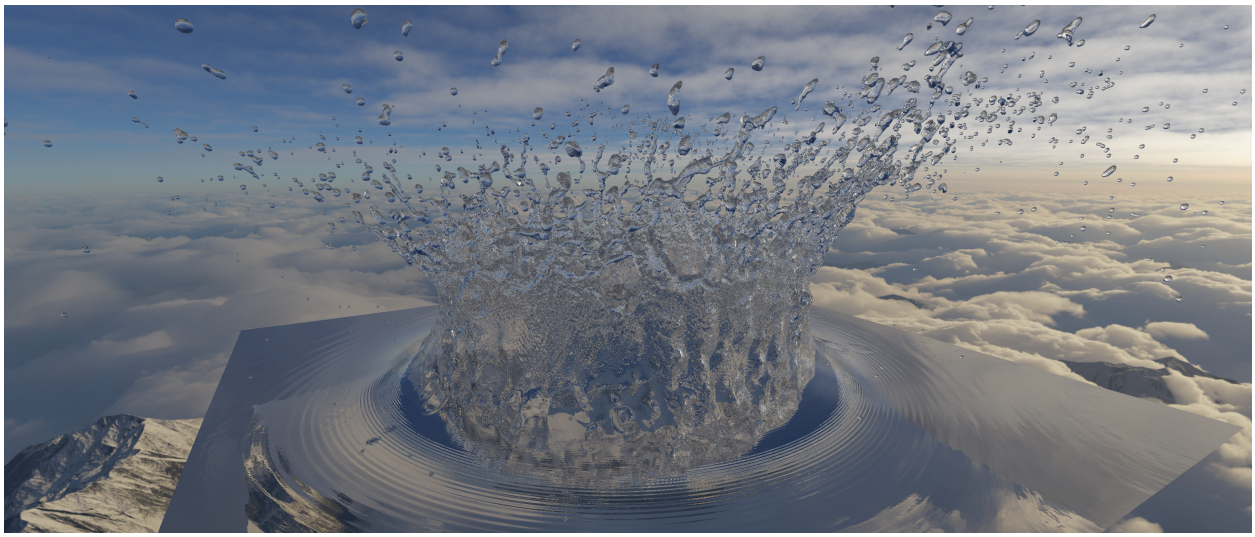
Computational study of microplastic transport at the water-air interface with a memory-optimized lattice Boltzmann method



DISSERTATION

submitted to obtain the academic degree
Doctor rerum naturalium (Dr. rer. nat.)

of the Bayreuth Graduate School of
Mathematical and Natural Sciences (BayNAT)
of the University of Bayreuth

by

Moritz Lehmann
born on April 16, 1997 in Bayreuth

Bayreuth, 2022

Biofluid Simulation and Modeling
Theoretical Physics VI
University of Bayreuth

UNIVERSITÄT BAYREUTH

This doctoral thesis was prepared at the department of Biofluid Simulation and Modeling – Theoretical Physics VI at the University of Bayreuth from January 2020 until November 2022 and was supervised by Prof. Dr. Stephan Gekle.

This is a full reprint of the thesis submitted to obtain the academic degree of Doctor of Natural Sciences (Dr. rer. nat.) and approved by the Bayreuth Graduate School of Mathematical and Natural Sciences (BayNAT) of the University of Bayreuth.

Date of submission: 29.11.2022

Approval by the governing board: 10.01.2023

Date of defense: 29.03.2023

Acting director: Prof. Dr. Hans Keppler

Doctoral committee:

Prof. Dr. Stephan Gekle (reviewer)
Prof. Dr. Jan Fleckenstein (reviewer)
Prof. Dr. Holger Kress (chairman)
Prof. Dr. Michael Wilczek

Additional reviewer: Prof. Dr. Mauro Sbragaglia

Abstract

Microplastics are a global pollutant. The microscopic particles can reach every place on the globe through wind and weather. Humans introduce ever more plastic via the rivers into the oceans, where it breaks down into microplastic. A part of it sinks to the ocean floor, a part remains in bulk, and a part floats on the ocean surface. I investigate how microplastic can get from the ocean surface into the atmosphere, to be transported by wind over vast distances, possibly even on land.

For the transition of particles from water to air, all mechanisms where small droplets are ejected are conceivable. Besides wave action during windy conditions and the bursting of bubbles on the surface, impacting raindrops could also contribute to the transfer. I study the latter mechanism in detail using computer simulations.

The simulation of an impacting raindrop allows precise measurement of each ejected spray droplet and counting contained particles. For statistical analysis of droplets and particles, I simulate many hundreds of these impacts. The concentration of particles in spray droplets resembles that at the water surface, meaning that the water from the raindrop, which is initially devoid of particles, for the most part does not get into droplets, but mainly sea water instead. Larger raindrops eject much more droplets and particles into the air, but occur much less frequent in nature than smaller raindrops. Using the Marshall-Palmer distribution of raindrop size and Rice-Holmberg distribution of rain intensity, I estimate the environmental relevance based on the simulation results: The transport of microplastic from the sea surface into the atmosphere by impacting raindrops is possible, especially during strong winds; but the estimated global flux of less than 10^{14} particles annually is small compared to other transport mechanisms like bursting of bubbles on the sea surface, and small compared to transport processes across other environmental compartments.

The particle concentration of microplastic at the sea surface, and why it is larger than in bulk, is of key importance for understanding water-air transport. One possibility how concentration at the surface can be enriched is when particles in bulk attach to rising air bubbles. I investigate this process in simulation specifically for microplastic, and find that aged particles are less affected by bubble scavenging compared to pristine particles which due to their hydrophobic surface better adhere to bubbles.

I also investigate the lateral transport of microplastics at the water-air interface in terrestrial environments, by simulating the flow profile of a thin water film on a rough plate, and I am able to explain the transporting behavior of particles on the plate with the microrelief.

The simulations in this thesis pose a computational workload so large that existing computational fluid dynamics (CFD) software could not handle it in a realistic time frame. Only with my self-written simulation software *FluidX3D* – based on the lattice Boltzmann method (LBM) with Volume-of-Fluid extension, and implemented in OpenCL for graphics processing units (GPUs) – this is feasible, due to the excellent computational efficiency and the very fast video memory of GPUs. Still the big issue with the LBM remains: the enormous memory demand that severely limits maximum grid resolution on GPUs. To reduce memory demand to 1/6 compared to two-grid FP64 implementations, I develop two new methods:

1) A new set of in-place streaming schemes to make the LBM require only a single copy of the computational grid in memory instead of two. There have already been solutions for this, like AA-Pattern and Esoteric-Twist, but these have never found widespread adoption due to several disadvantages. Based on the idea of Esoteric-Twist, I introduce two new streaming schemes termed Esoteric-Pull and Esoteric-Push that offer the same advantages, but are much simpler in implementation and are even slightly faster due to a more efficient memory access pattern.

2) Memory compression of the LBM density distribution functions to 16-bit number formats by separation of arithmetic precision and memory precision. After finding that FP32 produces the same accuracy as FP64 in all but corner cases with a detailed comparison study on six systems, I reduce memory precision to 16-bit with several arithmetic optimizations, and investigate how different floating-point and posit formats perform. With a re-scaled IEEE-754 FP16S, and a self-designed, more accurate FP16C format, the simulations succeed with only insignificant reduction in overall accuracy.

I also eliminate the very slow file export of gigantic volumetric datasets to the hard drive, by allowing to render simulation results directly in the fast video memory of the GPU and only needing to store image files and processed final simulation results.

The *FluidX3D* software resulting from my work, published on GitHub, reduces computation time significantly compared to existing CFD software. Possible applications extend far beyond my research on microplastics and cover aerospace, traffic, medical applications, industrial processes and countless areas of research.

Zusammenfassung

Mikroplastik ist ein Umweltschadstoff mit globaler Ausbreitung. Die mikroskopischen Partikel können durch Wind und Wetter an jeden Ort der Welt gelangen. Der Mensch bringt immer neues Plastik über die Flüsse in die Ozeane, wo es sich zu Mikroplastik zersetzt. Ein Teil davon sinkt auf den Meeresgrund, ein Teil verweilt in unterschiedlichen Tiefen, und ein Teil schwimmt auf der Meeresoberfläche. Ich untersuche wie Mikroplastik von der Wasseroberfläche in die Atmosphäre gelangen kann, sodass es als Aerosol vom Wind über große Distanzen, sogar zurück an Land, transportiert werden kann.

Für den Übergang der Partikel aus dem Wasser in die Luft sind alle Mechanismen denkbar, bei denen kleine Tröpfchen in die Luft geschleudert werden. Neben Wellenbewegung bei Wind und dem Platzen von Luftblasen an der Oberfläche könnte auch das Einschlagen von Regentropfen zum Transport beitragen. Letzteren Mechanismus erforsche ich im Detail mit Computersimulationen.

Die Simulation eines einschlagenden Regentropfens erlaubt die genaue Vermessung von jedem der hochgeschleuderten Tröpfchen und das Zählen der Partikel darin. Für eine statistische Erfassung der Tröpfchen und Partikel simuliere ich viele hunderte solcher Einschläge. Die Konzentration der Partikel in den Tröpfchen entspricht nahezu der an der Wasseroberfläche, was bedeutet, dass das von Partikeln freie Regenwasser zum Großteil nicht in die Tröpfchen geangt, sondern hauptsächlich Meerwasser. Größere Regentropfen schleudern deutlich mehr Tröpfchen und Partikel in die Luft, kommen jedoch in der Natur sehr viel seltener vor als kleine Regentropfen. Mit der Marshall-Palmer Verteilung der Regentropfengröße und der Rice-Holmberg Verteilung der Regenintensität schätze ich aus den Simulationsdaten die Umweltrelevanz ab: Der Transport von Mikroplastik von der Meeresoberfläche in die Atmosphäre durch einschlagende Regentropfen ist möglich, insbesondere bei starkem Wind; jedoch ist die geschätzte globale Menge von jährlich maximal 10^{14} Partikeln gering verglichen mit anderen Transportprozessen wie dem Platzen von Luftblasen an der Meeresoberfläche, sowie verglichen mit Transportprozessen in anderen Bereichen der Umwelt.

Von entscheidender Bedeutung für den Wasser-Luft Transport von Mikroplastik ist die Partikelkonzentration an der Meeresoberfläche, und warum diese höher ist als die Konzentration in tieferen Schichten. Eine Möglichkeit, wie Partikel an der Oberfläche angereichert werden können, ist wenn sich diese an aufsteigende Luftblasen anheften. Ich untersuche diesen Prozess mit Simulationen speziell für Mikroplastik, und stelle fest, dass gealterte Partikel davon weniger betroffen sind als neue Partikel, die durch ihre hydrophobe Oberfläche eher an Blasen haften bleiben.

Ich untersuche außerdem den lateralen Transport von Mikroplastik an der Wasser-Luft Grenzfläche in terrestrischen Umgebungen, indem ich das Strömungsprofil eines dünnen Wasserfilms auf einer rauen Platte simuliere, und kann damit das Transportverhalten von Partikeln auf der Platte mit dem Mikrorelief erklären.

Die Simulationen in meiner Arbeit stellen einen enormen Rechenaufwand dar, der mit bisheriger computational fluid dynamics (CFD) Software nicht in realistischer Zeit zu bewältigen ist. Erst mit meiner selbst entwickelten Simulationssoftware *FluidX3D* – basierend auf der Lattice Boltzmann Methode (LBM) mit Volume-of-Fluid Erweiterung und implementiert in OpenCL auf Grafikkarten (GPUs) – gelingt die Durchführung dank herausragender Recheneffizienz und dem sehr schnellen Videospeicher moderner GPUs. Dennoch bleibt das große Problem der LBM bestehen: der enorme Speicherbedarf, durch den auf GPUs nicht allzu hohe Gitterauflösung möglich ist. Um den Speicherbedarf verglichen mit zwei-Gitter FP64 Implementierungen auf etwa 1/6 zu reduzieren, entwickle ich zwei neue Methoden:

1) Ein neues Schema für in-place streaming, damit die LBM nur eine Kopie des Gitters im Speicher benötigt anstatt zwei. Hierfür gab es bereits Lösungen wie AA-Pattern und Esoteric-Twist, die jedoch aufgrund unterschiedlicher Nachteile bisher keine große Verbreitung gefunden haben. Aufbauend auf der Idee von Esoteric-Twist führe ich zwei neue Streaming-Schemata names Esoteric-Pull und Esoteric-Push ein, die dieselben Vorteile bieten, jedoch erheblich einfacher zu implementieren sind und dank effizienterem Speicher-Zugriffsmuster sogar etwas schneller sind.

2) Speicherkompression für die LBM density distribution functions in 16-bit Zahlenformate durch Trennung von arithmetischer Genauigkeit und Speichergenauigkeit. Nachdem ich in einer ausführlichen Vergleichsstudie anhand von sechs Systemen feststelle, dass FP32 bis auf in Ausnahmefällen die selbe Genauigkeit liefert wie FP64, reduziere ich mithilfe einiger arithmetischer Optimierungen die Speichergenauigkeit bis auf 16-bit, wobei ich unterschiedliche Fließkomma- sowie Posit-Formate untersuche. Mit skaliertem IEEE-754 FP16S und einem selbst entwickelten, genaueren FP16C Zahlenformat gelingen die Simulationen mit insgesamt keiner signigikanten Reduktion der Genauigkeit.

Zudem eliminiere ich den sehr langsamen Dateiexport riesiger volumetrischer Datensätze auf die Festplatte, indem ich die Simulationsergebnisse direkt im schnellen Videospeicher der GPU rendern kann, und nur noch die Bilddateien und Endergebnisse der Datenauswertung zu speichern brauche.

Die aus meiner Arbeit entstandene *FluidX3D* Software, veröffentlicht auf GitHub, verkürzt die Rechenzeit verglichen mit bestehender CFD Software erheblich. Die möglichen Anwendungen gehen weit über meine Forschung an Mikroplastik hinaus und beinhalten Luft- und Raumfahrt, Verkehr, Medizin, industrielle Prozesse und zahllose Bereiche der Grundlagenforschung.

Contents

Chapter 1

Introduction

Microplastics are an environmental pollutant that have spread to even the most remote places on the globe [1–5]. Being microscopic in size, they have efficient mobility in water and in the atmosphere as aerosol [1, 2, 5, 6]. Field experiments can detect the local concentrations in water and air, but how exactly microplastics transition between the aquatic and the atmospheric compartments is still not well understood. So far water bodies were considered a sink for microplastics [7–11], but recent observations indicate they can also act as a source [2, 6]. Once out of the water and in the air, mobility of microplastics is vastly increased [5] and the particles can even get back on land [6], possibly also reaching agricultural farmland. Moreover, frequent transition between water and air may have large effects on particle condition and further decomposition into smaller particles, until eventually entering the marine food chain as well [12].

In a similar process to the generation of sea salt aerosol – which is known to cause corrosion to infrastructure near the coastline [13, 14] – various tiny particles can make the transition from the ocean surface into the atmosphere [15–21], and recent experimental studies suggest that small microplastic particles are no exception [2, 6, 22–24]. I investigate this transition process at the water-air interface in microscopic detail using computer simulations, with the key question: How exactly can microplastics transition from water to air, and what mechanisms contribute how much?

The transfer can only possibly occur when sea spray is generated and spray droplets evaporate, leaving contained particles behind as an aerosol. Possible mechanisms are wave action, bubble bursting, and raindrop impacts. Spray production during wave action is a process happening across five orders of magnitude spatially, from meter-large waves to microscopic droplets, and unfortunately is still unfeasible to model in sufficient detail. The bubble bursting process is more accessible in experiments, studied in [22–24] and by us in [Pub7]; I only briefly estimate environmental relevance here based on our experimental findings. My main focus is on the third process, raindrop impacts, in [Pub1].

Particle transport during raindrop impacts represents a complex setup, for which it would be very difficult to provide sufficient data experimentally. Numerical simulations on the other hand allow for precise measurement of every single spray droplet, its velocity, and how many particles it might contain. Although numerical modeling of raindrop impacts still represents a major challenge, it has become feasible with the state-of-the-art Volume-of-Fluid lattice Boltzmann method (LBM) [25–37], running on current graphics processing

unit (GPU) technology, in quantities required for statistical analysis of ejected droplets. The memory capacity of GPU hardware allows for almost three orders of magnitude resolved in length scale, meaning is is possible to model an entire raindrop and at the same time accurately resolve the breakup of the crown rim into microscopic droplets. Subsequent modeling of their trajectory [38] and evaporation in air [39], depending on wind conditions [40], will reveal the fate of contained microplastic particles.

A key uncertainty of the water-air transport is the concentration of microplastics in the sea surface microlayer (SML). Microplastic concentration in the SML seems significantly enriched compared to bulk concentration [24, 41–43]. Besides buoyancy of polymers with lower density than water, even higher density particles can float to the surface if an air bubble sticks to them [44]. Interaction of rising air bubbles with particles – bubble scavenging – potentially increases water-air transfer rates during subsequent bubble bursting [45] and other transport mechanisms. In [Pub2], after thoroughly validating the bubble model [46] in [Pub6], I aim to find out how bubble scavenging works for microplastics, depending on their surface condition.

In a collaboration within the SFB 1357 in [Pub5], I also investigate lateral transport of microplastics at the water-air interface in terrestrial environments, when a film of water transports particles across a rough plate.

With an eye on future research, the second part of this thesis will focus on optimizations to enhance the capabilities of the employed LBM model significantly. The LBM on GPUs [28, 29, 31, 46–100] already is a very capable simulation tool, but the limited memory capacity of GPUs and the large memory demand of the LBM pose a major constraint on maximum grid resolution. To vastly reduce memory demand, two novel approaches are developed: First, in [Pub4] the optimal in-place streaming schemes are identified, building upon and improving existing solutions. In-place streaming allows to have only one copy of the density distribution functions (DDFs) in memory rather than two, which almost cuts memory demand in half. Next, in [Pub3] I take a very close look at what range of numbers is actually used by the DDFs, and then design custom 16-bit floating-point and posit formats with conversion algorithms, to compress the DDFs to half their size in memory, while only removing bits that carry no information or numerical noise.

Finally, in [Pub8] I eliminate the need for exporting large volumetric files to the hard drive by exploring combined simulation and rendering in OpenCL.

