VI International Conference on Computational Methods in Marine Engineering
MARINE 2015
F. Salvatore, R. Broglia and R. Muscari (Eds)

GPU-ACCELERATED LARGE-EDDY SIMULATION OF SHIP-ICE INTERACTIONS

DENNIS MIERKE, CHRISTIAN F. JANSSEN AND THOMAS RUNG

Institute for Fluid Dynamics and Ship Theory
Hamburg University of Technology (TUHH)
Am Schwarzenberg-Campus 4, 21073 Hamburg, Germany
E-Mail: {dennis.mierke,christian.janssen,thomas.rung}@tuhh.de
Web page: http://www.tuhh.de/elbe and http://www.tuhh.de/fds

Key words: LBM; Free Surface Flow; Physics Engine; FSI; GPGPU

Abstract. This paper reports on the applicability of the Lattice Boltzmann based free surface flow solver `elbe` to the simulation of complex ship-ice interactions in marine engineering. In order to model the dynamics of these colliding rigid multi-body systems, `elbe` is coupled to the ODE physics engine. First, basic validations of the ODE collision and friction models are presented, particularly focusing on interacting triangle meshes that later will serve to describe the ice floes. Then, the basic methodology and initial validation of the fluid-structure coupling of `elbe` and ODE is presented. Finally, performance is addressed: As `elbe` uses graphics processing units (GPUs) to accelerate the numerical calculations, the coupled numerical tool allows for investigations of ship-ice interactions in very competitive computational time and on off-the-shelf desktop hardware.

1 INTRODUCTION

In naval architecture and ocean engineering, efficient computational approaches based on accurate and robust modeling of viscous free surface flows and fluid-structure interactions are highly appreciated. This especially holds for the field of ship-fluid-ice interactions, where experimental studies are unwieldy, difficult to scale and typically very expensive. For this reason, we present a novel, very efficient numerical ice tank on the basis of the Lattice Boltzmann Method (LBM) for the viscous and turbulent flow field computations. The ice floe dynamics are modeled with a physics engine, considering effects of inertia, collisions and friction in floating and colliding multi-body systems of complex geometry.

The LBM has become a valuable alternative to conventional approaches, *e.g.*, Eulerian Finite-Volume methods or particle-based Lagrangian approaches (*e.g.*, SPH), for solving a variety of complex problems in computational fluid dynamics. The LBM solves

the discrete Boltzmann equation, which describes the evolution of particle distribution functions, typically on equidistant Cartesian grids. While Navier-Stokes procedures essentially model similar physics, LBM offers performance-related advantages, in terms of data locality and parallel computing. The employed in-house LBM solver `elbe` [1, 2, 3] uses graphics processing units (GPUs) to accelerate the hydrodynamic computations and allows for simulations in very competitive simulation time. The floating ice floe motions are described by the open-source Open Dynamics Engine (ODE) [4], that is coupled to the LBM in a bidirectional, explicit manner [5]. The ODE solves problems of rigid multi-body dynamics, including rigid constraints, friction and collisions in or near real-time.

After a short description of the basics of `elbe` and the fluid-structure coupling approach (Sec. 2), four selected validation cases are presented (Sec. 3) to show that the proposed numerical methodology is able to reproduce accurate results in a very competitive computational time. Finally, the applicability of the numerical `elbe` ice tank to a highly complex FSI problem is shown (Sec. 4).

2 NUMERICAL METHOD

The employed LBM solver `elbe`, the *efficient lattice boltzmann environment*, is a highly validated and efficient numerical tool for the simulation of two- and three-dimensional non-linear flow problems. While classical CFD solvers are based on the macroscopic Navier-Stokes equations, `elbe` handles CFD problems on a microscopic scale. Emphasis is given to marine- and coastal-engineering free-surface flows, such as wave breaking, tank sloshing or Tsunami propagation. The model considers non-linear flow behavior with and without a free surface, effects of viscosity and turbulence. For the implementation and parallelization, the NVIDIA CUDA toolkit is used, as it has been shown in various publications that LBM methods are especially well suited for implementation in a General Purpose Graphical Processing Unit (GPGPU) context [2]. Thanks to this very efficient numerical back end, three-dimensional simulations of complex flows are possible in very competitive simulation time.

