4):179–188, 2008. doi:10.1504/PCFD.2008.018088. C176

Author addresses

1. **J. Götz**, Chair for System Simulation, Department of Computer Science, Friedrich–Alexander University of Erlangen–Nuremberg, 91058 Erlangen, Germany.
mailto:jan.goetz@informatik.uni-erlangen.de

2. **C. Feichtinger**, System Simulation, Department of Computer Science, Friedrich–Alexander University of Erlangen–Nuremberg, 91058 Erlangen, Germany.

3. **K. Iglberger**, System Simulation, Department of Computer Science, Friedrich–Alexander University of Erlangen–Nuremberg, 91058 Erlangen, Germany.

4. **S. Donath**, System Simulation, Department of Computer Science, Friedrich–Alexander University of Erlangen–Nuremberg, 91058 Erlangen, Germany.

5. **U. Rüde**, System Simulation, Department of Computer Science, Friedrich–Alexander University of Erlangen–Nuremberg, 91058 Erlangen, Germany.