Chapter 2

Synopsis Part A: Application Studies

How can microplastics transition from the ocean surface into the atmosphere, and how environmentally relevant is this process? Experimental studies so far have pioneered in the pursuit to answer this question, measuring the concentration of microplastics at the ocean surface at various locations [9, 101–104], and even finding microplastics in the air at sea [2] and at the coastline when the wind originates from the marine side [6]. Especially the latter findings suggest that the oceans are not entirely a sink for microplastics and that at least one, possibly several water-air transfer mechanisms must be present. However there is no detailed understanding yet about what these processes are and how the water-air transfer of microplastics works exactly.

To figure out what is going on in the big picture, investigations of the different possible transport mechanisms on a microscopic scale are needed. Here, numerical modeling can greatly substitute experiments by providing much more detailed insights in these processes, allowing to see inside the flow field at all locations at once, and to slow down, freeze and even reverse time to see where exactly the ejected particles originate.

The current state-of-the-art simulation tools [28, 29, 31] allow me to investigate the ejection of marine microplastics during impacting raindrops, a mechanism that is particularly difficult to study in experiments. In the numerical model, raindrop impacts require at least between two and three orders of magnitude in spatial resolution ($\approx 4\,\mathrm{cm}$ to $\approx 86\,\mu\mathrm{m}$), which is just about feasible with a sharp interface model.

Microplastic transfer during bursting bubbles is better accessible in laboratory experiments [22–24]. Bursting bubbles can create spray droplets in two different ways: Film droplets for larger bubbles ($\approx 1\,\mathrm{cm}$ diameter), where a thin water film ($\approx 10\,\mu\mathrm{m}$) [18] tears apart, and jet droplets emerging when smaller bubbles burst. Simulation of the film tearing is still unfeasible, as the aspect ratio of film size divided by film thickness is too large to be sufficiently resolved. Simulation of the jet formation is possible, but particle count in the jet droplet cannot be predicted without an additional model for lateral particle movement on the water surface [45]. I focus more on what happens when bubbles rise in the water column: rising bubbles can pick up suspended particles on their way up – so-called bubble-scavenging – and enrich particle concentration in the sea surface microlayer (SML) compared to bulk [24, 41–43], to possibly vastly enhance the water-air transport.

Lastly, I use numerical modeling to close a gap in understanding terrestrial microplastic transport at the water-air interface in the lateral direction: when a film of water washes particles off a rough plate, peculiarly some of the particles don't move at all while others are very efficiently transported by the water flow.

2.1 Ejection of Marine Microplastics by Raindrops

2.1.1 Motivation

Raindrops impacting a water surface generate myriads of spray droplets. If the water surface is contaminated with microplastics, can they enter these spray droplets and transition into the atmosphere as an aerosol? What is the environmental relevance of this mechanism? In [Pub1] I answer these questions using computer simulations, supported by experiments. While experiments can only give limited insights into the underlying mechanisms during the raindrop impact, as it is very difficult to obtain high-speed volumetric tracing of both spray droplets and particles, my simulations provide much more detailed information, like volume, velocity and particle content of every single spray droplet. This allows secondary analysis of the trajectories of spray droplets in the air, and even estimation of the global relevance of the mechanism for spreading microplastic particles in the environment.

However, numerical simulations of impacting raindrops still present a major technical challenge, requiring immense amounts of computation to resolve close to three orders of magnitude spatially. Further, for statistical analysis of the spray droplets, a single simulation is not sufficient, and hundreds of simulated impacts are required. With existing computational fluid dynamics (CFD) software, this task is not feasible, as estimated compute time exceeds several years even with low resolution models, and with the most detailed models even exceeds a human lifetime [105]. Fortunately, the lattice Boltzmann method, extended by a state-of the-art Volume-of-Fluid model, is efficient enough when parallelized on graphics processing units (GPUs), such as in the *FluidX3D* software [28, 29, 31], to make the computational workload manageable at sufficiently large grid resolution.

2.1.2 Methods

To simulate single raindrop impact events, I use the lattice Boltzmann method (LBM) [25–27] as Navier-Stokes solver for the fluid phase. The Volume-of-Fluid (VoF) model [28, 32–37] extension handles a sharp interface to the gas phase as well as surface tension with piecewise linear interface construction [29, 106]. The immersed-boundary method (IBM) [28, 107, 108] is used to model microplastic particles, whereby one microplastic particle corresponds to one IBM particle with a hydrodynamic diameter of the simulation lattice constant (43-151 μm). These methods are implemented as part of the *FluidX3D* software [28, 29, 31]. Upon touching the ceiling of the simulation domain, individual spray droplets are identified using a Hoshen-Kopelman algorithm [46, 109], and their velocity, volume and particle content is measured, before being removed from simulation.

Since it is a new software, the simulation model is rigorously validated in the Supporting Information (SI) of [Pub1], on various free surface systems: Plateau-Rayleigh instability [110], oblique droplet impact [111, 112], drop impact on a shallow pool [113, 114], and finally direct comparison to experimental high-speed images for a 4.1 mm diameter raindrop impact [115] (figure 2.1). Laboratory experiments are also used to validate numerical results.

As a main model system, I choose the 4 mm diameter raindrop impact at 8.8 $\frac{\mathrm{m}}{\mathrm{s}}$ terminal velocity [116–118]. I also examine different raindrop diameters between 1-7 mm and oblique impacts up to 40°. The first 10 milliseconds after the impact are simulated, where simulations show good agreement with experimental high-speed images [115], to capture especially the initial small and fast spray droplets relevant to atmospheric pickup.

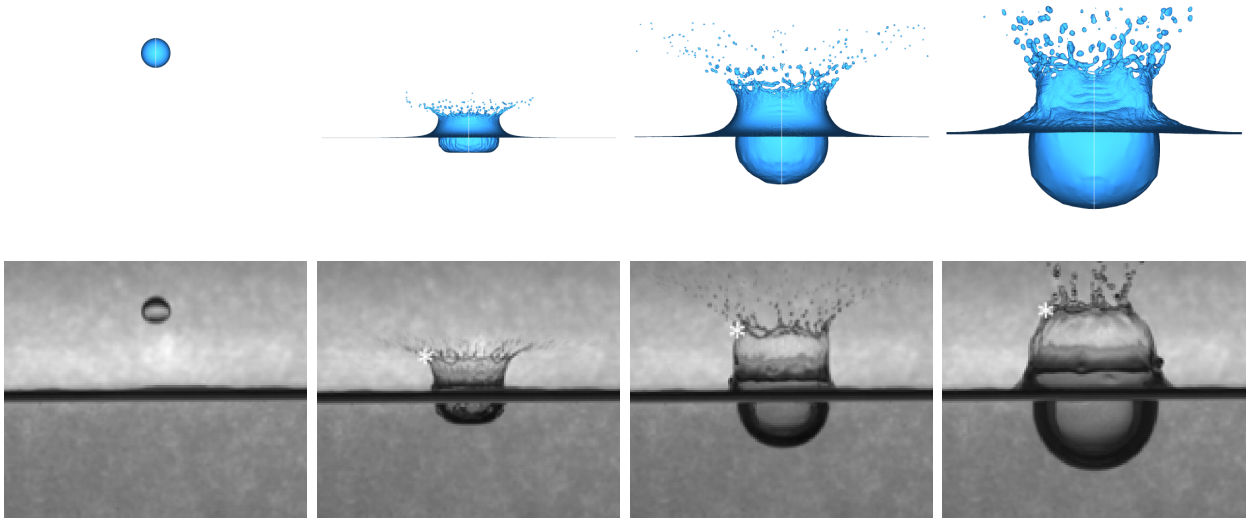


Figure 2.1: The VoF-LBM simulation model compared to the experiment from Murphy et al. [115], figure 3 (a)-(d), at times $t \in \{-2, 1, 3, 8\}$ ms.

To get sufficient statistics, for each parameter set, the impact simulation is repeated 100 times. The simulations generate a list of all generated spray droplets with their properties (volume, velocity, number of contained particles, and time of detection). This data is used for secondary analysis to numerically simulate droplet trajectories in air – considering drag [38] and evaporation [39] – under different wind conditions [40], to determine maximum altitude for comparison with experiments and airborne time for determining the fate of the particles.

2.1.3 Key Results

I observe that microplastic particles move inside of the spray droplets upon raindrop impact (figure 2.2). The particle concentration in spray droplets is approximately 90% of the concentration in the reservoir, which means that the water of the raindrop itself, which initially is devoid of particles, ends up in the reservoir, and predominantly the contaminated reservoir water is ejected as spray. Because it is a simulation, I can trace back time after detecting the particles in the spray droplets, to see where exactly they originate in the reservoir: ejected particles initially are located in a flat, ring-shaped volume in the sea surface microlayer around the impact site. No particles from a depth larger than the radius of the raindrop are ejected (figure 2.2).

Repeating the simulation for the 4 mm diameter raindrop 100 times and measuring the spray droplets each time, I obtain a clear picture of their properties (figure 2.3). The first droplets to separate from the crown rim are small and fast, but contain only few microplastic particles. The concentration of microplastic particles in spray droplets does not significantly differ depending on droplet diameter, so the number of contained particles scales with droplet volume, meaning larger droplets carry the vast majority of particles. These larger droplets separate later and at lower velocity.

Simulating different raindrop diameters, from 1-7 mm in diameter, each simulated 100 times, I take a look at the spray droplet distribution and where the particles are (figure 2.4). Small 1 mm diameter raindrops do not produce any crown spray droplets upon impact. The

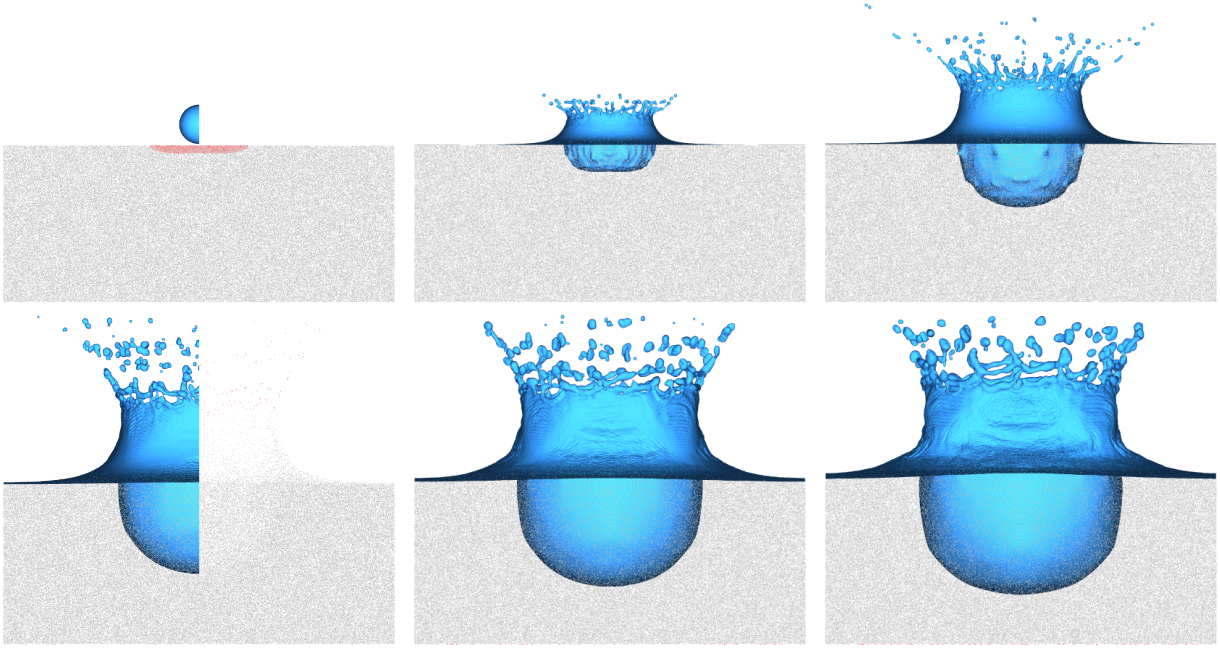


Figure 2.2: The 4 mm diameter raindrop impact simulation with microplastic particles. Time stamps (left to right) are $t \in \{0.0, 1.0, 2.5, 5.0, 7.5, 10.0\}$ ms. In the figures for $t \in \{0.0, 5.0\}$ ms, the surface is cut open to visualize particles. Particles that are later captured in the spray droplets are drawn in red.

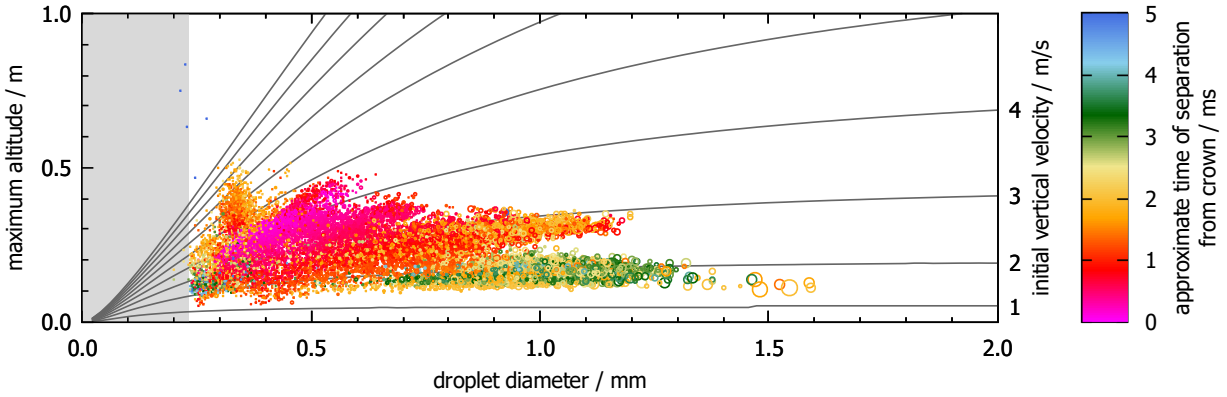


Figure 2.3: Diameter and maximum altitude of droplets detected within 10 ms after impact of 100 4 mm diameter raindrops. The size of the circles indicates the number of microplastic particles contained in each droplet. The lines indicate constant initial vertical velocity. Tiniest droplets below 0.23 mm diameter (gray area) cannot be resolved in the simulation. The color represents the approximate time the droplet separated from the crown rim. The maximum altitude of ejected droplets decreases over time.

size distribution of spray droplets for the smaller 2-5 mm raindrops has a double peak for 2nd (smaller) and 3rd (larger) generation droplets [119] that transitions into a continuous distribution for larger 6-7 mm raindrops. The microplastic particles predominantly enter the larger spray droplets as they carry disproportionately more fluid volume. Larger raindrops

generate a larger number of spray droplets and also eject more particles. Tiniest droplets cannot be resolved in the simulations, but since these do not carry significant amounts of particles, they are not relevant for this study.

I also simulate oblique impacts, as the ocean surface is expected to be wavy in windy conditions. For larger impact inclination, the particle concentration in spray droplets slightly decreases, as a larger fraction of the clean raindrop water is redirected into the then asymmetric crown.

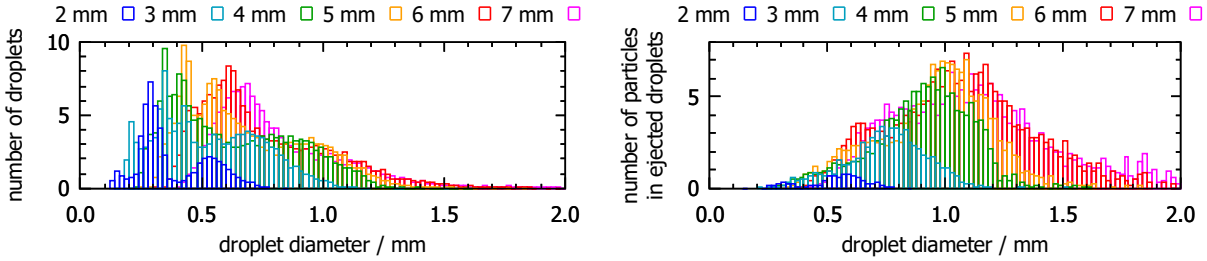


Figure 2.4: Size distribution of ejected droplets (left) and distribution of particles in ejected droplets (right) depending on droplet size for various raindrop diameters.

Knowing the properties of spray droplets, I investigate their fate in the atmosphere. For each of the measured spray droplets in the long list with their volume, velocity and particle count, the 3D trajectory through air is computed for several upwind velocities numerically, with Runge-Kutta-4 integration of the sophisticated drag model by Feng [38] together with the evaporation model by Holterman [39]. As a result, I get the maximum altitude of the trajectory as well as the airborne time.

The values for the maximum altitude are used to compute how many spray droplets will reach a glass plate mounted at a certain altitude over the reservoir surface to capture droplets. This is compared with experiments. While the number of droplets agrees very well with experiments (1.3x larger in experiments), the number of particles is found to be 2.5x larger in the experiment. This might be due to the pristine polystyrol particles in the experiment tending to stick to the water surface, so that a larger concentration is present in spray droplets.

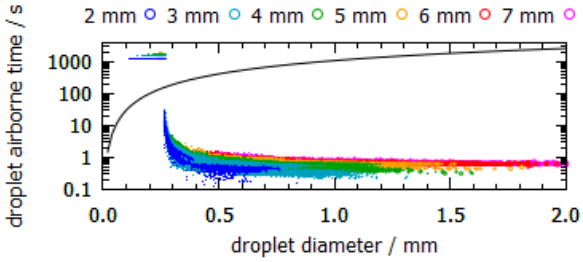


Figure 2.5: With 1 $\frac{\mathrm{m}}{\mathrm{s}}$ vertical updraft velocity, all droplets smaller than 0.26 mm diameter have diverging airborne time (capped at finite values so that the data points are visible in the diagram). The black curve represents the lifetime (time until full evaporation) of droplets depending on diameter. If the airborne time is larger than the lifetime, the droplets are considered picked up by the atmosphere.

The airborne time for all droplets is less than 1 second without wind. Only when updraft is present, the smallest droplets have diverging airborne time and can fully evaporate (figure 2.5). From radar measurements it is known that the size distribution of raindrops follows an exponential distribution, with small raindrops being much more frequent than large ones, the Marshall-Palmer law [120, 121]. This raindrop size distribution depends on the rain rate: for more intense rain, the fraction of larger raindrops increases. Based on these models, and assuming an average microplastic concentration in the SML of 2.9 particles per liter [9, 101], I first estimate the maximum transition rate of microplastic particles during local rain events as a function of rain rate (figure 2.6). The smallest 1 mm diameter raindrops do not contribute as they don't produce spray droplets. The 2 mm diameter raindrops generate only few spray droplets, but they outnumber all other sizes by far, so have the largest contribution to water-air transfer overall. Larger raindrops make up only an insignificant fraction, even at large rain rate.