2.1 Lattice Boltzmann Method

LBM's fundamental variable is the particle distribution function $f(t, \boldsymbol{x}, \boldsymbol{\xi})$, which specifies the probability to meet a fictive particle with velocity $\boldsymbol{\xi}$ at position $\boldsymbol{x}$ and time t. To obtain a model with reduced computational costs, the velocity space is discretized and discrete particle velocities $\boldsymbol{e}_i$ are introduced. In the present work, the D3Q19 model [6] with 19 discrete microscopic particle velocities $\boldsymbol{e}_i$ and corresponding particle distribution functions $f_i(t, \boldsymbol{x})$ is used. A subsequent standard finite difference discretization of the velocity-discrete Boltzmann equation on an equidistant Cartesian grid then leads to the *lattice* Boltzmann equation

$$f_i(t + \Delta t, \boldsymbol{x} + \Delta t\, \boldsymbol{e}_i) - f_i(t, \boldsymbol{x}) = \Omega_i \ , \tag{1}$$

where the left-hand side is an advection-type expression and the discrete collision operator Ω_i on the right hand side models the interactions of particles on the microscopic scale. For the latter, the advanced multiple relaxation time (MRT) model [7] is used in this work. The MRT transforms the particle distribution functions into moment space, where they are relaxed to an equilibrium state with several different relaxation rates. The benefits of this operator are an increased stability and the possibility to develop more accurate boundary conditions [8]. The solutions of the lattice Boltzmann equation satisfy the incompressible Navier-Stokes equations up to errors of $\mathcal{O}(\Delta x^2, \mathrm{Ma}^2)$ [9]. The first two hydrodynamic moments of the particle distribution functions include the macroscopic values for the density and momentum fluctuation:

$$\rho = \sum_{i=0}^{18} f_i \qquad \text{and} \qquad \rho_0 \, \boldsymbol{u} = \sum_{i=0}^{18} f_i \, \boldsymbol{e}_i \; . \tag{2}$$

A Smagorinsky large eddy model (LES) captures turbulent structures in the flow, including an additional turbulent viscosity ν_T for effects of unresolved sub-grid eddies [10]. Since after the advection step the incoming particle distribution functions at the domain boundaries are missing, they are reconstructed with the help of boundary conditions. For no-slip and velocity boundaries, a simple bounce back scheme is used [11]. Since sub-grid wall distances are not taken into account in this model, the scheme is only second-order accurate for boundaries which are exactly located in the middle of two lattice nodes. At the free surface, the anti-bounce-back rule [12] balances the fluid pressure and the surrounding atmospheric pressure. Volume forces like gravity are added directly to the distribution functions in every time step [13].

2.2 Free surface model

Free surface flows are immiscible two-phase flows which are dominated by the denser phase, due to high viscosity and density ratios between the two phases. If capillarity is neglected, the simulation of the denser phase is sufficient for many applications. The influence of the lighter phase on the flow dynamics can be approximated by kinematic and dynamic boundary conditions at the interface. Numerically, the free surface represents a moving boundary, which is allowed to move freely, but at the same time has to be kept sharp. A common and straightforward way to simulate free surface flows in the scope of LBM is the combination of a volume of fluid (VOF) method and a flux-based mesoscopic advection scheme [12]. In contrast to common VOF methods, the flux terms are expressed directly in terms of LBM distribution functions. The VOF approach captures the interface via the fluid fill level ε of a cell: $\varepsilon = 0.0$ marks an empty cell in the inactive gas domain and $\varepsilon = 1.0$ corresponds to a filled cell inside the fluid domain. Fluid and gas cells are separated by a closed interface layer with a fill level $\varepsilon \in (0.0; 1.0)$. The new fill level of a cell is calculated by balancing the mass fluxes between the neighboring cells and updating the fill level via

$$\varepsilon^{t+1} = \frac{m^{t+1}}{\rho^{t+1}} = \frac{\rho^t \, \varepsilon^t + \sum_i \Delta m_i}{\rho^{t+1}} \qquad \text{and} \qquad \Delta m_i = \Delta t \, A_i \, \left[f_I(\boldsymbol{x}, t) - f_i(\boldsymbol{x}, t) \right] \, , \tag{3}$$

where Δm_i describes the mass flux terms between neighboring cells, while f_I and f_i are particle distribution functions entering respectively leaving the corresponding cell. A_i denotes the wet area between two cells and is calculated on the basis of a simplified surface reconstruction, *e.g.* as the arithmetic mean of the fill levels of two neighboring cells. In contrast to higher-order schemes (such as presented in [2]), the normal vector information is not considered here. To sum up, the Lattice Boltzmann VOF advection scheme can be considered as a specialized, geometry-based VOF method on the basis of a mesoscopic advection model.

2.3 Fluid-structure interaction

Floating-body motions follow from a motion solver which converts the external and hydrodynamic forces exerted on the body into the equivalent motions. For the coupling of the explicit LBM and the motion solver, a bidirectional and explicit coupling approach is used [5]. First, the hydrodynamic loads on the rigid bodies are evaluated by integrating the stress tensor over the bodies' surface (stress-integration method, SIM) [14]. Since the rigid bodies are considered to be infinitely stiff and do not allow elastic or plastic deformations, the evaluation of the integral force on each obstacle is sufficient. Once the hydrodynamic loads are calculated, the force and torque information are transferred to the motion solver and the subsequent time step of rigid body motion is computed. The resulting displacements and velocities are passed back to `elbe`, where the geometries are updated and a modified bounce back scheme serves to incorporate the rigid body boundary velocity. Note that the bodies are represented by triangulated surface meshes and are mapped to the fluid field by our highly efficient in-house voxelizator [15], so that the grid update is up to two orders of magnitude faster than one flow field update.

2.3.1 Open Dynamics Engine

To solve for the rigid multi-body dynamics, the open-source Open Dynamics Engine [4] was selected. The ODE is a fully featured, stable, mature and platform independent high-performance library, which provides various joint types and an integrated rigid multi-body collision solver in consideration of frictional effects. ODE is well known for its high performance, free and easy access and the real-time simulation capabilities. It is mainly used in robotics and vehicle simulations, virtual reality environments, computer games and simulation tools. Several evaluations of ODE can be found in the literature, also in comparison with other public-domain physics engines [16, 17, 18]. In general, a good accuracy and performance is found. However, the collision detection/handling for triangulated surface mesh geometries – as discussed in our work – was not investigated

by other groups in detail yet. Moreover, several research projects extended the ODE with additional features, *e.g.*, with performance related parallelizations [19, 20], or higher computational efficiency, support for joint-dampening, solver robustness and a more accurate friction model [21]. These extensions, even though not embedded into the official ODE source code, impressively demonstrate the large applicability and extensibility of the ODE.

ODE splits the collision handling into two separate parts, the collision *detection* and the collision *handling*. Two different collision detection libraries are available [22, 23], that highly efficiently detect collisions between geometric primitives (*e.g.*, spheres, boxes, cones, cylinders, capsules and planes), convex and triangulated mesh geometries, and heightfields. Both methods approximate the contact area of the colliding geometries based on a set of contact points, the intersection depth and the normal direction for each contact point. Once calculated, the contact surface can be further configured by additional physical properties, *e.g.*, and among others, the friction coefficient μ and the restitution coefficient k.

To accelerate the collision detection, it is split up into two phases. First, it is tested if the axis-aligned bounding boxes (AABB) of the geometries under consideration overlap (broad phase). If this is the case, the selected collision detection library tests for actual intersections of the two putatively colliding geometries (narrow phase). In certain situations, further efficiency improvements can be gained by using one of four different hierarchical bounding spaces or an arbitrary combination of them.

After the detection of intersecting body geometries and the composition of the corresponding contact points, the simulation advances in time by solving the impulse-based equations of motion as a linear complementary problem (LCP). Note that, while the geometrical constraints are treated as bilateral joints, the detected collisions are handled via a hard contact model by temporarily adding additional unilateral joints. Since collisions occur only among non-deformable rigid bodies, the equations of motion can be reduced to impulse-based motion equations in the following way:

First of all the governing constraint is the Newton-Euler equation $\boldsymbol{M}\,\boldsymbol{a} = \boldsymbol{F}_{\mathrm{E}} + \boldsymbol{F}_{\mathrm{C}}$, where $\boldsymbol{F}_{\mathrm{E}}$ donates the external and $\boldsymbol{F}_{\mathrm{C}}$ the constraint forces. The latter results in $\boldsymbol{F}_{\mathrm{C}} = \boldsymbol{J}^T\boldsymbol{\lambda}$, where $\boldsymbol{J}$ is the constraint Jacobian and $\boldsymbol{\lambda}$ presents the unknown Lagrange multipliers [24, 25]. Additionally, the rigid body constraints are described by $\boldsymbol{J}\boldsymbol{v} = \boldsymbol{c}$, with $\boldsymbol{c}$ as the time derivative of the constraint equations. Afterwards, a backward Euler time discretization for the acceleration term $\boldsymbol{a}^{t+1}$ and rearranging yields the LCP

$$\left[\boldsymbol{J}^t\boldsymbol{M}^{-1}\boldsymbol{J}^{t^T}\right]\boldsymbol{\lambda}^{t+1} = \frac{\boldsymbol{c}^{t+1}}{\Delta t} - \boldsymbol{J}^t\left[\frac{\boldsymbol{v}^t}{\Delta t} + \boldsymbol{M}^{-1}\boldsymbol{F}_{\mathrm{E}}^t\right] , \tag{4}$$