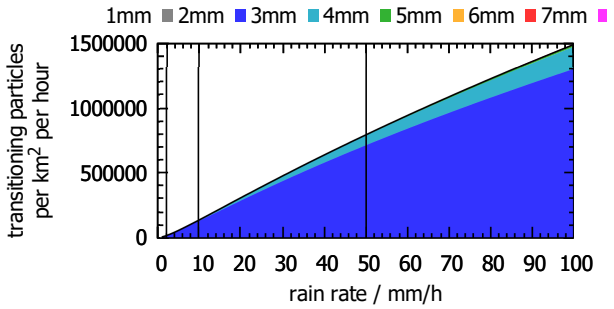


Figure 2.6: Contribution of different raindrop diameters to estimated number of microplastic particles transitioning from the oceans into the atmosphere per km^2 per hour depending on rain rate for 1 $\frac{m}{s}$ vertical updraft velocity.

The rain rate itself over time follows the Rice-Holmberg model for the rain rate distribution [122], stating that light rain events are exponentially more frequent than heavy rain events. Based on the known annual global amount of precipitation [123], our simulations for various different raindrop diameters, an estimated average concentration of 2.9 particles per liter in the SML, and an upper limit estimate for typical updraft velocity of 0.75 $\frac{m}{s}$ [40], I am able to give at least a very coarse estimate of 10^{14} for the upper bound of the number of microplastic particles transitioning from global oceans into the atmosphere annually during impacting raindrops. This estimate contains a number of vast simplifications and big uncertainties, such as the particle concentration in the SML. For the latter, satellite radar measurements [124] and simulations of particle advection in ocean currents [11, 125, 126] could provide better locally resolved estimates in the future.

I conclude that raindrop impacts as transport mechanism for microplastics from water to air is possible, especially in storm events. This underlines the large mobility of microplastic particles in the environment. However our upper limit estimate for the global annual number indicates that the environmental relevance of this particular mechanism is low compared to other mobility pathways [5, 24].

2.2 Vertical Transport of Microplastics by Rising Bubbles

2.2.1 Motivation

With wave motion on the ocean surface, air bubbles are constantly created, rising back to the surface and bursting. Depending on bubble diameter, the bursting process can release a spray of droplets either from the tearing of a thin film, or from a fast jet upon collapse of the cavity [127–135]. Microplastic transport during the bubble bursting process is already well covered by experiments [22–24], as well as by our own work in [Pub7]. Based on our experimental results, and the findings of [Pub1] that only the smallest droplets can fully evaporate, I estimate the microplastic transfer rate by bursting bubbles to be 10-1000 times larger than by impacting raindrops, which is still in line with other estimates [24]. While modeling the bursting of small bubbles with VoF-LBM produces correct results for the jet droplet diameter and velocity (figure 2.7), the model cannot handle the additional physics of particles moving laterally on the water surface when the capillary wave travels down the cavity, so numerical results on the particle count do not match experimental findings of [23, 24, 136] and [Pub7], where the particle concentration in jet droplets is observed significantly enriched.

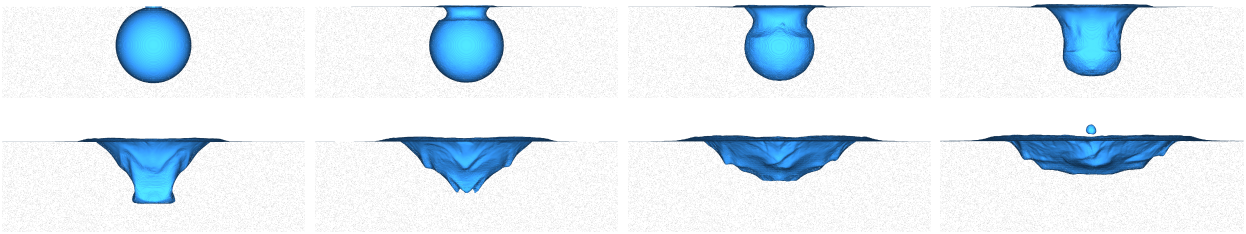


Figure 2.7: *FluidX3D* VoF-LBM simulation of a bursting 4 mm diameter bubble in water, shown at times $t \in \{0, 1, 2, \dots 7\}$ ms. Starting from the rim, a capillary wave accelerates down the bubble cavity and coalesces into an upward jet. A single droplet with a diameter of 535 μm and velocity of 1.24 $\frac{\text{m}}{\text{s}}$ separates from the top of the jet, which is in line with experimental measurements [127, 130, 137].

In [Pub2] I thus focus on the second effect that bubbles have: they can possibly enrich the microplastic concentration in the sea surface microlayer through bubble scavenging, enhancing water-air transport not only for subsequent bubble bursts [45, 136] but also other water-air transport mechanisms, even if the density of the particles is larger than the density of water [44].

2.2.2 Methods

Rising bubbles are simulated with the Volume-of-Fluid free surface lattice Boltzmann method (VoF-LBM/FSLBM), extended to track individual bubbles, and merger and splitting events for multiple bubbles [46]. The FSLBM model is validated and compared with the alternative Allen-Cahn phase-field LBM approach in [Pub6]; both models show good agreement in

simulating rising bubbles in various parameter regimes, yet the FSLBM simulations require significantly less memory and are thus preferred.

As a model system, I simulate the rising of a 4 mm diameter air bubble in water, for a distance of 12 cm. The simulation box has periodic boundary conditions in lateral directions. Microplastic particles are modeled with the immersed-boundary method (IBM) [28, 107, 108] extension, where each microplastic particle is modeled as one IBM particle with a hydrodynamic diameter of the simulation lattice constant (54 μm). To eliminate another possible complication, particles are modeled as neutrally buoyant. Particles in the model also do not interact with each other, so a very large concentration of 100000 particles per cm^3 is simulated and only two simulations are performed for particles sticking and not sticking to the water interface, modeled with short-ranged attracting/repelling hard potentials on the bubble surface. Particle positions are compared at the beginning and end of the simulation, to obtain the vertical travel distance of each particle.

2.2.3 Key Results

In accordance with experimental observations [138], the simulated bubble shows roughly spherical shape and wobbling motion. The simulated rising velocity at 136 $\frac{\text{mm}}{\text{s}}$ is lower than experimental measurements at 200 $\frac{\text{mm}}{\text{s}}$ [139]. Figure 2.8 shows the bubble at various points in time, dragging an upward plume of particles behind it, a clear sign of deviations from Stokes flow as expected at $Re = 545$. Only a plane of particles, colored by initial vertical position, is shown, and the bubble rises in a spiraling motion, going behind and then back in front of the visualized plane.

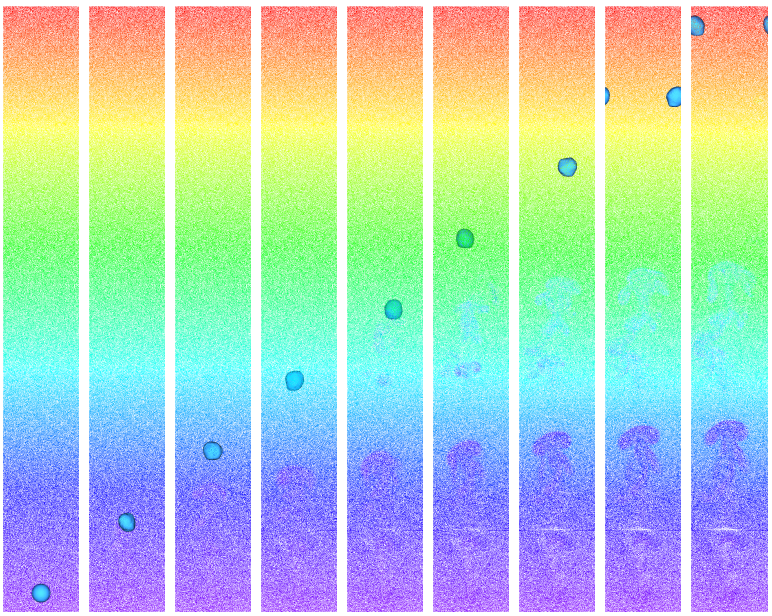


Figure 2.8: Simulation of the rising bubble. Microplastic particles are colored by their initial vertical position.

For all particles, I track the vertical travel distance relative to the vertical travel distance of the bubble. The data is visualized in histograms in figure 2.9. Depending on if particles

do or do not stick to the bubble surface, the histograms look vastly different, yet both are asymmetrical due to non-laminar flow. With particles not sticking to the bubble, some particles move considerable vertical distance as part of the upward plume. Particles sticking to the bubble show the same behavior, but additionally there is a plateau in the histogram, indicating a column of fluid over the bubble in which the bubble picks up any particle and transports it all the way to the top. Based on the plateau level, I calculate the cross-section of this vertical column of fluid as 23% of the cross-section area of the bubble at the finest simulated grid resolution (lattice constant of 54 μm). For coarser grid resolution, the modeled hard potential where the bubble captures particles (1 lattice constant wide) increases relative to the bubble diameter, slightly enlarging the resulting cross-section.

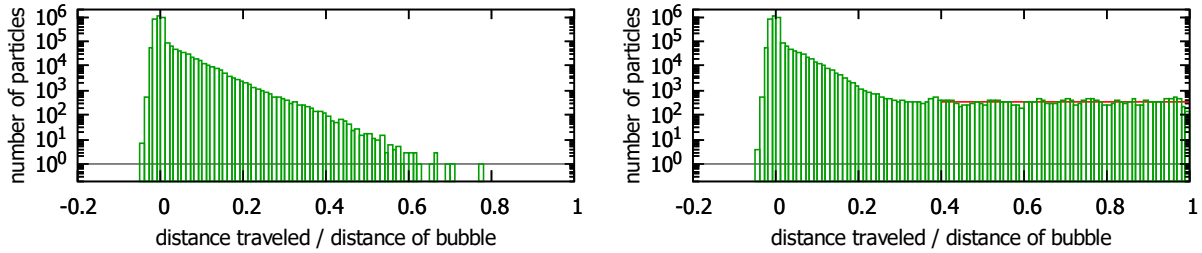


Figure 2.9: If particles do not stick to the water interface (left), the distribution of vertical travel distance of particles is already asymmetric due to deviations from laminar flow in the proximity of the bubble. While most particles do not move (notice the logarithmic scaling), some particles are entrained and travel upwards considerable distances. If particles stick to the water interface (right), the bubble picks up all particles in a vertical column above and transports them to the top, visible as plateau in the distribution.

The weighted sum over all histogram bins provides the net movement of particles once the ascending volume of the bubble, which itself is devoid of particles, is subtracted. When the particles do not stick to the bubble, the average particle travels upward 0.02% of the distance the bubble traverses. When particles stick to the bubble, this significantly increases to 0.39%. Pristine particles are more hydrophobic than weathered particles [44, 140–142] and are expected to stick more to bubbles, significantly enhancing vertical transport during bubble scavenging.

2.3 Horizontal Transport of Microplastics on Rough Surfaces

2.3.1 Motivation

Microplastics do not only contaminate marine systems but also terrestrial systems, where they can be transported laterally at the water-air interface by a thin water film on a solid surface. Contamination of agricultural land seems particularly troublesome. Still it's not clear how exactly microplastic particles can move on land surfaces. In [Pub5] we study

the transport of microplastics by a thin water film on a rough plate, a model resembling microplastics being washed across surfaces like asphalt by storm water runoff.

Tracking PMMA particles (around 1-2 mm in diameter) with an sCMOS camera in the experiment provides a good picture of transport behavior on the larger scale, but lacks understanding of why some particles are washed down the rough plate while others stay in place. My computer simulations of the flow pattern on a small scale close this gap in understanding the transport mechanism.

2.3.2 Methods

The water flow over the inclined rough surface is modeled with the Volume-of-Fluid [28, 34, 37] lattice Boltzmann [25, 26] implementation *FluidX3D* [28, 29, 31]. A 1 cm^2 section of the plate is simulated and the rough surface is modeled with 2D periodic Perlin noise [143], with an average grain size of 0.4 mm and average water film thickness of 0.3 mm, to match the experimental flow rate of 7.2 $\frac{\mathrm{L}}{\mathrm{h}}$.

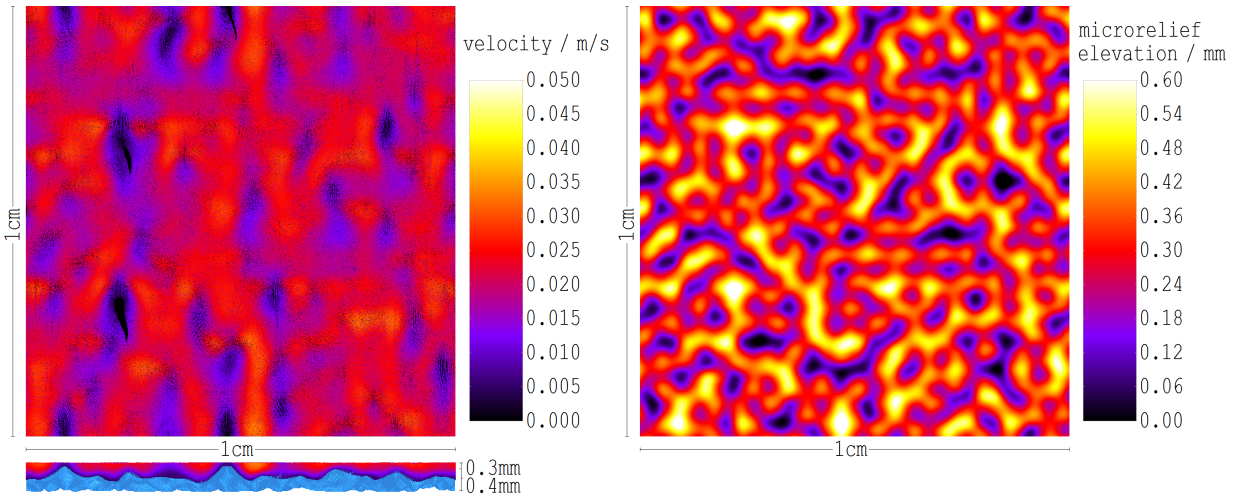


Figure 2.10: Velocity profile on a 1 cm × 1 cm section of the microrelief illustrated with colored passive tracer particles (left). The flow direction is top to bottom. The stationary flow pattern forms channels of increased velocity around teardrop shaped regions of decreased velocity behind bumps in the microrelief. On the bottom left, a side view of the microrelief and water layer above is shown, and on the right the microrelief elevation is illustrated.

2.3.3 Key Results

Experimental observations show that some of the particles are quickly washed down the inclined rough plate, some move only small distances, and some stay in place. The simulated flow profile over the microrelief (figure 2.10) reveals that a stationary pattern of preferential flow channels is formed and in some spots flow is inhibited. This explains particle movement: particles located in the flow channels are easily washed down the plate, while particles outside of these channels lock into the microrelief and the low hydrodynamic forces in these spots cannot overcome friction forces.

Chapter 3

Synopsis Part B: Enhancing the Lattice Boltzmann Method for Future Studies

The application studies in the first part of this thesis have proven the lattice Boltzmann method (LBM) on graphics processing units (GPUs) [28, 29, 31, 46–100] to be a very capable tool for simulating fluid flow, yet also emphasized the limitations on grid resolution imposed by its large memory demand. Although their much higher memory bandwidth [28, 48–55, 58–66, 74–87, 100, 144–149] and efficiency compared to CPUs offers significantly faster performance, the limited memory capacity of GPUs still poses a major constraint [99] for grid resolution in the LBM, despite the exponential trend in hardware development over the past decades (Moore's law). Larger grid resolution means that smaller details in the flow field can be resolved while at the same time a much larger system is captured in its entirety. The memory demand has always been one of the biggest disadvantages of the LBM compared to other fluid solvers such as the finite volume method, although compute performance is far superior. To fix the problem of memory demand, I develop two novel approaches:

1. A new set of in-place streaming schemes in [Pub4] that are much simpler than previous solutions and offer ideal performance characteristics.
2. A framework for 16-bit memory compression in [Pub3] by decoupling arithmetic precision and memory precision.

Together, these two methods vastly reduce memory demand of the LBM, to 55 Bytes/node (D3Q19) – 1/6 compared to two-grid FP64 implementations at 344 Bytes/node – while not significantly impacting accuracy. Today's datacenter GPUs offer up to 64 GB (AMD MI200), 80 GB (Nvidia A100/H100) and 128 GB (Intel Ponte Vecchio) memory capacity – *FluidX3D* runs on all of them via OpenCL [150]. At only 55 Bytes/node, their memory capacity now allows for maximum single-GPU resolutions of 1068^3, 1148^3, and 1344^3 respectively.

However with a very efficient LBM GPU implementation, overall simulation performance can still be poor, for a maybe unexpected reason: file export. Writing large volumetric data files to the hard disk for later analysis or rendering can quickly become the major part of overall compute time. In [Pub8] I explore an alternative: rendering the data directly while it already is in fast GPU memory, and avoiding file export altogether.

These upgrades are implemented as part of the re-written *FluidX3D* software, and the source code is published on GitHub [150].

3.1 Esoteric Pull and Esoteric Push In-Place Streaming

3.1.1 Motivation

The parallel implementation of the LBM on GPUs means that the density distribution functions (DDFs) on the computational grid are accessed and modified for each grid point simultaneously. Access conflicts must not occur. Commonly this is solved simply by having two copies of the computational grid in memory, and alternatingly only reading from one and writing to the other [28, 29, 31, 46, 57–85].

Two copies of the grid mean double the memory demand. There has been intricate solutions to avoid duplicate grids and still enable parallel conflict-free data access, namely AA-Pattern [48], Shift-and-Swap-Streaming [49] and Esoteric-Twist [47], but these have not found too much adoption in codes [51–53, 56] due to their complexity.

The Esoteric-Twist scheme ingeniously introduces implicit bounce-back, meaning bounce-back boundaries emerge from the streaming directly without needing explicit implementation, leading to a reduction in bandwidth as neighbor flags do not have to be checked during streaming. This side-effect almost entirely compensates the bandwidth penalty resulting from more misaligned writes of the DDFs.

In [Pub4] I introduce two new streaming schemes termed Esoteric-Pull and Esoteric-Push, building upon the idea of Esoteric-Twist, but changing the access pattern such that index calculation is trivial with the common sorting of streaming directions in the velocity set, where opposing directions are accessible with "+1"/"-1" in the direction index, as introduced in [28]. This way, streaming directions always correspond to the directions of neighboring grid points, which is not the case for Esoteric-Twist, and the implementation is vastly simplified.