where $\lambda \geq 0$ and $c \geq 0$ counts for unilateral constraints [21, 26]. The equation has the form $\boldsymbol{A}\,\boldsymbol{\lambda} = \widetilde{\boldsymbol{c}} + \boldsymbol{b}$, where $\boldsymbol{\lambda}$ and $\widetilde{\boldsymbol{c}}$ are complementary. ODE provides two methods to solve the LCP, a numerical exact and a iterative one. The first is based on the Danzig algorithm [27] with optimizations and is used in the present paper. The other uses the

projected Gauss-Seidel (PGS) method with successive overrelaxation (SOR) [21, 28]. With $\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + \Delta t\, \boldsymbol{v}^{t+1}$ the entire motion solver has a semi-implicit manner [29].

ODE's dry friction model is stringently implemented into the LCP and linearizes Coulombs friction law $|\boldsymbol{F}_\mathrm{T}| \leq \mu |\boldsymbol{F}_\mathrm{N}|$. Cumulatively, the normal force $\boldsymbol{F}_\mathrm{N}$ is calculated under the assumption that no friction occurs. Under these approximations the friction cone is decimated to an anisotropic friction pyramid [24]. Additionally, ODE supports rolling friction around all tree axes.

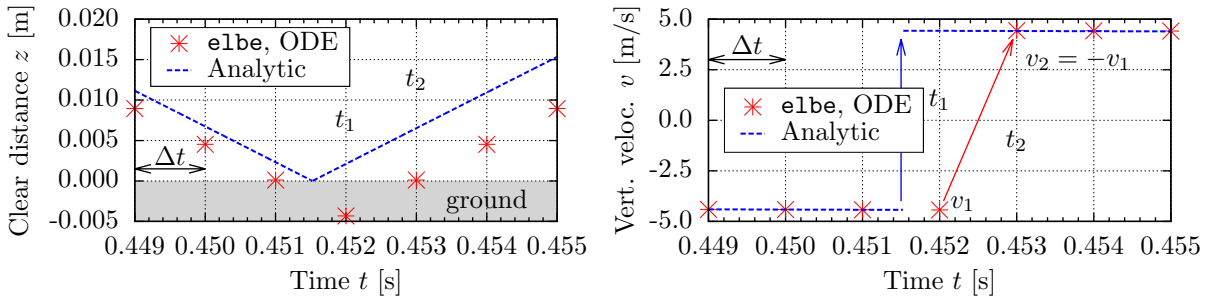
Since Eq. (4) does not necessarily hold the constraint equations in state, ODE provides an additional error reduction parameter (ERP) based on the penalty principle. Furthermore, constraint force mixing (CFM) provides non-hard constraints by dominating the diagonally of the system matrix. Further details can be found in [24].

3 RESULTS

In the following, the numerical results for four validation cases are given. The in-house flow solver `elbe` [1] in combination with the ODE release v0.13 is used. The first three test cases were run in a dry mode of `elbe` with a very limited number of lattice nodes. Since none of the previously published ODE validations investigated triangulated mesh geometries in detail, they are the primary focus of our work.

3.1 Collision detection & handling of a falling basketball

The first test case is concerned with the detection and handling of collisions between a freely falling body and a rigid ground plane. The basketball-like rigid body ($m = 1\,\mathrm{kg}$, $r = 0.5\,\mathrm{m}$, $z_0 = 1\,\mathrm{m}$) is represented by an icosphere with 5120 triangles. The gravity is set to $g = 9.81\,\mathrm{m/s^2}$ and the collision is fully elastic with a restitution coefficient of $k = 1.0$. The time step size is chosen to be $\Delta t = 10^{-3}\,\mathrm{s}$.



(a) Clear distance to ground z (b) Vertical velocity v

Figure 1: Collision handling for an elastic collision ($k = 1.0$)

The results follow the expectations: The basketball oscillates in a parabolic shape with a relative error in time of $\approx 0.3\,\%$ due to the implicit time integration scheme of first order. The penetration depth d is less than 1% of the ball diameter ($d \leq 0.005\,\mathrm{m}$). Fig. 1 shows the clear distance z of the basketball to the ground (left) as well as the vertical

velocity around the first ground contact (right). It can be seen that the first geometry intersection occurs at discrete time step t_1 and is detected and handled at the subsequent time step, the discrete time step t_2. This delay supplies the system with additional kinetic energy, mainly for two reasons: Firstly, the velocity inversion occurs at t_2 instead of t_1, see Fig. 1b. Since the velocity is inverted numerically exact ($v_2 = -v_1$), no external loads are considered for this time step and, consequently, the velocity magnitude is increased by $\Delta v_2 = g\,\Delta t$. Secondly, the delayed collision detection leads to an additionally falling distance d^* and therefore a further increase in velocity of $\Delta v_1 \leq g\,\Delta t$ at t_1 right before collision detection. Ultimately after the collision handling, the rigid body velocity is increased by $g\,\Delta t \leq \Delta v \leq 2\,g\,\Delta t$. As a consequence, a small amount of kinetic energy is added to the system every time a collision takes place, mainly due to the discrete time approximations of first order and the delayed collision detection and handling.

3.2 Friction model on a 2DOF inclined plane

In the following, the friction model is tested. A triangulated box on a 2DOF inclined plane is considered. The plane is tiltable around two axes and it is observed for which inclination (θ, φ) the box starts to slide, for a constant friction coefficient $\mu = 0.5$. For $F_\mathrm{T} < \mu\,F_\mathrm{N}$, the box is expected not to move (static friction). At $F_\mathrm{T} = \mu\,F_\mathrm{N}$, sliding occurs. Fig. 2 shows the results for 360 numerical tests. As due to numerical errors, the box is never completely at rest, the exact determination of permanent sliding is not possible. Therefore the results for two selected creep velocities are given in Fig. 2. Nevertheless, the anisotropy of the friction method and the shape of the friction pyramid (in comparison to the real friction cone, dashed line) can clearly be seen. Moreover, the expected friction angle $\tan\theta = \mu = 0.5$ is reproduced very well, at least for box motions in axial directions.

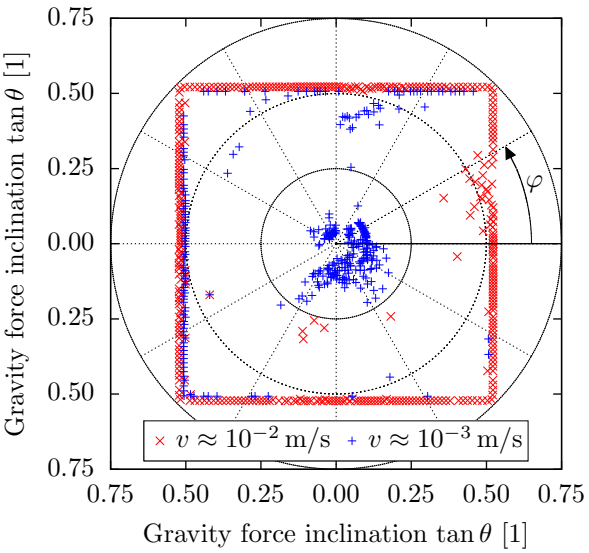


Figure 2: Anisotropic friction pyramid

3.3 Performance investigations

After the initial physical validations, the third test case serves to investigates the calculation costs of ODE's collision method. Two complex triangulated mesh geometries are considered [32]. One geometry is fixed and the second geometry is freely falling and accelerated towards the first one by gravitational forces. To achieve as much contacts

as possible in time the restitution coefficient is $k = 0.01$. The time step size is set to $\Delta t = 10^{-5}$ s. A total of five different surface mesh sizes were tested. In the first simulation, both surface meshes consists of 10 960 triangles. In the following runs, the triangle count is increased quadratically by the factors 2^2, 3^2, 4^2 and 5^2. Two time-consuming operations have to be considered: (i) the collision detection and (ii) the collision handling.

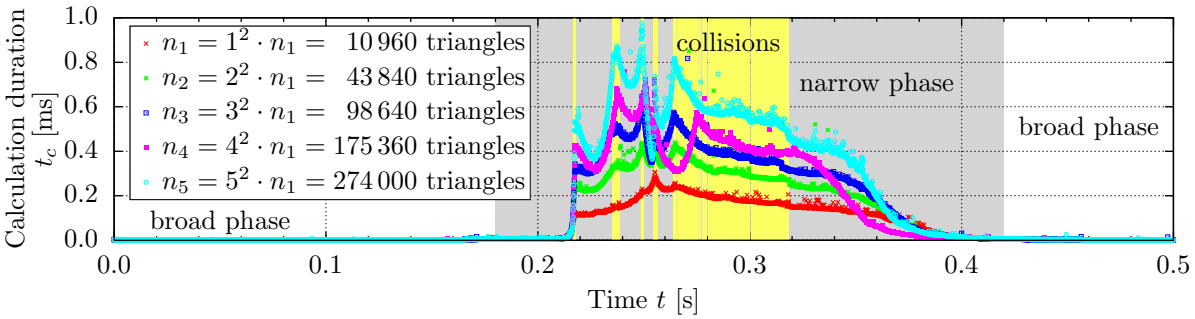


Figure 3: Performance of ODE's collision detection method, coupled to `elbe`, $\Delta t = 10^{-5}$ s.