Underlining the demand for simpler in-place streaming algorithms, independently and in parallel to [Pub4], another different in-place scheme termed one-step index (OSI) has been found in [151] that utilizes mainly misaligned memory access.

3.1.2 Methods

Following the idea of Esoteric-Twist, my new Esoteric-Pull and Esoteric-Push in-place streaming schemes do the first part of streaming at the end of one stream-collide kernel and the second part at the beginning of the next to avoid data access conflicts on parallel hardware. The access pattern however is chosen differently such that direct addressing is now possible, with the directional index always corresponding to the direction of the streaming neighbor (see figure 3.1). This makes the implementation very straight-forward (see listings 3.1 and 3.2).

Of great relevance is also how well in-place streaming is compatible with LBM extensions, particularly free surface LBM (FSLBM). Therefore the existing FSLBM implementation [28] is modified, and on top the flag bits are optimized such that now only 3 bits are required per grid point for FSLBM instead of the previous 5.

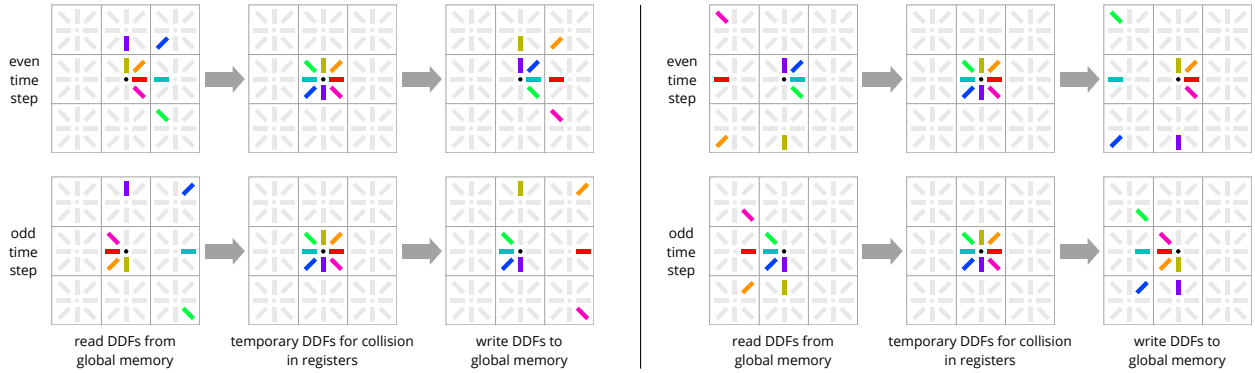


Figure 3.1: Esoteric-Pull (left) and Esoteric-Push (right) in-place streaming schemes.

```
1  void load_f(const uint n, float* fhn, const global fpxx* fi, const uint* j, const ulong t) {
2    fhn[0] = load(fi, index_f(n, 0u)); // Esoteric-Pull
3    for(uint i=1u; i<def_velocity_set; i+=2u) {
4      fhn[i   ] = load(fi, index_f(n    , t%2ul ? i    : i+1u));
5      fhn[i+1u] = load(fi, index_f(j[i], t%2ul ? i+1u : i   ));
6    }
7  }
8  void store_f(const uint n, const float* fhn, global fpxx* fi, const uint* j, const ulong t) {
9    store(fi, index_f(n, 0u), fhn[0]); // Esoteric-Pull
10   for(uint i=1u; i<def_velocity_set; i+=2u) {
11     store(fi, index_f(j[i], t%2ul ? i+1u : i   ), fhn[i   ]);
12     store(fi, index_f(n    , t%2ul ? i    : i+1u), fhn[i+1u]);
13   }
14 }
```

Listing 3.1: Esoteric-Pull implementation in OpenCL C.

```
1  void load_f(const uint n, float* fhn, const global fpxx* fi, const uint* j, const ulong t) {
2    fhn[0] = load(fi, index_f(n, 0u)); // Esoteric-Push
3    for(uint i=1u; i<def_velocity_set; i+=2u) {
4      fhn[i   ] = load(fi, index_f(j[i+1u], t%2ul ? i+1u : i   ));
5      fhn[i+1u] = load(fi, index_f(n      , t%2ul ? i    : i+1u));
6    }
7  }
8  void store_f(const uint n, const float* fhn, global fpxx* fi, const uint* j, const ulong t) {
9    store(fi, index_f(n, 0u), fhn[0]); // Esoteric-Push
10   for(uint i=1u; i<def_velocity_set; i+=2u) {
11     store(fi, index_f(n      , t%2ul ? i    : i+1u), fhn[i   ]);
12     store(fi, index_f(j[i+1u], t%2ul ? i+1u : i   ), fhn[i+1u]);
13   }
14 }
```

Listing 3.2: Esoteric-Push implementation in OpenCL C.

3.1.3 Key Results

Like other in-place streaming schemes, Esoteric-Pull/Push cut memory demand of the LBM almost in half. In the same way as with the Esoteric-Twist scheme, bandwidth is reduced with implicit bounce-back. However misaligned writes are minimized, slightly increasing efficiency over Esoteric-Twist. Overall, performance is mostly unchanged compared to the One-Step-Pull two-grid implementation, and on CPUs performance even is significantly increased (see figure 3.2).

For the FSLBM, in-place streaming requires a 4th kernel to compute the mass transfer across the grid, that now cannot be integrated into the stream-collide kernel anymore. This requires more memory access, reducing FSLBM performance by about 20%. The savings in memory demand, and thus larger possible grid resolution, however are much more important here than the moderate performance reduction.

Overall, the Esoteric-Pull/Push are the simplest possible solution to in-place streaming and at the same time outperform previous solutions. Implementation is even simpler than two-grid schemes as neighboring flags do not have to be checked during streaming anymore. These advantages should finally enable widespread adoption of in-place streaming in LBM implementations.

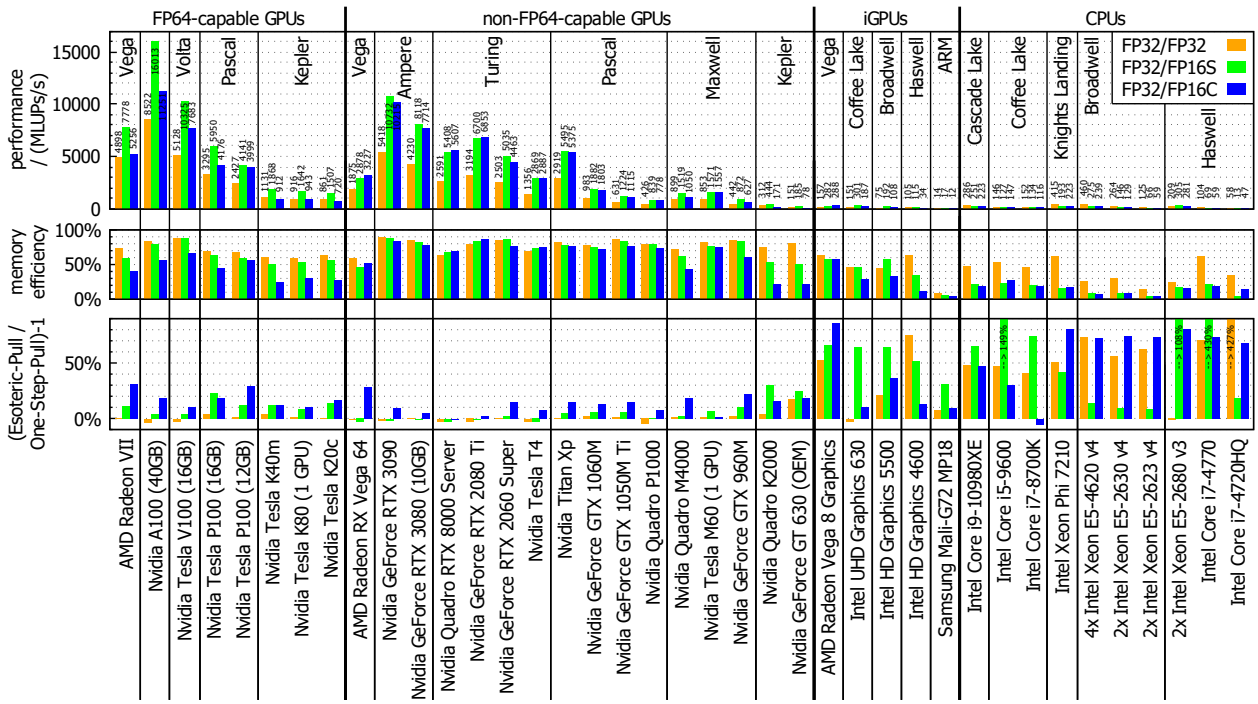


Figure 3.2: Performance of Esoteric-Pull with D3Q19 SRT on different hardware in the *FluidX3D* OpenCL implementation, in million lattice updates per second (MLUPs/s). Performance comparison with the One-Step-Pull streaming scheme [58] shows only insignificant differences on most dedicated GPUs, but a large speedup on integrated GPUs and CPUs.

3.2 Accuracy and Performance of the Lattice Boltzmann Method with 64-bit, 32-bit, and Customized 16-bit Number Formats

3.2.1 Motivation

Due to their excellent performance with vectorizable algorithms, GPUs are becoming the platform of choice for LBM simulations [28–31, 46–57, 59–100]. Their limited memory capacity however is in conflict with the large memory demand of the LBM. A more memory-efficient in-place streaming implementation alone is not enough to fit the largest computational grids in the limited memory of a GPU, and additional ways to reduce memory demand are desired. The density distribution functions (DDFs) of the LBM make up most of its memory demand, but are not direct observables, in contrast to velocity and density. If the DDFs alone were compressed from FP32 to a 16-bit format in memory, the overall memory demand of the LBM could be almost cut in half.

However so far it was entirely unclear what effect the choice of floating-point format has on the accuracy of LBM simulations, with only few studies shedding some light on comparing FP64 and FP32 [53, 63, 78, 84, 100, 152]. Since GPUs are vastly more efficient with lower precision FP32, a persistent discussion about the choice of precision has been present [48, 50, 52, 53, 59, 61, 63, 65, 66, 74–78, 84, 85, 96–98, 100, 145, 146, 153]. A full comparison study about the impact of floating-point precision on LBM accuracy is needed. I provide this comparison in [Pub3]. With feasibility of lower precision assessed, I then propose to reduce memory precision even further to 16-bit.

3.2.2 Methods

I first decouple arithmetic precision and memory precision [154, 155] and introduce the notation FPxx/FPyy, where FPxx is the arithmetic precision format and FPyy is the memory precision format. Regarding arithmetic precision formats, the vast majority of GPUs support only FP32 at full performance and FP64 with much reduced performance, but FP16 is not widely supported. For memory precision on the other hand, any customized or even exotic number format can be used on any hardware, as long as it is 64-bit, 32-bit or 16-bit in size, and as long as conversion algorithms from/to the arithmetic precision format are provided.

To make best use of the available bits for the memory precision format, I first identify the range of numbers that the LBM is working with: the DDFs are clustered around the lattice weights of the velocity set in use. By subtracting the lattice weights in the equilibrium DDF equation

$$f_i^{\text{eq-shifted}}(\rho,\vec{u}) := f_i^{\text{eq}}(\rho,\vec{u}) - w_i = \tag{3.1}$$

$$= w_i\,\rho\cdot\left(\frac{(\vec{u}\circ\vec{c}_i)^2}{2\,c^4} - \frac{\vec{u}\circ\vec{u}}{2\,c^2} + \frac{\vec{u}\circ\vec{c}_i}{c^2}\right) + w_i\,(\rho-1) \tag{3.2}$$

and adding them again in the density summation

$$\rho = \sum_i \left(f_i^{\text{shifted}} + w_i\right) = \left(\sum_i f_i^{\text{shifted}}\right) + 1 \tag{3.3}$$

I can work with these shifted DDFs throughout LBM [65, 100, 149, 152, 156], such that the numeric values are now all centered around 0, where floating-point accuracy is best. Moreover, I identify a number range of ± 2 to be sufficient.

Knowing the number range, I tailor customized 16-bit number formats to this specific application (see figure 3.3). IEEE-754 [157–160] FP16 has large unused range, but by pre-multiplication of 2^{15} its range is shifted so that smaller numbers can be represented instead (FP16S). For halved truncation error at the cost of decreased range towards small numbers, I designed a customized FP16C format, moving 1 bit from the exponent into the mantissa; however now manual conversion in software is required, for which I provide ultrafast, branchless algorithms. A promising alternative to floating-point is the posit format [161], of which I investigate three variants ($P16_0S$, $P16_1S$, $P16_2C$), all of which require custom conversion algorithms that I also provide. 16-bit fixed-point is insufficient for the LBM.

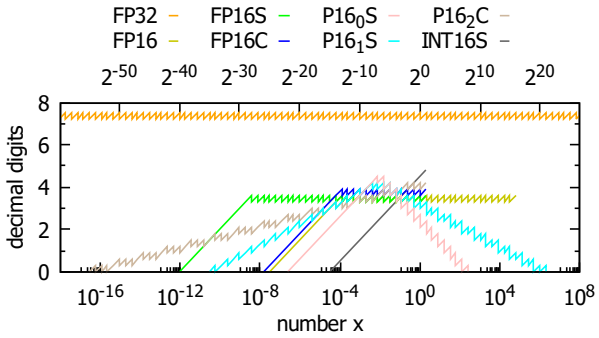


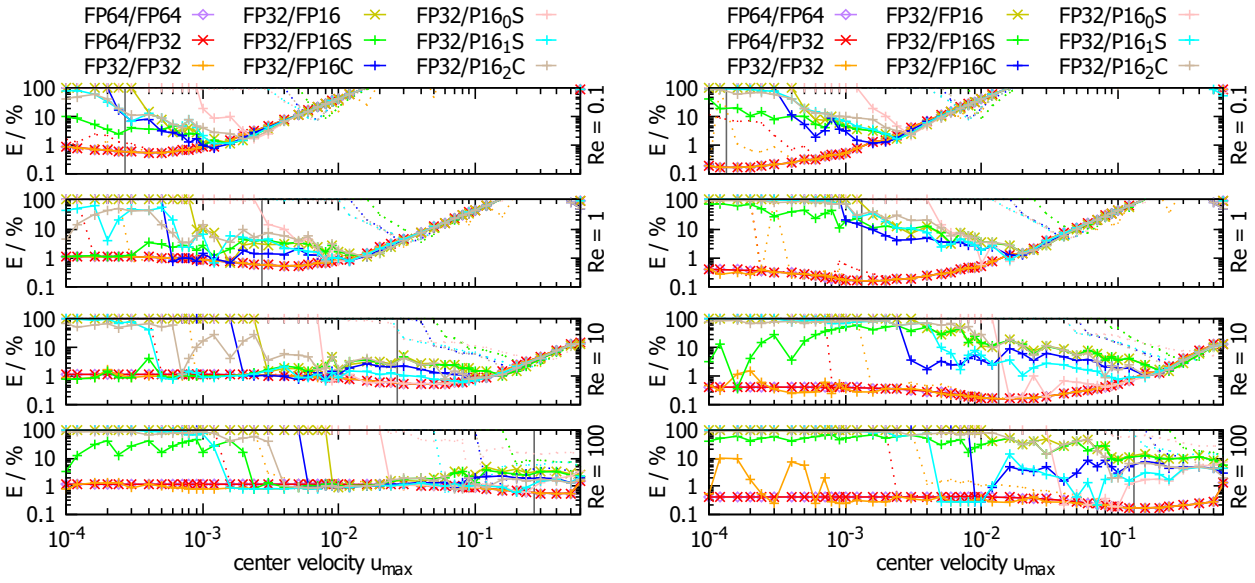
Figure 3.3: Accuracy characteristics of proposed number formats for memory precision. Floating-point reselbles a roofline, posit a pyramid and fixed-point a slope.

Six setups are studied to examine LBM accuracy:

- Poiseuille flow through a cylinder [162]
- Taylor-Green vortex energy dissipation [152, 163]
- Karman vortices [164] from flow around a cylinder
- lid-driven cavity [60, 61, 64, 72, 79, 84, 165–170]
- deformation of a microcapsule in shear flow [171–173]
- microplastic particle transport during a raindrop impact [68]

3.2.3 Key Results

With the Posieuille flow setup I probe a large range of LBM parameters in kinematic viscosity, center velocity and channel radius (figure 3.4). In all but corner cases, FP32 for both arithmetic and memory precision is as accurate as FP64. 16-bit formats for memory precision are equally accurate for large velocities in LBM units, but can fall short at very low velocity and very low volume force. The customized FP16C format shows overall best results, only outperformed by $P16_1S$ posit in some cases. DDF-shifting increases accuracy and is essential for all 16-bit formats.



(a) $R = 31$ (b) $R = 63$

Figure 3.4: Error of Poiseuille flow through a cylinder (D3Q19 SRT LBM) for varying center velocity $u_{\max}$ at constant Reynolds number $Re \in \{0.1, 1, 10, 100\}$ and constant channel radius (a) $R = 31$ and (b) $R = 63$. The dashed curves represent corresponding simulations without DDF-shifting. The vertical lines indicate the LBM relaxation time $\tau = 1$.

The Taylor-Green vortex setup [152, 163] shows exactly where each precision level breaks down during when energy dissipates in the system and flow velocity decreases (figure 3.5). Initially the simulated kinetic energy for all precision levels follows the analytic exponential decay. Once velocity becomes too small however, the energy cannot decrease further and plateaus. The plateau is located approximately at the floating-point round-off error ϵ squared. Again, DDF-shifting is essential for accurate results.

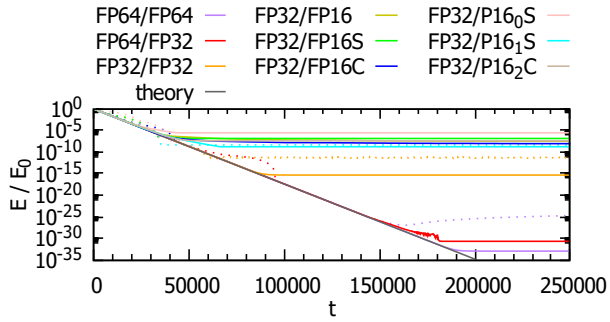


Figure 3.5: Relative kinetic energy $E(t)/E_0$ of a D2Q9 SRT simulation of Taylor-Green vortices compared to the analytic exponential decay. Dashed lines represent corresponding simulations without DDF-shifting.