Fig. 3 shows the detailed time measurements of the collision detection part of the computation, plotted against the total simulation time t. In the *broad phase*, the computational costs are almost zero (beginning and end of the simulation), while the computational costs in the *narrow phase* (once the AABBs intersect) are significantly higher (middle part, dyed gray, $0.18\,\text{s} < t < 0.42\,\text{s}$). When the actual collisions take place, the calculation time rises again significantly (dyed yellow). Two fundamental conclusions can be drawn here: First, since the calculation times of the five simulations nearly increase linearly, while the triangle count n increases quadratically, the computational cost of the collision detection is approximately of $\mathcal{O}(\sqrt{n})$. Second, as a triangle mesh refinement has no direct impact on the size of the LCP itself, the computational cost of the collision handling remains constant for all five cases, $t \approx 0.01$ ms.

3.4 Colliding and floating ducks

For a first validation of the explicit coupling of `elbe` and ODE's motion solver, a complex test case involving hydrodynamic and rigid multi-body interactions of colliding and floating bodies is examined. The simulation deals with five bodies falling into a water basin with a rough ground, see Fig. 4. The bodies are again represented by triangulated mesh geometries [32] with 10 960 triangles each, while the ground consists of 8953 triangles.

Fig. 5 shows results for selected time steps. During the simulation of $t = 2.5$ s, ODE detected and handled 51 707 collisions with none more than $d < 2 \times 10^{-5}$ m intersection depth.

More results can be found in [5] and will be presented at the conference, including additional quantitative investigations of coupled FSI simulations and more detailed comparisons of `elbe` results and experimental and numerical reference data.

Parameter	Value
Domain	$1 \times 1 \times 0.5\,\mathrm{m}^3$
Lattice	$300 \times 300 \times 150$
Δx	$\approx 0.0033\,\mathrm{m}$
Δt	$\approx 10^{-5}\,\mathrm{s}$
Ma	0.05
Re	1.5×10^6
Rubber ducks	$15 \times 15 \times 10\,\mathrm{cm}^3$
Water height h	$0.25\,\mathrm{m}$
Friction coeff. μ	0.5
Restitut. coeff. k	0.5

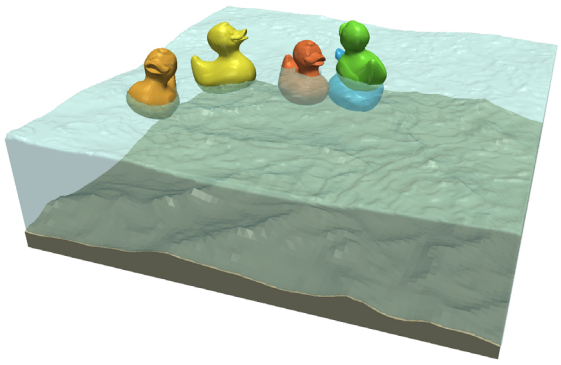
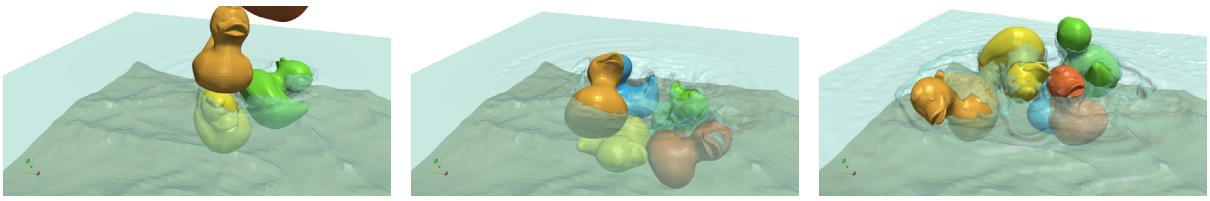


Figure 4: Simulation parameters & result at $t = 2.5\,\mathrm{s}$.



(a) $t = 0.35\,\mathrm{s}$ (b) $t = 0.50\,\mathrm{s}$ (c) $t = 1.00\,\mathrm{s}$

Figure 5: Colliding and floating ducks for selected time steps, $\Delta t \approx 10^{-5}\,\mathrm{s}$.