The Karman vortex street setup [164] for flow around a cylinder in 2D provides insights in the behavior at large and small vorticity magnitude for the different precision levels. With DDF-shifting implemented, all precision levels produce physically accurate results with unnoticeable phase-shift even after 50 vortex cycles (figure 3.6). In domains of very low vorticity magnitude, the 16-bit formats show numerical noise. This is expected, as vorticity is computed as a finite difference of velocity, and such fine granularity in velocity cannot be

resolved anymore. The customized FP16C format with more accurate mantissa than FP16S mitigates the noise. Without DDF-shifting, there is significant phase-lag and much more noise for the 16-bit formats.

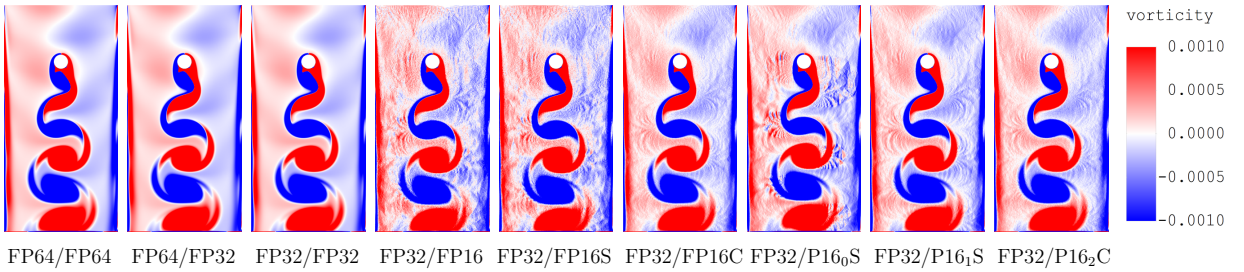


Figure 3.6: Vorticity in the Karman vortex setup with DDF-shifting.

The lid-driven cavity [60, 61, 64, 72, 79, 84, 165–170] shows no significant discrepancies between all precision levels and is in very good agreement with results from Delbosc et al. [61].

The microcapsule in shear flow setup [171–173] appears sensitive to memory precision (figure 3.7). Without DDF-shifting, the 16-bit simulations all show nonphysical behavior. With DDF-shifting, only FP32/FP16C stays close to ground truth. FP64 and FP32 appear identical.

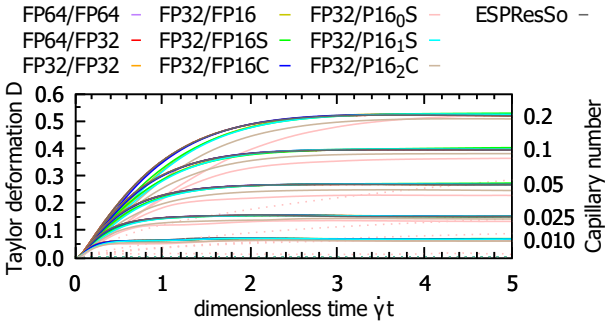


Figure 3.7: Taylor deformation of the capsule in shear flow over time. Deformation first increases and then plateaus at different levels depending on the Capillary number. Dashed lines represent corresponding simulations without DDF-shifting.

The raindrop setup [68] shows no significant differences between FP32 and 16-bit memory precision, neither qualitatively (figure 3.8) nor quantitatively with the data on microplastic particles in spray droplets.

Although the main purpose of reduced memory precision is to reduce memory demand and enable larger grid resolution, as an additional benefit, bandwidth during the LBM streaming step is also almost cut in half when going from FP32 to 16-bit, so performance is almost doubled. Looking at hardware benchmarks (figure 3.9), I find that FP64 arithmetic only performs efficiently on the very few GPUs with high FP64:FP32 compute ratio. FP32/FP32 performs very efficiently on all GPUs. FP32/FP16S is close to the expected 80% theoretical speedup. The more accurate FP32/FP16C as well as posit formats are generally a bit less efficient, because the conversion is not supported in hardware and takes 51 arithmetic operations per number.

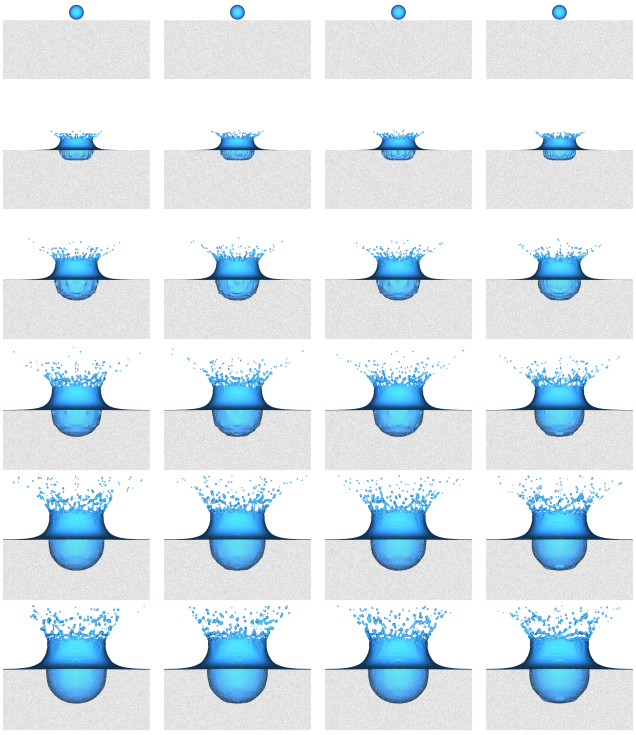


Figure 3.8: A 4 mm diameter raindrop impacting a deep pool at 8.8 $\frac{\mathrm{m}}{\mathrm{s}}$ terminal velocity, illustrated at times $t \in \{0, 1, 2, 3, 4, 5\}$ ms after impact as used in [68].

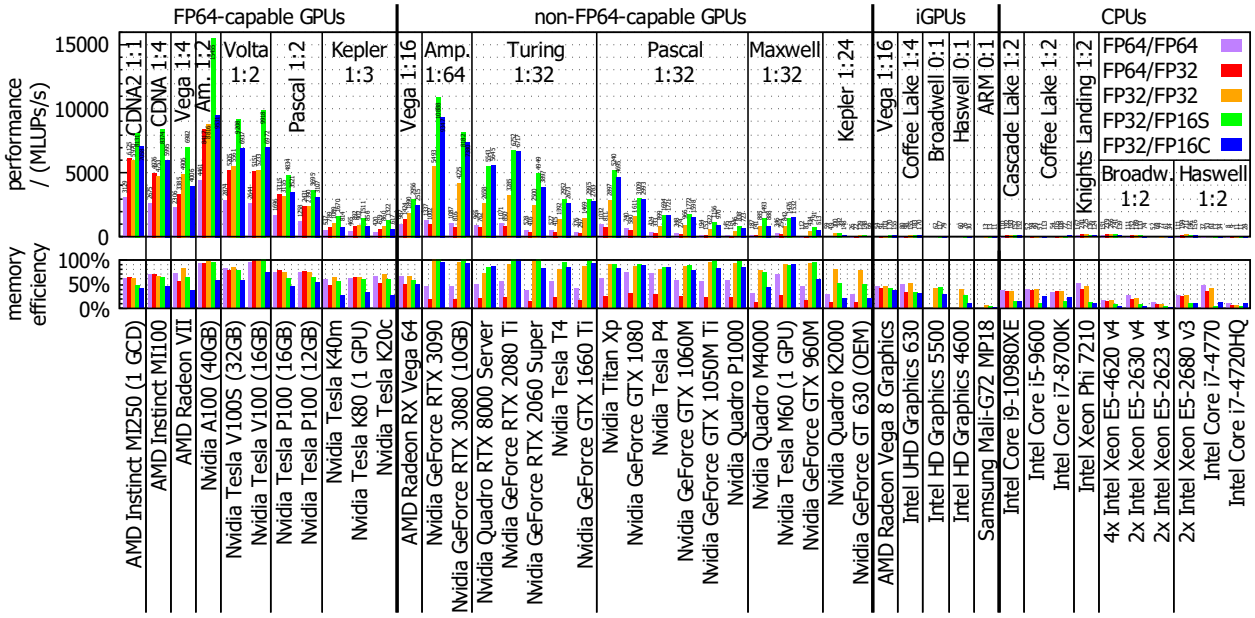


Figure 3.9: Performance of *FluidX3D* with D3Q19 SRT on different hardware, with the common One-Step-Pull streaming implementation. MLUPs/s stands for million lattice (point) updates per second.

3.3 Combined Scientific CFD Simulation and Interactive Raytracing with OpenCL

3.3.1 Motivation

With an efficient LBM GPU implementation, compute time can still be very long, for a reason outside of LBM: file export is slow. A single volumetric frame of the velocity field can be tens of GigaByte in size. With volumetric data exported regularly for time-dependent analysis or rendering, the total dataset becomes hundreds of TeraByte – unfeasible on most hardware. If possible at all, writing all this data to the hard drive is a major bottleneck that can extend compute time by orders of magnitude.

In [Pub8] I explore an approach to combine simulation and rendering, to allow avoiding time-consuming file export altogether. The idea is to briefly pause the LBM simulation, and then render the data from multiple camera positions/angles on the GPU while it already resides in ultra-fast GPU memory. Then only rendered images (a few MegaByte in size) have to be copied to CPU memory and can be viewed interactively or stored as compressed PNG/QOI images.

3.3.2 Methods

The LBM computation is done in OpenCL. To render the raw simulation data, I implemented an entire graphics engine in OpenCL [150]:

Rasterization: The traditional approach to render computer graphics is to convert vector geometry to pixels on the screen (rasterization). I use the standard Bresenham algorithm [174] to rasterize basic shapes such as lines and triangles.

Raytracing: Raytracing computes physically accurate light transport with ray optics. In computer graphics, the light rays are computed in reverse – shot out of the camera through the grid of screen pixels, here one ray per pixel. These camera rays can intersect geometry in the scene, in this case the water surface. For complicated geometry, the intersection checks are computationally too intensive, and an acceleration structure is required. The most common such acceleration structure is the bounding volume hierarchy (BVH), a tree-like data structure that groups triangles and bounding boxes into larger bounding boxes. For the water surface in LBM however, I use the LBM grid itself as the acceleration structure, in an alternative approach that is much faster than BVH: fast ray-grid traversal. Similar to Bresenham algorithm, the light ray traverses the 3D LBM grid and only traversed grid cells are checked for intersections. In each traversed grid cell, triangles are generated on-the-fly from a marching-cubes [175] isosurface, and ray-triangle intersections [176] are computed. The normal vector for refraction/reflection is trilinearly interpolated from 8 central-difference stencils, to make the surface appear smooth. A reflection ray and an internal ray are computed via Snell's law, and the internal ray is intersected with the surface again to produce a refraction ray. The reflectivity of the water surface is modeled depending on the angle of incident, and colors of the reflection and refraction ray – determined by where the rays hit the skybox – are mixed accordingly. Optionally, the ray-surface intersection is repeated recursively multiple times for both reflection and refraction ray.

Multi-GPU domain decomposition rendering: If the LBM is parallelized across multiple GPUs [31], the simulation box is split into multiple cuboid domains, with one domain per GPU (domain decomposition). Each GPU holds only the volumetric data of its own domain in memory, and data is exchanged only at the boundaries between adjacent domains. I extend the domain decomposition for rendering: Each GPU renders only its own domain to its own frame buffer, but with the camera position moved by the 3D domain offset. The rendered frames and z-buffers from all GPUs are then transferred to the CPU, and overlayed pixel-by-pixel, based on the corresponding z-buffer values. This allows rasterizing volumetric data from multiple GPUs in parallel, without having to copy any volumetric data.

3.3.3 Key Results

The combined simulation and rendering is so fast that simulations can be viewed (rasterized and/or raytraced) interactively in real time, while they are running, with as large grid resolution as fits in GPU memory. Real-time raytracing even works on GPUs that lack RTX/DXR raytracing hardware, since no BVH is required. The raytraced free surface shows physically accurate reflection and refraction (figure 3.10). The graphics engine developed here is used to create the illustrations in [Pub1], [Pub3], [Pub4], [Pub5], [Pub2] and [Pub8].

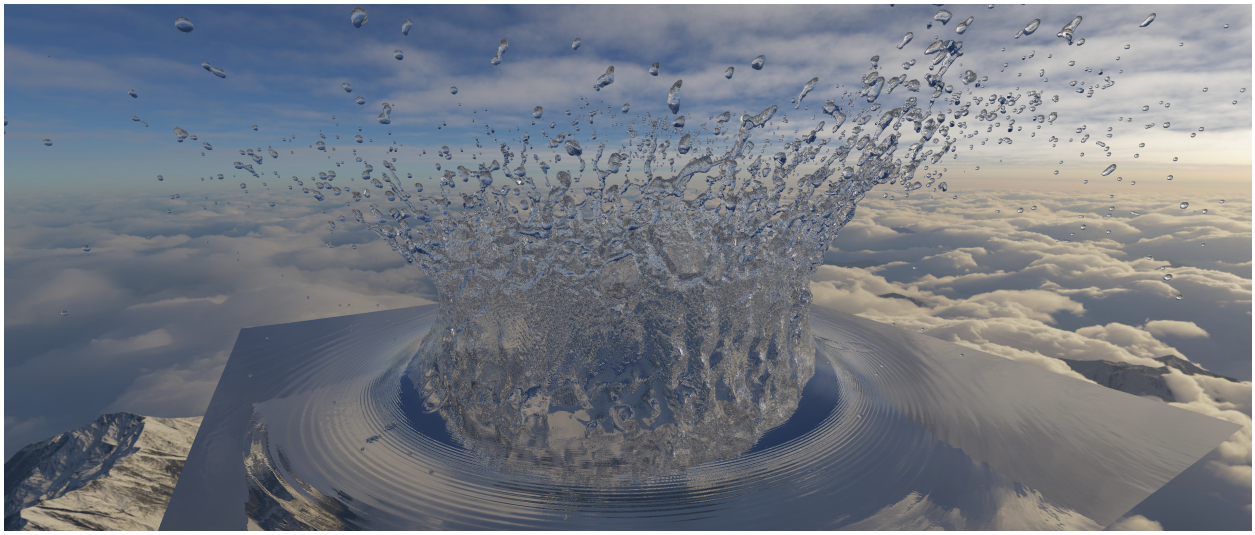


Figure 3.10: 7 mm diameter, 20° inclination, terminal velocity raindrop impact at $1060 \times 1060 \times 900$ grid resolution, 1.55 ms after impact, raytraced on a single 64 GB AMD Instinct MI200 GPU. Computing the 8970 LBM time steps takes approximately 28 minutes, and raytracing the image takes only ≈ 2 seconds at 4K resolution with one recursive repetition for both reflection and refraction ray, so up to 10 ray-grid traversal calls per pixel. This simulation is available as a video at https://youtu.be/VadLwt9OqMo.

Chapter 4

Conclusions

My work has significantly advanced understanding of microplastic transport at the water-air interface in marine and terrestrial systems. Simulations with the lattice Boltzmann method allowed for a microscopically detailed look in the mechanisms of particle transport during impacting raindrops and rising bubbles, as well as particle transport on rough surfaces by a thin water film.

By simulating more than 1600 impacting raindrops and measuring each and every ejected spray droplet, I collected enough statistical data on the droplets to see where the microplastic particles go. The microplastic particles – originating from a ring-shaped volume around the impact site at up to a few millimeters below the water surface in the sea surface microlayer (SML) – do indeed enter the spray, with almost the same concentration as in the SML. Most particles are found in larger spray droplets since they carry disproportionately more fluid volume. However these are too heavy to be picked up by wind, and only smallest droplets can fully evaporate, leaving contained particles behind as an aerosol. Larger raindrops produce more and larger spray droplets, but occur much more infrequent during rain. Small 2 mm diameter raindrops contribute the biggest amount of particles to atmospheric uptake by far. So although I found impacting raindrops to be able to transport micrioplastics from the ocean surface into the atmosphere during typical wind conditions, the estimated amount is small compared to other environmental transport processes. At the same time, my findings underline just how large the mobility of microplastics is in the environment.

As a precursor process to the transport from water to air, I also modeled bubble scavenging of micrioplastics in simulation, and found that rising bubbles indeed are able to induce a net vertical transport of particles, possibly enriching microplastic particle concentration in the sea surface microlayer. Pristine particles with more hydrophobic surface properties are especially affected by this mechanism as they tend to stick to bubbles directly.

In terrestrial systems, my simulations could close a gap in understanding how microplastics are washed down an inclined rough surface by a thin water film. The simulated flow pattern through the microrelief forms preferable channels of high flow velocity, along which particles can be transported efficiently. In regions of no such flow channels, the lower hydrodynamic forces cannot overcome the friction force of particles locking into the microrelief, so these particles stay in place.

In the process of conducting the application studies on microplastics, I discovered two key improvements for the lattice Boltzmann method on graphics processing units (GPUs), finally fixing its greatest disadvantage: memory demand. The first is a new set of in-place streaming schemes termed Esoteric-Pull/Push, building upon the idea and benefits of the Esoteric-Twist scheme, but significantly simplifying implementation and even slightly increasing performance. The second is 16-bit memory compression for the density distribution functions, cutting memory demand almost in half again, while not significantly impacting accuracy in all but corner cases compared to FP64 precision. These two improvements reduce the memory demand of the LBM to 1/6 compared to two-grid FP64 codes, making implementation on GPUs with much faster memory bandwidth, but limited memory capacity, a lot more appealing, at a very large speedup compared to CPU implementations with similar power consumption.

On top, for creating almost all visualizations in this thesis, I implemented an entire graphics engine in OpenCL, enabling rasterization and raytracing directly on the GPU with the raw simulation data already residing in fast video memory. This allows to eliminate slow export of large volumetric datasets to the hard drive, which significantly speeds up the overall compute time to the point where simulations can be viewed interactively while they are running, even at the largest grid resolutions that fit in GPU memory.

The resulting *FluidX3D* software, which I have published on GitHub, will be a powerful tool for many applications far beyond microplastics research, such as aerospace, traffic, industrial, medical and other research applications.

Chapter 5

References

[1] Steve Allen et al. "Atmospheric transport and deposition of microplastics in a remote mountain catchment". In: *Nature Geoscience* 12.5 (2019), pp. 339–344.

[2] Miri Trainic et al. "Airborne microplastic particles detected in the remote marine atmosphere". In: *Communications Earth & Environment* 1.1 (2020), pp. 1–9.