4 APPLICATION

Finally, after the basic validations, the code is applied to a full-scale ship-ice interaction problem, involving an ice-going ship and several ice floes. The hull of the German ice-going ship Polarstern (IMO 8013132) is used and discretized with 257 438 triangles. Each ice floe is described by 480 triangles. The ship hull is fixed in the numerical ice tank and subjected to an incoming flow of $v = 8\,\mathrm{m/s} \approx 15.6\,\mathrm{kn}$. The ice floes then are injected upstream of the ship and are advected towards the ship, where they finally start interacting with the hull. Further simulation parameters and a selected snapshot of the simulation can be found in Fig. 6.

Parameter	Value
Domain	$180 \times 70 \times 25\,\mathrm{m}^3$
Lattice	$1008 \times 392 \times 140$
Δx	$\approx 0.179\,\mathrm{m}$
Δt	$\approx 1.7 \times 10^{-4}\,\mathrm{s}$
Ma	0.05
Re	1.5×10^9
Friction coeff. μ	0.05
Restitut. coeff. k	0.05

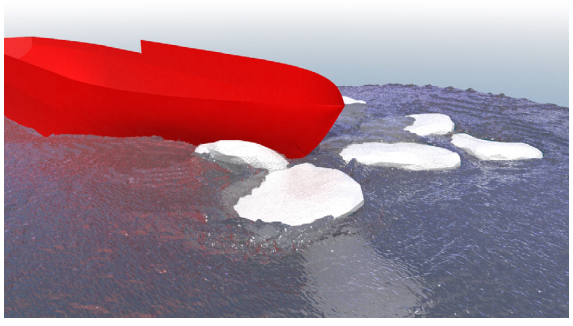


Figure 6: Application to ship-ice interactions: Ice-going *Polarstern*. Simulation parameters (left) and snapshot of the simulation at $t = 10\,\mathrm{s}$ (right)

5 CONCLUSION

In this paper, the validation of the Lattice Boltzmann based fluid solver `elbe` for free surface flow problems involving non-linear interactions of fluid and rigid multi-body systems was presented. The model uses a VOF interface capturing approach and is coupled to a physics engine – including collision detection and collision handling – in a bidirectional, explicit manner. Four validation test cases with complex geometries of triangulated meshes were addressed. The results demonstrate that the proposed numerical methodology is generally able to produce accurate results for three-dimensional FSI applications, such as floating and colliding multi-body systems.

Once a few remaining challenges, such as a local grid refinement around the ice floes and ice breaking, have been addressed, the coupled `elbe`-ODE solver will represent a major breakthrough for ship-ice interaction simulations. Opposite to previous work, that, *e.g.*, employs potential flow or RANSE approximations of the hydrodynamic parts, results of the full 3D non-linear Navier-Stokes solutions with sub-grid turbulence closure are considered in our work. Moreover, the innovative GPU implementation allows for large-scale investigations of real-world engineering problems, without tedious access to supercomputers. Exemplarily, the simulation results for a large-scale numerical ice tank involving a complex ship hull and a couple of hundred ice floes will be presented at the conference.

The authors gratefully acknowledge support for this research from the NVIDIA Academic Partnership Program (APP).

REFERENCES

[1] C.F. Janßen, `elbe` – *efficient Lattice Boltzmann environment*, `http://www.tuhh.de/elbe`.

[2] C.F. Janßen and M. Krafczyk, A lattice Boltzmann approach for free-surface-flow simulations on non-uniform block-structured grids. *Computers & Mathematics with Applications*, **59**(7):2215–2235, 2010.

[3] C.F. Janßen and M. Krafczyk, Free surface flow simulations on GPUs using LBM. *Computers & Mathematics with Applications*, **61**(12):3549–3563, 2011.

[4] R. Smith, *ODE – Open Dynamics Engine*, `http://www.ode.org`.

[5] D. Mierke, *Coupling of a GPU-Accelerated Lattice-Boltzmann-Method to a Physics Engine to Simulate Colliding Multibody Systems* (in German), Project thesis, Hamburg University of Technology, 2014.

[6] Y.H. Qian, D. d'Humières and P. Lallemand, Lattice BGK Models for Navier-Stokes Equation. *Europhysics Letters*, **17**(6):479–484, 1992.

[7] D. d'Humières, I. Ginzburg, M. Krafczyk, P. Lallemand and L.S. Luo, Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Royal Society of London Philosophical Transactions Series A*, **360**(1792):437–451, 2002.

[8] I. Ginzburg and D. d'Humières, Multi-reflection boundary conditions for lattice Boltzmann models. *Physical Review E*, **68**(6):066614.1–066614.30, 2003.

[9] M. Junk and Z. Yang, Pressure boundary condition for the lattice Boltzmann method. *Computers & Mathematics with Applications*, **58**(5):922-929, 2009.

[10] M. Krafczyk, J. Tölke and L.S. Luo, Large-eddy simulations with a multiple-relaxation-time LBE model. *Modern Physics B*, **17**(1&2):33-39, 2003.

[11] X. He and L.S. Luo, Lattice Boltzmann model for the incompressible Navier-Stokes equation. *Journal of statistical Physics*, **88**(3-4):927–944, 1997.

[12] 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**(1-2):179–196, 2005.

[13] Z. Guo, C. Zheng and B. Shi, Discrete lattice effects on the forcing term in the lattice Boltzmann method. *Physical Review E*, **65**(4):046308.1–046308.6, 2002.

[14] S. Geller, *Ein explizites Modell für die Fluid-Struktur-Interaktion basierend auf LBM und p-FEM* (in German). PhD thesis, Fakultät Architektur, Bauingenieurwesen und Umweltwissenschaften der Technischen Universität Carolo-Wilhelmina zu Braunschweig, 2010.

[15] C.F. Janßen, N. Koliha and T. Rung, A fast and rigorously parallel surface voxelization technique for GPGPU-accelerated CFD simulations. *Communications in Computational Physics*, accepted for publication, Oct. 2014.

[16] A. Boeing and T. Bräunl, Evaluation of Real-time Physics Simulation Systems. *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, 281–288, 2007.

[17] J. Hummel, R. Wolff, T. Stein, A. Gerndt and T. Kuhlen, An Evaluation of Open Source Physics Engines for Use in Virtual Reality Assembly Simulations. *Advances in Visual Computing*, **7432**:346–357, 2012.

[18] I. Metrikin, A. Borzov, R. Lubbad and S. Løset, Numerical Simulation of a Floater in a Broken-Ice Field: Part II – Comparative Study of Physics Engines. *31st International Conference on Ocean, Offshore and Arctic Engineering (OMAE2012)*, **6**:477–486, 2012.

[19] J. Sreeram and S. Pande, Parallelizing a Real-Time Physics Engine Using Transactional Memory. *Euro-Par 2011 Parallel Processing*, **6853**:206–223, 2011.

[20] J. Reinders, Intel Threading Building Blocks: *Outfitting C++ for Multi-core Processor Parallelism*. O'Reilly Media, ISBN: 9780596514808, 2007.

[21] E. Drumwright, J. Hsu, N. Koenig and D. Shell, Extending Open Dynamics Engine for Robotics Simulation. *Simulation, Modeling, and Programming for Autonomous Robots*, **6472**:38–50, 2010.

[22] P. Terdiman, *OPCODE – Optimized Collision Detection*. http://www.codercorner.com/Opcode.htm.

[23] F. Leon, *Gimpact – Geometric tools for VR*. http://gimpact.sourceforge.net.

[24] K. Erleben, *Stable, robust, and versatile multibody dynamics animation*. PhD. thesis, University of Copenhagen, 2004.

[25] D. Baraff, Linear-time Dynamics Using Lagrange Multipliers. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 137–146, 1996.

[26] M. Anitescu and F.A. Potra, Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, **14**(3):231–247, 1997.

[27] R.W. Cottle and G.B. Dantzig, Complementary pivot theory of mathematical programming. *Linear Algebra and its Applications*, **1**(1):103–125, 1968.

[28] G. Arechavaleta, E. López-Damian and J.L. Morales, On the use of iterative LCP solvers for dry frictional contacts in grasping. *International Conference on Advanced Robotics*, 2009.

[29] D.E. Stewart and J.C. Trinkle, An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Inelastic Collisions and Coulomb Friction, *International Journal for Numerical Methods in Engineering*, **39**:2673–2691, 1996.

[30] Y. Zhao, L. Wang, F. Qiu, A. Kaufman and K. Mueller, Melting and flowing in multiphase environment. *Computers & Graphics*, **30**(4):519–528, 2006.

[31] J. Tölke, Implementation of a Lattice Boltzmann kernel using the Compute Unified Device Architecture developed by nVIDIA. *Computing and Visualization in Science*, **13**(1):29–39, 2010.

[32] willie, *Rubber Duck*, STL-file. [online, Sep. 2014] http://www.thingiverse.com/thing:139894.