[3] V Tirelli, G Suaria, and Amy L Lusher. "Microplastics in polar samples". In: *Handbook of Microplastics in the Environment*. Springer, 2022, pp. 281–322.

[4] Imogen E Napper et al. "Reaching new heights in plastic pollution—preliminary findings of microplastics on Mount Everest". In: *One Earth* 3.5 (2020), pp. 621–630.

[5] Nikolaos Evangeliou et al. "Atmospheric transport is a major pathway of microplastics to remote regions". In: *Nature communications* 11.1 (2020), pp. 1–11.

[6] Steve Allen et al. "Examination of the ocean as a source for atmospheric microplastics". In: *PloS one* 15.5 (2020), e0232746.

[7] Chelsea M Rochman. "Microplastics research—from sink to source". In: *Science* 360.6384 (2018), pp. 28–29.

[8] Andrés Cózar et al. "Plastic debris in the open ocean". In: *Proceedings of the National Academy of Sciences* 111.28 (2014), pp. 10239–10244.

[9] Katsiaryna Pabortsava and Richard S Lampitt. "High concentrations of plastic hidden beneath the surface of the Atlantic Ocean". In: *Nature communications* 11.1 (2020), pp. 1–11.

[10] Lucy C Woodall et al. "The deep sea is a major sink for microplastic debris". In: *Royal Society open science* 1.4 (2014), p. 140317.

[11] Victor Onink et al. "Global simulations of marine plastic transport show plastic trapping in coastal zones". In: *Environmental Research Letters* 16.6 (2021), p. 064053.

[12] Outi Setälä et al. "Microplastics in marine food webs". In: *Microplastic contamination in aquatic environments*. Elsevier, 2018, pp. 339–363.

[13] S Feliu, M Morcillo, and B Chico. "Effect of distance from sea on atmospheric corrosion rate". In: *Corrosion* 55.9 (1999), pp. 883–891.

[14] Shengxi Li and LH Hihara. "Aerosol salt particle deposition on metals exposed to marine environments: a study related to marine atmospheric corrosion". In: *Journal of The Electrochemical Society* 161.5 (2014), p. C268.

[15] Josephine Y Aller et al. “The sea surface microlayer as a source of viral and bacterial enrichment in marine aerosols”. In: *Journal of aerosol science* 36.5-6 (2005), pp. 801–812.

[16] AH Woodcock. “Bursting bubbles and air pollution”. In: *Sewage and Industrial Wastes* (1955), pp. 1189–1192.

[17] GM Afeti and FJ Resch. “Distribution of the liquid aerosol produced from bursting bubbles in sea and distilled water”. In: *Tellus B* 42.4 (1990), pp. 378–384.

[18] Henri Lhuissier and Emmanuel Villermaux. “Bursting bubble aerosols”. In: *Journal of Fluid Mechanics* 696 (2012), pp. 5–44.

[19] Duncan C Blanchard and Lawrence D Syzdek. “Water-to-air transfer and enrichment of bacteria in drops from bursting bubbles”. In: *Applied and environmental microbiology* 43.5 (1982), pp. 1001–1005.

[20] Duncan C Blanchard. “The ejection of drops from the sea and their enrichment with bacteria and other materials: a review”. In: *Estuaries* 12.3 (1989), pp. 127–137.

[21] Ruo-Shan Tseng et al. “Sea-to-air transfer of surface-active organic compounds by bursting bubbles”. In: *Journal of Geophysical Research: Oceans* 97.C4 (1992), pp. 5201–5206.

[22] Maria Masry et al. “Experimental evidence of plastic particles transfer at the water-air interface through bubble bursting”. In: *Environmental Pollution* 280 (2021), p. 116949.

[23] Ruei-Feng Shiu et al. “New insights into the role of marine plastic-gels in microplastic transfer from water to the atmosphere via bubble bursting”. In: *Water Research* 222 (2022), p. 118856.

[24] Shanye Yang et al. “Constraining Microplastic Particle Emission Flux from the Ocean”. In: *Environmental Science & Technology Letters* (2022).

[25] Timm Krüger et al. “The lattice Boltzmann method”. In: *Springer International Publishing* 10 (2017), pp. 978–3.

[26] Sydney Chapman, Thomas George Cowling, and David Burnett. *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*. Cambridge, UK: Cambridge university press, 1990.

[27] Acep Purqon et al. “Accuracy and Numerical Stabilty Analysis of Lattice Boltzmann Method with Multiple Relaxation Time for Incompressible Flows”. In: *Journal of Physics: Conference Series*. Vol. 877. 1. IOP Publishing. 2017, p. 012035.

[28] Moritz Lehmann. *High Performance Free Surface LBM on GPUs*. 2019. URL: https://epub.uni-bayreuth.de/5400/.

[29] Moritz Lehmann and Stephan Gekle. “Analytic Solution to the Piecewise Linear Interface Construction Problem and Its Application in Curvature Calculation for Volume-of-Fluid Simulation Codes”. In: *Computation* 10.2 (2022), p. 21.

[30] Moritz Lehmann. “Esoteric Pull and Esoteric Push: Two Simple In-Place Streaming Schemes for the Lattice Boltzmann Method on GPUs”. In: *Computation* 10.6 (2022), p. 92.

[31] Fabian Häusl. *MPI-based multi-GPU extension of the Lattice Boltzmann Method*. 2019. URL: https://epub.uni-bayreuth.de/5689/.

[32] Thomas Pohl. *High performance simulation of free surface flows using the lattice Boltzmann method*. Verlag Dr. Hut, 2008.

[33] Stefan Donath. *Wetting Models for a Parallel High-performance Free Surface Lattice Boltzmann Method: Benetzungsmodelle Für Eine Parallele Lattice-Boltzmann-Methode Mit Freien Oberflächen*. Erlangen, Germany: Verlag Dr. Hut, 2011.

[34] Carolin Körner et al. "Lattice Boltzmann model for free surface flow for modeling foaming". In: *Journal of Statistical Physics* 121.1-2 (2005), pp. 179–196.

[35] Nils Thürey, C Körner, and U Rüde. "Interactive free surface fluids with the lattice Boltzmann method". In: *Technical Report05-4. University of Erlangen-Nuremberg, Germany* (2005).

[36] Martin Schreiber and DTMP Neumann. "GPU based simulation and visualization of fluids with free surfaces". PhD thesis. Diploma Thesis, Technische Universität München, 2010.

[37] Christian Janßen and Manfred Krafczyk. "Free surface flow simulations on GPGPUs using the LBM". In: *Computers & Mathematics with Applications* 61.12 (2011), pp. 3549–3563.

[38] Zhi-Gang Feng and Efstathios E Michaelides. "Drag coefficients of viscous spheres at intermediate and high Reynolds numbers". In: *J. Fluids Eng.* 123.4 (2001), pp. 841–849.

[39] HJ Holterman. *Kinetics and evaporation of water drops in air*. Vol. 2012. Wageningen, Netherlands: Citeseer, 2003.

[40] Huug G Ouwersloot et al. "Vertical wind velocity observations at the Cabauw tower". In: *19th Symposium on Boundary Layers and Turbulence, American Meteorological Society, 2-6 August 2010, Keystone, Colorado, USA*. 2010, P1–2.

[41] Young Kyoung Song et al. "Large accumulation of micro-sized synthetic polymer particles in the sea surface microlayer". In: *Environmental science & technology* 48.16 (2014), pp. 9014–9021.

[42] Doo-Hyeon Chae et al. "Abundance and distribution characteristics of microplastics in surface seawaters of the Incheon/Kyeonggi coastal region". In: *Archives of environmental contamination and toxicology* 69.3 (2015), pp. 269–278.

[43] Zachary T Anderson et al. "A rapid method for assessing the accumulation of microplastics in the sea surface microlayer (SML) of estuarine systems". In: *Scientific Reports* 8.1 (2018), pp. 1–11.

[44] P Ahmadi et al. "Systematic Evaluation of Physical Parameters Affecting the Terminal Settling Velocity of Microplastic Particles in Lakes Using CFD. Front". In: *Environ. Sci* 10 (2022), p. 875220.

[45] Bingqiang Ji, Amrit Singh, and Jie Feng. "Water-to-Air Transfer of Nano/Microsized Particulates: Enrichment Effect in Bubble Bursting Jet Drops". In: *Nano Letters* (2022).

[46] Fabian Häusl. *Soft Objects in Newtonian and Non-Newtonian Fluids: a Computational Study of Bubbles and Capsules in Flow*. 2022. URL: https://epub.uni-bayreuth.de/5960/.

[47] Martin Geier and Martin Schönherr. "Esoteric twist: an efficient in-place streaming algorithmus for the lattice Boltzmann method on massively parallel hardware". In: *Computation* 5.2 (2017), p. 19.

[48] Peter Bailey et al. "Accelerating lattice Boltzmann fluid flow simulations using graphics processors". In: *2009 international conference on parallel processing*. IEEE. 2009, pp. 550–557.

[49] M. Mohrhard et al. "An Auto-Vecotorization Friendly Parallel Lattice Boltzmann Streaming Scheme for Direct Addressing". In: *Computers & Fluids* 181 (2019), pp. 1–7. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2019.01.001.

[50] Adrian Kummerländer et al. *Implicit Propagation of Directly Addressed Grids in Lattice Boltzmann Methods*. 2021.

[51] Martin Schreiber et al. "Free-surface lattice-Boltzmann simulation on many-core architectures". In: *Procedia Computer Science* 4 (2011), pp. 984–993.

[52] Christoph Riesinger et al. "A holistic scalable implementation approach of the lattice Boltzmann method for CPU/GPU heterogeneous clusters". In: *Computation* 5.4 (2017), p. 48.

[53] Eirik O Aksnes and Anne C Elster. "Porous rock simulations and lattice Boltzmann on GPUs". In: *Parallel Computing: From Multicores and GPU's to Petascale*. Amsterdam, Netherlands: IOS Press, 2010, pp. 536–545.

[54] Markus Holzer, Martin Bauer, and Ulrich Rüde. "Highly Efficient Lattice-Boltzmann Multiphase Simulations of Immiscible Fluids at High-Density Ratios on CPUs and GPUs through Code Generation". In: *arXiv preprint arXiv:2012.06144* (2020).

[55] Julien Duchateau et al. "Accelerating physical simulations from a multicomponent Lattice Boltzmann method on a single-node multi-GPU architecture". In: *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*. IEEE. 2015, pp. 315–322.

[56] Wei Li et al. "Efficient Kinetic Simulation of Two-Phase Flows". In: *ACM Transactions on Graphics* 41.4 (2022), p. 114.

[57] Stuart DC Walsh et al. "Accelerating geoscience and engineering system simulations on graphics hardware". In: *Computers & Geosciences* 35.12 (2009), pp. 2353–2364.

[58] Moritz Lehmann et al. "Accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit, and customized 16-bit number formats". In: *Phys. Rev. E* 106 (1 2022), p. 015308. DOI: 10.1103/PhysRevE.106.015308.

[59] Michal Takáč and Ivo Petráš. "Cross-Platform GPU-Based Implementation of Lattice Boltzmann Method Solver Using ArrayFire Library". In: *Mathematics* 9.15 (2021), p. 1793.

[60] Mark J Mawson and Alistair J Revell. "Memory transfer optimization for a lattice Boltzmann solver on Kepler architecture nVidia GPUs". In: *Computer Physics Communications* 185.10 (2014), pp. 2566–2574.

[61] Nicolas Delbosc et al. “Optimized implementation of the Lattice Boltzmann Method on a graphics processing unit towards real-time fluid simulation”. In: *Computers & Mathematics with Applications* 67.2 (2014), pp. 462–475.

[62] Nhat-Phuong Tran, Myungho Lee, and Sugwon Hong. “Performance optimization of 3D lattice Boltzmann flow solver on a GPU”. In: *Scientific Programming* 2017 (2017).

[63] Christian Obrecht et al. “Multi-GPU implementation of the lattice Boltzmann method”. In: *Computers & Mathematics with Applications* 65.2 (2013), pp. 252–261.

[64] Christian Obrecht et al. “A new approach to the lattice Boltzmann method for graphics processing units”. In: *Computers & Mathematics with Applications* 61.12 (2011), pp. 3628–3638.

[65] Christian Feichtinger et al. “A flexible Patch-based lattice Boltzmann parallelization approach for heterogeneous GPU–CPU clusters”. In: *Parallel Computing* 37.9 (2011), pp. 536–549.

[66] Enrico Calore et al. “Massively parallel lattice–Boltzmann codes on large GPU clusters”. In: *Parallel Computing* 58 (2016), pp. 1–24.

[67] Christian Obrecht et al. “Global memory access modelling for efficient implementation of the lattice Boltzmann method on graphics processing units”. In: *International Conference on High Performance Computing for Computational Science*. Springer. 2010, pp. 151–161.

[68] Moritz Lehmann et al. “Ejection of marine microplastics by raindrops: a computational and experimental study”. In: *Microplastics and Nanoplastics* 1.18 (2021), pp. 1–19.

[69] Hannes Laermanns et al. “Tracing the horizontal transport of microplastics on rough surfaces”. In: *Microplastics and Nanoplastics* 1.11 (2021), pp. 1–12.

[70] Hans-Jörg Limbach et al. “ESPResSo – an extensible simulation package for research on soft matter systems”. In: *Computer Physics Communications* 174.9 (2006), pp. 704–727.

[71] Institute for Computational Physics, Universität Stuttgart. *ESPResSo User’s Guide*. http://espressomd.org/wordpress/wp-content/uploads/2016/07/ug_07_2016.pdf. Accessed: 2018-06-15. 2016.

[72] Pei-Yao Hong et al. “Scalable multi-relaxation-time lattice Boltzmann simulations on multi-GPU cluster”. In: *Computers & Fluids* 110 (2015), pp. 1–8.

[73] Wang Xian and Aoki Takayuki. “Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster”. In: *Parallel Computing* 37.9 (2011), pp. 521–535.

[74] Minh Quan Ho et al. “Improving 3D Lattice Boltzmann Method stencil with asynchronous transfers on many-core processors”. In: *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*. IEEE. 2017, pp. 1–9.

[75] Johannes Habich et al. “Performance engineering for the lattice Boltzmann method on GPGPUs: Architectural requirements and performance results”. In: *Computers & Fluids* 80 (2013), pp. 276–282.

[76] Jonas Tölke and Manfred Krafczyk. "TeraFLOP computing on a desktop PC with GPUs for 3D CFD". In: *International Journal of Computational Fluid Dynamics* 22.7 (2008), pp. 443–456.

[77] Gregory Herschlag et al. "GPU data access on complex geometries for D3Q19 lattice Boltzmann method". In: *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2018, pp. 825–834.

[78] Waine B de Oliveira Jr, Alan Lugarini, and Admilson T Franco. "Performance analysis of the lattice Boltzmann method implementation on GPU". In: (2019).

[79] Pablo R Rinaldi et al. "A Lattice-Boltzmann solver for 3D fluid simulation on GPU". In: *Simulation Modelling Practice and Theory* 25 (2012), pp. 163–171.

[80] Pablo R Rinaldi et al. "Fluid Simulation with Lattice Boltzmann Methods Implemented on GPUs Using CUDA". In: 2009.

[81] Jeff Ames et al. "Multi-GPU immersed boundary method hemodynamics simulations". In: *Journal of Computational Science* 44 (2020), p. 101153.

[82] QinGang Xiong et al. "Efficient parallel implementation of the lattice Boltzmann method on large clusters of graphic processing units". In: *Chinese Science Bulletin* 57.7 (2012), pp. 707–715.

[83] Hongyin Zhu et al. "An Efficient Graphics Processing Unit Scheme for Complex Geometry Simulations Using the Lattice Boltzmann Method". In: *IEEE Access* 8 (2020), pp. 185158–185168.

[84] Frédéric Kuznik et al. "LBM based flow simulation using GPU computing processor". In: *Computers & Mathematics with Applications* 59.7 (2010), pp. 2380–2392.

[85] Adrian Horga. "With lattice Boltzmann models using CUDA enabled GPGPUs". In: *Master Thesis* (2013).

[86] Markus Geveler et al. "Lattice-Boltzmann simulation of the shallow-water equations with fluid-structure interaction on multi-and manycore processors". In: *Facing the multicore-challenge*. Wiesbaden, Germany: Springer, 2010, pp. 92–104.

[87] Joel Beny and Jonas Latt. "Efficient LBM on GPUs for dense moving objects using immersed boundary condition". In: *arXiv preprint arXiv:1904.02108* (2019).

[88] Predrag M Tekic, Jelena B Radjenovic, and Milos Rackovic. "Implementation of the Lattice Boltzmann method on heterogeneous hardware and platforms using OpenCL". In: *Advances in Electrical and Computer Engineering* 12.1 (2012), pp. 51–56.

[89] Joël Bény, Christos Kotsalos, and Jonas Latt. "Toward full GPU implementation of fluid-structure interaction". In: *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE. 2019, pp. 16–22.

[90] G Boroni, J Dottori, and P Rinaldi. "FULL GPU implementation of lattice-Boltzmann methods with immersed boundary conditions for fast fluid simulations". In: *The International Journal of Multiphysics* 11.1 (2017), pp. 1–14.

[91] Michael Griebel, Marc Alexander Schweitzer, et al. *Meshfree methods for partial differential equations II*. Cham, Switzerland: Springer, 2005.

[92] Stefan Zitz, Andrea Scagliarini, and Jens Harting. "Lattice Boltzmann simulations of stochastic thin film dewetting". In: *Physical Review E* 104.3 (2021), p. 034801.

[93] Christian F Janßen et al. "Validation of the GPU-accelerated CFD solver ELBE for free surface flow problems in civil and environmental engineering". In: *Computation* 3.3 (2015), pp. 354–385.

[94] Johannes Habich et al. "Performance analysis and optimization strategies for a D3Q19 lattice Boltzmann kernel on nVIDIA GPUs using CUDA". In: *Advances in Engineering Software* 42.5 (2011), pp. 266–272.

[95] Enrico Calore et al. "Optimizing communications in multi-GPU Lattice Boltzmann simulations". In: *2015 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE. 2015, pp. 55–62.

[96] Naoyuki Onodera et al. "Locally mesh-refined lattice Boltzmann method for fuel debris air cooling analysis on GPU supercomputer". In: *Mechanical Engineering Journal* 7.3 (2020), pp. 19–00531.

[97] Giacomo Falcucci et al. "Extreme flow simulations reveal skeletal adaptations of deep-sea sponges". In: *Nature* 595.7868 (2021), pp. 537–541.

[98] Stefan Zitz et al. "Lattice Boltzmann method for thin-liquid-film hydrodynamics". In: *Physical Review E* 100.3 (2019), p. 033313.

[99] Cao Wei et al. "An improved LBM approach for heterogeneous GPU-CPU clusters". In: *2011 4th International Conference on Biomedical Engineering and Informatics (BMEI)*. Vol. 4. IEEE. 2011, pp. 2095–2098.

[100] Farrel Gray and Edo Boek. "Enhancing computational precision for lattice Boltzmann schemes in porous media flows". In: *Computation* 4.1 (2016), p. 11.

[101] C Anela Choy et al. "The vertical distribution and biological transport of marine microplastics across the epipelagic and mesopelagic water column". In: *Scientific reports* 9.1 (2019), pp. 1–9.

[102] Ana L d F Lacerda et al. "Plastics in sea surface waters around the Antarctic Peninsula". In: *Scientific reports* 9.1 (2019), pp. 1–12.

[103] Trishan Naidoo and David Glassom. "Sea-surface microplastic concentrations along the coastal shelf of KwaZulu–Natal, South Africa". In: *Marine pollution bulletin* 149 (2019), p. 110514.

[104] Tamara Gajšt et al. "Sea surface microplastics in Slovenian part of the Northern Adriatic". In: *Marine pollution bulletin* 113.1-2 (2016), pp. 392–399.

[105] Hui Wang et al. "Dynamics of high-speed drop impact on deep liquid pool". In: *Conference on Modelling Fluid Flow (CMFF'22)*. Department of Fluid Mechanics Budapest University of Technology and Economics, 2022.

[106] Simon Bogner, Ulrich Rüde, and Jens Harting. "Curvature estimation from a volume-of-fluid indicator function for the simulation of surface tension and wetting with a free-surface lattice Boltzmann method". In: *Physical Review E* 93.4 (2016), p. 043302.

[107] Charles S Peskin. "The immersed boundary method". In: *ANU* 11 (July 2003), pp. 479–517.

[108] Timm Krüger. "Introduction to the immersed boundary method". In: *LBM Workshop, Edmonton*. 2011.

[109] Stefan Frijters, Timm Krüger, and Jens Harting. "Parallelised Hoshen–Kopelman algorithm for lattice-Boltzmann simulations". In: *Computer Physics Communications* 189 (2015), pp. 92–98.

[110] Oren Breslouer. "Rayleigh-Plateau Instability: Falling Jet". In: *Project Report* (2010).

[111] Marise V Gielen et al. "Oblique drop impact onto a deep liquid pool". In: *Physical review fluids* 2.8 (2017), p. 083602.

[112] Sten A Reijers et al. "Oblique droplet impact onto a deep liquid pool". In: *arXiv preprint arXiv:1903.08978* (2019).

[113] An-Bang Wang and Chi-Chang Chen. "Splashing impact of a single drop onto very thin liquid films". In: *Physics of fluids* 12.9 (2000), pp. 2155–2158.

[114] Gangtao Liang et al. "Crown behavior and bubble entrainment during a drop impact on a liquid film". In: *Theoretical and Computational Fluid Dynamics* 28.2 (2014), pp. 159–170.

[115] David W Murphy et al. "Splash behaviour and oily marine aerosol production by raindrops impacting oil slicks". In: *Journal of Fluid Mechanics* 780 (2015), p. 536.

[116] Federico Porcù et al. "Effects of altitude on maximum raindrop size and fall velocity as limited by collisional breakup". In: *Journal of the atmospheric sciences* 70.4 (2013), pp. 1129–1134.

[117] John H van Boxel et al. "Numerical model for the fall speed of rain drops in a rain fall simulator". In: *Workshop on wind and water erosion*. 1997, pp. 77–85.

[118] Athelstan F Spilhaus. "Raindrop size, shape and falling speed". In: *Journal of Meteorology* 5.3 (1948), pp. 108–110.

[119] LV Zhang et al. "Evolution of the ejecta sheet from the impact of a drop with a deep pool". In: *Journal of Fluid Mechanics* 690 (2012), pp. 5–15.

[120] John S Marshall and W Mc K Palmer. "The distribution of raindrops with size". In: *Journal of meteorology* 5.4 (1948), pp. 165–166.

[121] Emmanuel Villermaux and Benjamin Bossa. "Single-drop fragmentation determines size distribution of raindrops". In: *Nature Physics* 5.9 (2009), pp. 697–702.

[122] Philip Rice and Nettie Holmberg. "Cumulative time statistics of surface-point rainfall rates". In: *IEEE Transactions on Communications* 21.10 (1973), pp. 1131–1136.

[123] Robert F Adler et al. "Global precipitation: Means, variations and trends during the satellite era (1979–2014)". In: *Surveys in Geophysics* 38.4 (2017), pp. 679–699.

[124] Madeline C Evans and Christopher S Ruf. "Toward the Detection and Imaging of Ocean Microplastics With a Spaceborne Radar". In: *IEEE Transactions on Geoscience and Remote Sensing* (2021).

[125] David Wichmann, Philippe Delandmeter, and Erik van Sebille. "Influence of near-surface currents on the global dispersal of marine microplastic". In: *Journal of Geophysical Research: Oceans* 124.8 (2019), pp. 6086–6096.

[126] David Wichmann et al. "Mixing of passive tracers at the ocean surface and its implications for plastic transport modelling". In: *Environmental Research Communications* 1.11 (2019), p. 115001.

[127] Donald E Spiel. "The sizes of the jet drops produced by air bubbles bursting on sea-and fresh-water surfaces". In: *Tellus B: Chemical and physical meteorology* 46.4 (1994), pp. 325–338.

[128] Jin Wu. "Jet drops produced by bubbles bursting at the surface of seawater". In: *Journal of physical oceanography* 32.11 (2002), pp. 3286–3290.

[129] Peter LL Walls, Louis Henaux, and James C Bird. "Jet drops from bursting bubbles: How gravity and viscosity couple to inhibit droplet production". In: *Physical Review E* 92.2 (2015), p. 021002.

[130] Elisabeth Ghabache and Thomas Séon. "Size of the top jet drop produced by bubble bursting". In: *Physical Review Fluids* 1.5 (2016), p. 051901.

[131] Peter LL Walls et al. "Quantifying the potential for bursting bubbles to damage suspended cells". In: *Scientific reports* 7.1 (2017), pp. 1–9.

[132] Luc Deike et al. "Dynamics of jets produced by bursting bubbles". In: *Physical Review Fluids* 3.1 (2018), p. 013603.

[133] Alexis Berny et al. "Role of all jet drops in mass transfer from bursting bubbles". In: *Physical Review Fluids* 5.3 (2020), p. 033605.

[134] Alexis Berny et al. "Statistics of jet drop production". In: *Geophysical Research Letters* 48.10 (2021), e2021GL092919.

[135] Francisco J Blanco-Rodríguez and JM Gordillo. "On the jets produced by drops impacting a deep liquid pool and by bursting bubbles". In: *Journal of Fluid Mechanics* 916 (2021).

[136] Masashi Sakai et al. "Enrichment of suspended particles in top jet drops from bursting bubbles". In: *Journal of colloid and interface science* 125.2 (1988), pp. 428–436.

[137] Elisabeth Ghabache et al. "On the physics of fizziness: How bubble bursting controls droplets ejection". In: *Physics of Fluids* 26.12 (2014), p. 121701.

[138] Roland Clift, John R Grace, and Martin E Weber. "Bubbles, drops, and particles". In: (2005).

[139] N Kugou, K Ishida, and A Yoshida. "Experimental study on motion of air bubbles in seawater (terminal velocity and drug coefficient of air bubble rising in seawater)". In: *WIT Transactions on The Built Environment* 68 (2003).

[140] Ahmed Al Harraq and Bhuvnesh Bharti. "Microplastics through the Lens of Colloid Science". In: *ACS Environmental Au* 2.1 (2021), pp. 3–10.

[141] Yang Changfu et al. "Interface behavior changes of weathered polystyrene with ciprofloxacin in seawater environment". In: *Environmental Research* 212 (2022), p. 113132.

[142] Guangzhou Liu et al. "Sorption behavior and mechanism of hydrophilic organic chemicals to virgin and aged microplastics in freshwater and seawater". In: *Environmental Pollution* 246 (2019), pp. 26–33.

[143] Stefan Gustavson. *Perlin Noise*. Accessed 12 Feb 2021. 2020. URL: https://github.com/stegu/perlin-noise (visited on 02/12/2021).

[144] G Wellein et al. "Towards optimal performance for lattice Boltzmann applications on terascale computers". In: *Parallel Computational Fluid Dynamics 2005*. Amsterdam, Netherlands: Elsevier, 2006, pp. 31–40.

[145] Markus Wittmann et al. "Comparison of different propagation steps for lattice Boltzmann methods". In: *Computers & Mathematics with Applications* 65.6 (2013), pp. 924–935.

[146] Markus Wittmann. "Hardware-effiziente, hochparallele Implementierungen von Lattice-Boltzmann-Verfahren für komplexe Geometrien". In: (2016).

[147] M.J. Krause. "Fluid Flow Simulation and Optimisation with Lattice Boltzmann Methods on High Performance Computers: Application to the Human Respiratory System". PhD thesis. Karlsruhe Institute of Technology (KIT), Universität Karlsruhe (TH), 2010. URL: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000019768.

[148] Sauro Succi et al. "Towards exascale lattice Boltzmann computing". In: *Computers & Fluids* 181 (2019), pp. 107–115.

[149] Dominique d'Humières. "Multiple–relaxation–time lattice Boltzmann models in three dimensions". In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 360.1792 (2002), pp. 437–451.

[150] Moritz Lehmann. *FluidX3D*. 2022. URL: https://github.com/ProjectPhysX/FluidX3D.

[151] Kuang Ma et al. "A Simple One-step Index Algorithm for Implementation of Lattice Boltzmann Method on GPU". In: *Computer Physics Communications* (2022), p. 108603.

[152] PA Skordos. "Initial and boundary conditions for the lattice Boltzmann method". In: *Physical Review E* 48.6 (1993), p. 4823.

[153] Fabio Bonaccorso et al. "Lbsoft: A parallel open-source software for simulation of colloidal systems". In: *Computer Physics Communications* 256 (2020), p. 107455.

[154] Thomas Grützmacher and Hartwig Anzt. "A modular precision format for decoupling arithmetic format and storage format". In: *European Conference on Parallel Processing*. Springer. 2018, pp. 434–443.

[155] Hartwig Anzt et al. "Toward a modular precision ecosystem for high-performance computing". In: *The International Journal of High Performance Computing Applications* 33.6 (2019), pp. 1069–1078.

[156] Xiaoyi He and Li-Shi Luo. "Lattice Boltzmann model for the incompressible Navier–Stokes equation". In: *Journal of statistical Physics* 88.3 (1997), pp. 927–944.

[157] IEEE Computer Society. Standards Committee and American National Standards Institute. "IEEE standard for binary floating-point arithmetic". In: *IEEE Std 754-2019 (Revision of IEEE 754-2008)* 754 (1985).

[158] David Goldberg. "What every computer scientist should know about floating-point arithmetic". In: *ACM computing surveys (CSUR)* 23.1 (1991), pp. 5–48.

[159] IS Committee and others. "754–2008 IEEE standard for floating-point arithmetic". In: *IEEE Computer Society Std* 2008 (2008).

[160] William Kahan. "IEEE standard 754 for binary floating-point arithmetic". In: *Lecture Notes on the Status of IEEE* 754.94720-1776 (1996), p. 11.

[161] John L Gustafson and Isaac T Yonemoto. "Beating floating point at its own game: Posit arithmetic". In: *Supercomputing Frontiers and Innovations* 4.2 (2017), pp. 71–86.

[162] Timm Krüger. "Unit conversion in LBM". In: *LBM Workshop, Edmonton, Canada*. Vol. 8. 2011.

[163] Geoffrey Ingram Taylor and Albert Edward Green. "Mechanism of the production of small eddies from large ones". In: *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences* 158.895 (1937), pp. 499–521.

[164] Th. von Karman. "Ueber den Mechanismus des Widerstandes, den ein bewegter Körper in einer Flüssigkeit erfährt". In: *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse* (1911), pp. 509–517.

[165] Alberto Cattenone, Simone Morganti, and Ferdinando Auricchio. "Basis of the Lattice Boltzmann Method for Additive Manufacturing". In: *Archives of Computational Methods in Engineering* 27.4 (2020), pp. 1109–1133.

[166] Usman R Alim, Alireza Entezari, and Torsten Moller. "The lattice-Boltzmann method on optimal sampling lattices". In: *IEEE Transactions on Visualization and Computer Graphics* 15.4 (2009), pp. 630–641.

[167] Philipp Neumann and Tobias Neckel. "A dynamic mesh refinement technique for Lattice Boltzmann simulations on octree-like grids". In: *Computational Mechanics* 51.2 (2013), pp. 237–253.

[168] UKNG Ghia, Kirti N Ghia, and CT Shin. "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method". In: *Journal of computational physics* 48.3 (1982), pp. 387–411.

[169] Bo-Nan Jiang, TL Lin, and Louis A Povinelli. "Large-scale computation of incompressible viscous flow by least-squares finite element method". In: *Computer Methods in Applied Mechanics and Engineering* 114.3-4 (1994), pp. 213–231.

[170] Jaw-Yen Yang et al. "Implicit weighted ENO schemes for the three-dimensional incompressible Navier–Stokes equations". In: *Journal of Computational Physics* 146.1 (1998), pp. 464–487.

[171] Dominique Barthes-Biesel. "Motion and Deformation of Elastic Capsules and Vesicles in Flow". In: *Annual Review of Fluid Mechanics* 48.1 (Jan. 2016), pp. 25 –52. DOI: 10.1146/annurev-fluid-122414-034345.

[172] Achim Guckenberger et al. "On the bending algorithms for soft objects in flows". In: *Computer Physics Communications* 207 (2016), pp. 1–23.

[173] Achim Guckenberger and Stephan Gekle. "Theory and algorithms to compute Helfrich bending forces: A review". In: *Journal of Physics: Condensed Matter* 29.20 (2017), p. 203001.

[174] J. E. Bresenham. “Algorithm for computer control of a digital plotter”. In: *IBM Systems Journal* 4.1 (1965), pp. 25–30. DOI: 10.1147/sj.41.0025.

[175] Paul Bourke. *Polygonising a scalar field*. 1994.

[176] Tomas Möller and Ben Trumbore. “Fast, minimum storage ray/triangle intersection”. In: *ACM SIGGRAPH 2005 Courses*. 2005, 7–es.

[177] Moritz Lehmann, Sebastian Johannes Müller, and Stephan Gekle. “Efficient viscosity contrast calculation for blood flow simulations using the lattice Boltzmann method”. In: *International Journal for Numerical Methods in Fluids* 92.11 (2020), pp. 1463–1477.

Chapter 6

Acknowledgements

Starting from nothing and getting to this point in my life has not been easy. I feel very grateful to have had many strong supporters, especially in the phases of my life where I was not strong.

I thank my brother, Maximilian, for accompanying me throughout my life with the greatest friendship and support. I thank my former school director, Gerd Koppitz, for welcoming me with open arms into his Gymnasium, and for bringing me on the right track. I thank my grandpa, Anton Haas, for his tireless love and support especially during my studies. I thank my friends Julia Lang and Niklas Stenger for making studying physics much more enjoyable. Most notably I thank my supervisor, Prof. Dr. Stephan Gekle, for believing in me from the very first day we met, and for supporting me since my bachelor's thesis all the way to my PhD, always with guidance and good advice. I thank Fabian Häusl for working with me on the software for such a long time, and for contributing so many ideas and knowledge. I thank all the others at our working group for the nice time in the office. I thank Claudia Brandt for her excellent administration at our working group, and I thank Markus Hilt for the technical support. I thank Melanie Pöhlmann for being the kindest and best SFB coordinator, and I thank all the many people in the SFB Mikroplastik for the collaborations, discussions and their overwhelming inclusiveness and support. I thank everyone who collaborated with me, especially Lisa Marie Oehlschlägel, Mathias J. Krause, Giorgio Amati, Hannes Laermanns, Martin Löder, Christina Bogner, Christoph Schwarzmeier, Wolfgang Groß, Simon Wieland and Mauro Sbragaglia. I thank Jan-Pascal Boos and Marcel Meinhart for the fun time at many conferences and SFB symposia. I thank my roommates in the WG for becoming close friends over the years.

I also thank the Nvidia Corporation for making me spend less time waiting for computations and instead focus more my research, by donating a Titan Xp GPU. And I thank the Bayreuth Centre for High Performance Computing (BZHPC), the Leibnitz-Rechenzentrum (LRZ) and the Jülich Supercomputing Centre (JSC), especially Andreas Herten, for providing computational resources and support.

Chapter 7

Publications

7.1 The Author's Contributions

7.1.1 Major Publications

[**Pub1**] Moritz Lehmann, Lisa Marie Oehlschlägel, Fabian Häusl, Andreas Held, and Stephan Gekle. "Ejection of marine microplastics by raindrops: a computational and experimental study". In: *Microplastics and Nanoplastics* 1.18 (2021), pp. 1–19.

Breakdown of the individual contributions:

- ML designed the study (80%), wrote the simulation software for the raindrop simulations (90%), conducted the simulations (100%), analyzed the data with additional simulation scripts (100%), created all figures except figure 12 (a) and 12 (b), contributed literature research (70%), and wrote the manuscript (90%) and SI (95%).
- LMO conducted the experiments and contributed to literature research and to writing and reviewing the manuscript.
- FH contributed to the software used for the simulations and to reviewing the manuscript.
- AH contributed to conducting the experiments, to literature research, and to writing and reviewing the manuscript.
- SG contributed to the design of the study, to literature research, and to writing and reviewing the manuscript.

[**Pub2**] Moritz Lehmann, Fabian Häusl, and Stephan Gekle. "Modeling of vertical microplastic transport by rising bubbles". In: *Microplastics and Nanoplastics* 3.4 (2023), pp. 1–6.

Breakdown of the individual contributions:

- ML contributed to implementing and validating the simulation software for the bubble simulations (50%), designed the study (80%), conducted the simulations (100%), evaluated the data (100%), created all figures, did literature research (80%), and wrote the manuscript (90%).

- FH contributed to implementing and validating the simulation software for the bubble simulations, and reviewed the manuscript.
- SG contributed to study design and to literature research and reviewed the manuscript.

[**Pub3**] Moritz Lehmann, Mathias J Krause, Giorgio Amati, Marcello Sega, Jens Harting, and Stephan Gekle. "Accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit, and customized 16-bit number formats". In: *Physical Review E* 106, 015308 (2022).

Breakdown of the individual contributions:

- ML contributed the original concept (85%), designed the study (70%), wrote the simulation software for this study (100%), conducted the simulations (95%), evaluated the data (100%), created all figures, did the literature research (85%), and wrote the manuscript (90%).
- MK contributed the original concept, contributed essential ideas, contributed to review the manuscript, and contributed to literature research.
- GA contributed essential ideas, contributed to review the manuscript, and contributed to literature research, and contributed test setups and benchmarks.
- MS contributed test setups and benchmarks and contributed to review the manuscript.
- JH contributed test setups and benchmarks and contributed to review the manuscript.
- SG contributed essential ideas, contributed to review the manuscript, and contributed to literature research.

[**Pub4**] Moritz Lehmann. "Esoteric Pull and Esoteric Push: Two Simple In-Place Streaming Schemes for the Lattice Boltzmann Method on GPUs". In: *Computation* 10.6 (2022), p. 92.

[**Pub5**] Hannes Laermanns, Moritz Lehmann, Marcel Klee, Martin GJ Löder, Stephan Gekle, and Christina Bogner. "Tracing the horizontal transport of microplastics on rough surfaces". In: *Microplastics and Nanoplastics* 1.11 (2021), pp. 1–12.

Breakdown of the individual contributions:

- ML simulated the water flow (100%), created figure 5, contributed to discussion of the results (15%), contributed to writing the first draft of the manuscript (5%).
- HL contributed to design of the the study, supervised the experimental work, contributed to discussion of the results, and contributed to writing the first draft of the manuscript.
- MK contributed to design of the the study, did the irrigation experiments, and contributed to discussion of the results.
- MGJL prepared and characterized the PMMA particles, and contributed to discussion of the results.

- SG contributed to discussion of the results.
- CB contributed to design of the the study, wrote the code and analyzed the image data, contributed to discussion of the results, and contributed to writing the first draft of the manuscript.

[**Pub6**] Christoph Schwarzmeier, Markus Holzer, Travis Mitchell, Moritz Lehmann, Fabian Häusl, and Ulrich Rüde: "Comparison of free surface and conservative Allen-Cahn phase field lattice Boltzmann method". In: *Journal of Computational Physics* 111753 (2022).

Breakdown of the individual contributions:

- ML contributed to methodology (20%), software (20%), validation(15%), investigation (20%), writing original draft (10%), and visualization (figures 21-24).
- CS contributed conceptualization, methodology, software, validation, investigation, data curation, writing the original draft, review & editing, visualization, project administration, and funding acquisition.
- MH contributed methodology, software, validation, investigation, and writing the original draft.
- TM contributed methodology, software, validation, investigation, writing the original draft, and review & editing.
- FH contributed software and review & editing.
- UR contributed resources, review & editing, supervision, and funding acquisition.

[**Pub7**] Lisa Marie Oehlschlägel, Sebastian Schmid, Moritz Lehmann, Stephan Gekle, and Andreas Held. "Water-air transfer rates of microplastic particles through bubble bursting as a function of particle size". Manuscript submitted for peer-review in *Microplastics and Nanoplastics*, (2022).

Breakdown of the individual contributions:

- ML made the formal analysis on environmental implications (90%), and contributed to writing (5%) and review (20%).
- LMO did the conceptualization, methodology, investigation and writing, and contributed to the visualization.
- SS was responsible for methodology, investigation and writing, and contributed to the visualization and review.
- SG contributed the funding acquisition and contributed to review & editing.
- AH was responsible for project administration, funding acquisition, conceptualization, writing and the resources, and contributed to the visualization and review.

[**Pub8**] Moritz Lehmann. "Combined scientific CFD simulation and interactive raytracing with OpenCL". In: *International Workshop on OpenCL*. (2022), pp. 1–2.

7.1.2 Further Publications

[29] Moritz Lehmann and Stephan Gekle. "Analytic Solution to the Piecewise Linear Interface Construction Problem and Its Application in Curvature Calculation for Volume-of-Fluid Simulation Codes". In: *Computation* 10.2 (2022), p. 21.

[177] Moritz Lehmann, Sebastian J Müller, and Stephan Gekle. "Efficient viscosity contrast calculation for blood flow simulations using the lattice Boltzmann method". In: *International Journal for Numerical Methods in Fluids* 92.11 (2020): 1463-1477.

Wolfgang Gross, Anja FRM Ramsperger, Simon Wieland, Moritz Lehmann, Thomas Witzmann, Stephan Gekle, Günter Auernhammer, Andreas Fery, Christian Laforsch, and Holger Kress: "Same but different: the choice of model microplastics strongly affects particle-cell interactions". Manuscript in preparation, (2022).

7.1.3 Conference Contributions

- MICRO2020 talk: Moritz Lehmann, Fabian Häusl, and Stephan Gekle. LBM Simulations of Raindrop Impacts demonstrate Microplastic Transport from the Ocean into the Atmosphere.
  https://micro2020.sciencesconf.org/334143/document
- DSFD2021 talk: Moritz Lehmann, Mathias J. Krause, Giorgio Amati, and Stephan Gekle. On the impact of FP64, FP32 and FP16 floating-point precision on accuracy and performance of lattice Boltzmann method simulations.
  https://youtu.be/XyF7HWjelcs?t=1432
- IWOCL & SYCLcon 2022 technical talk: Moritz Lehmann. Combined scientific CFD simulation and interactive raytracing with OpenCL.
  https://youtu.be/9q3lXsrVF30
- EGU22 talk: Moritz Lehmann, Lisa Marie Oehlschlägel, Fabian Häusl, Andreas Held, and Stephan Gekle. Ejection of marine microplastics by raindrops: a computational and experimental study.
  https://doi.org/10.5194/egusphere-egu22-1506
- DPG Regensburg22 talk: Moritz Lehmann, Lisa Marie Oehlschlägel, Fabian Häusl, Andreas Held, and Stephan Gekle. Ejection of marine microplastics by raindrops: a computational and experimental study.
  https://www.dpg-verhandlungen.de/year/2022/conference/regensburg/part/dy/session/34/contribution/1

Chapter 8

Attached Publications

8.1 Publication 1

Ejection of marine microplastics by raindrops: a computational and experimental study

by

Moritz Lehmann, Lisa Marie Oehlschlägel, Fabian Häusl, Andreas Held, and Stephan Gekle

Microplastics and Nanoplastics 1.18 (2021), pp. 1–19
https://doi.org/10.1186/s43591-021-00018-8

Ejection of marine microplastics by raindrops: a computational and experimental study

Moritz Lehmann[1*], Lisa Marie Oehlschlägel[2], Fabian Häusl[1], Andreas Held[2] and Stephan Gekle[1]

November 12, 2021

*Correspondence:
moritz.lehmann@uni-bayreuth.de
[1]Biofluid Simulation and Modeling – Theoretische Physik VI, University of Bayreuth
[2]Technischer Umweltschutz, Fachgebiet Umweltchemie und Luftreinhaltung, Technische Universität Berlin

Abstract

Raindrops impacting water surfaces such as lakes or oceans produce myriads of tiny droplets which are ejected into the atmosphere at very high speeds. Here we combine computer simulations and experimental measurements to investigate whether these droplets can serve as transport vehicles for the transition of microplastic particles with diameters of a few tens of μm from ocean water to the atmosphere. Using the Volume-of-Fluid lattice Boltzmann method, extended by the immersed-boundary method, we performed more than 1600 raindrop impact simulations and provide a detailed statistical analysis on the ejected droplets. Using typical sizes and velocities of real-world raindrops – parameter ranges that are very challenging for 3D simulations – we simulate straight impacts with various raindrop diameters as well as oblique impacts. We find that a 4 mm diameter raindrop impact on average ejects more than 167 droplets. We show that these droplets indeed contain microplastic concentrations similar to the ocean water within a few millimeters below the surface. To further assess the plausibility of our simulation results, we conduct a series of laboratory experiments, where we find that microplastic particles are indeed contained in the spray. Based on our results and known data – assuming an average microplastic particle concentration of 2.9 particles per liter at the ocean surface – we estimate that, during rainfall, about 4800 microplastic particles transition into the atmosphere per square kilometer per hour for a typical rain rate of 10 $\frac{\text{mm}}{\text{h}}$ and vertical updraft velocity of 0.5 $\frac{\text{m}}{\text{s}}$.

Keywords microplastic; ocean; atmosphere; transport; raindrop; sea spray; lattice Boltzmann method; Volume-of-Fluid; GPU

1 Introduction

Large water basins such as oceans or lakes are commonly considered as sinks where microplastic produced on land surfaces will accumulate over time [1–4], especially in coastal waters [5]. Atmospheric winds, on the other hand, can act as efficient transporters of microplastic leading to long-range, in fact even global, redistribution of atmospherically suspended microplastic [6–9]. Taken together, these two observations suggest that a mechanism transporting microplastic from the hydro- to the atmosphere might contribute significantly to the global spreading of microplastic. Indeed, hydrodynamic processes such as bursting bubbles [10–13] or breaking waves eject myriads of small water droplets into the air thus constituting an important mechanism for the transport of sea salt [14], organic material [15, 16] or particles [17, 18] which can have sizes up to 100 μm [19]. It is to be expected that these processes are most relevant for particles near the ocean surface [20–23] and especially those that accumulate directly at the surface due to hydrophobicity, low density and/or bubble scavenging [24] as is very often the case for microplastic [25–27]. Indeed, evidence has recently been provided that these transport mechanisms could be relevant for microplastic as well [7, 28].
Another droplet-producing mechanism that has received much less attention is the impact of raindrops onto ocean or lake surfaces. Upon contact of the raindrop with the water surface, a thin wall of fluid around the impact site shoots upward at high speed. Due to hydrodyncamic instabilities, any distortion in this ring of fluid amplifies, leading to an uneven breakup into small droplets that resembles a crown in appearance. Depending on the raindrop diameter which is between 1 and 7 mm [29–33], each impact can eject more than hundred droplets during

the initial splash. Quantifying this process is therefore crucial to understand if and how raindrop impacts can act as a possible pathway for microplastic transition from the hydro- into the atmosphere. While there are many experimental observations on various types of drop impacts into water [34–44] – some even investigating particle transport [45] – significantly fewer works study raindrop impacts at terminal velocity [46–48]. This may be due to the fact that the drop height required to reach terminal velocity is several meters [30, 49]. Similarly, numerical simulations treating impact scenarios in the parameter range relevant for raindrops are also limited [50].

Here, we use a novel state-of-the-art GPU implementation of the Volume-of-Fluid lattice Boltzmann method (LBM), combined with the immersed-boundary method [51–53]. The LBM is a powerful tool for simulating fluid flow in countless fields such as microfluidics for medical applications and engineering; we use it due to its exceptional computational efficiency on graphics processing units (GPUs). We simulate more than 1600 impacts of raindrops with different diameters and impact angle, an amount unfeasible with other computational methods. We include microplastic of varying densities into the simulations to examine the potential for ejection. For each setting, we determine the size, altitude and airborne time distribution of droplets and ejected microplastic particles. In addition, we conduct laboratory experiments demonstrating the presence of microplastic particles in splash droplets after the impact of an artificial "raindrop" in good qualitative agreement with our simulations.
Based on our observations for single raindrops of different diameters, the raindrop size distribution [31, 33], typical microplastics concentrations in sea surface water [3, 54], precipitation data [55, 56] and typical vertical wind speeds close to the ground [57], we estimate the amount of microplastics that transition from the global oceans into the atmosphere annually due to raindrop impacts.

2 Methods

2.1 Fluid Solver

2.1.1 The Lattice Boltzmann Method

For solving the Navier-Stokes equations, we use the simulation software *FluidX3D* [51–53], a full (multi) GPU implementation [53, 58–65] of the lattice Boltzmann method [66–68] which we thoroughly validated in-house (see SI section S2 and [51]). We use the single-relaxation-time collision operator [66], as both two-relaxation-time [66, 69] and multiple-relaxation-time [70–72] turned out to be unstable at such high Reynolds numbers in combination with Volume-of-Fluid. Gravity is incorporated using the Guo forcing scheme [66, 73]. The simulations are done in single-precision (FP32) floating-point.

2.1.2 The Volume-of-Fluid Model

FluidX3D contains a full GPU implementation of the Volume-of-Fluid (VoF) model [51, 52, 74–79]. VoF introduces three flag types for LBM lattice points: *fluid*, *interface* and *gas*. The *fluid* phase is computed with regular LBM, the *interface* is kept sharp (width of a single lattice point) at any time and the *gas* phase is not calculated at all. On the *interface* layer, surface tension is handled based on the Young-Laplace equation and surface curvature, which is calculated using the paraboloid fit method [51, 74, 80], built upon the full analytic solution to the piecewise linear interface construction (PLIC) problem [52, 80–83].

2.1.3 Hardware and Illustrations

Simulations are performed in parallel on up to four AMD Radeon VII GPUs (16 GB video memory each) as well as on a Nvidia Titan Xp GPU (12 GB video memory). With our efficient GPU implementation, the compute time for one impact simulation at $L_x = 464$ on a single Radeon VII is between 5 and 15 minutes depending on data acquisition.
For grahpical illustrations, we use an OpenCL implementation of the marching-cubes algorithm [51, 84] that has direct read-only access to the raw simulation data in video memory. This way, rendering is fully parallelized on the GPU. On multiple GPUs, each one renders its own simulation domain and the individual images are stitched together based on their accompanying z-buffer. Lines in the images indicate the multi-GPU domain boundaries.

2.2 Microplastic Particles

2.2.1 The Immersed-Boundary Method

The immersed-boundary method (IBM) [51, 85, 86] couples particles to the fluid and is implemented fully parallelized on the GPU using trilinear velocity interpolation [87] (no-slip condition) and floating-point atomic addition [88]. IBM ensures proper two-way coupling between particles and fluid, whilst allowing particles to move freely between LBM lattice points. Each IBM particle corresponds to one microplastic particle.

2.2.2 Particle Properties

The individual IBM particles have no coupling forces among each other, but they are buoyant and also

prohibited from leaving the water and getting into the air. This is realized with a hard potential perpendicular to the local surface normal for particles between *interface* and *gas* VoF lattice points.
The particles are mathematical points without an intrinsic size. However there is a length scale of hydrodynamic interaction – the distance between two neighboring LBM lattice points converted back into SI-units – providing an upper bound for the particle diameter. Since the physical grid distance is adjusted to the diameter of the impacting raindrop, this upper limit for the microplastic particles d_p is directly proportional to the raindrop diameter d and varies between $d_p \lessapprox 22\,\mu$m and $d_p \lessapprox 151\,\mu$m for the 1 mm and 7 mm raindrops, respectively. This represents a realistic scenario as particles between $50 - 80\,\mu$m are especially abundant in nature [3]. More details are given in S1.1.

2.3 Analysis

2.3.1 Droplet Detection

Whenever a droplet touches the ceiling of the simulation box (i.e. in the uppermost lattice layer at least one lattice point becomes either *interface* or *fluid*), a Hoshen-Kopelman tracking algorithm [89] is triggered and all droplets in the simulation box are indexed. The droplets touching the ceiling are identified, measured and removed. Then the simulation continues until the next trigger event is detected.
Velocity $\vec{u}_0$ and radius R_0 are provided by the tracking algorithm for each droplet individually at the moment when the droplet touches the ceiling. For all *fluid* and *interface* lattice points belonging to a droplet, we average the velocity and calculate $R_0 = \sqrt[3]{\frac{3\,V_0}{4\,\pi}}$ from the volume V_0.

2.3.2 Estimating Time of Droplet Separation from Crown and Cut-off Altitude h_{cut}

Our droplet detection algorithm delivers the time at which a droplet touches the ceiling of the simulation box. The physically relevant quantity, however, is the actual time of separation from the crown splash which we compute from the velocity and time at the moment of detection as described in the SI in section S1.2.

Because our simulations only cover the initial 10 ms of the splashing, only the droplets within this time frame are detected. It is an important observation that the first droplets that separate from the crown are the fastest, and later separating droplets are slower (figure S3 (b)). We thus from our data define a cut-off altitude h_{cut} in figure S3 (b). After the simulated time frame, we expect only insignificant numbers of droplets to be ejected above h_{cut}. This gives us a threshold of confidence: below h_{cut}, almost all droplets are counted whereas above h_{cut} some droplets could turn up after the simulation has finished. In the cumulative distributions of maximum altitude in this work, we mark h_{cut} with a colored asterisk $*$, and we mark the regime of incomplete data, for that the cumulative distribution is considered the lower bound and the upper bound is undefined, as shaded areas.

2.3.3 Trajectory of Spray Droplets: Maximum Altitude, Airborne Time and Pickup by Wind

An impacting raindrop can eject a multitude of small droplets into the air, each of which may contain microplastic particles dispersed in the ocean water. If the microplastic particles are to be transported further into the atmosphere, pickup of these droplets by wind as well as their subsequent evaporation are essential. While we cannot simulate the maximum altitude of ejected droplets directly in our LBM simulation, since this would require very large simulation boxes, we can nevertheless calculate their 3D trajectories based on the initial position and velocity, initial drop radius R_0 at the moment of droplet detection as well as air properties. Although the droplets are tiny in size (with the minimum size resolvable in our simulations being ($\approx 233\,\mu$m)), their velocity is several meters per second, so Reynolds numbers are in the range $20 - 400$ (see figure S4) and simple Stokesian friction does not apply in the first part of the trajectory [90] (see figure S5). We therefore use the more general drag force model [91, p.116][92]

$$\vec{F}_{\mathrm{drag}} = -\frac{1}{2}\,\rho_{\mathrm{a}}\,A\,C_D\,|\vec{u}|\,\vec{u} = -\beta\,C_D\,|\vec{u}|\,\vec{u} \qquad (1)$$

with ρ_{a} being the air density and $A = \pi\,R^2$ being the cross-section of a (spherical) droplet with radius R. $\vec{u} = \vec{u}(t)$ is the 3D velocity of the droplet. The parameter β and the droplet mass m can be written as functions of R:

$$\beta\,(R) = \frac{1}{2}\,\rho_{\mathrm{a}}\,A = \rho_{\mathrm{a}}\,\frac{\pi}{2}\,R^2 \qquad (2)$$

$$m(R) = \rho\,\frac{4\,\pi}{3}\,R^3 \qquad (3)$$

$C_D = C_D(Re(|\vec{u}|, R))$ is the drag coefficient for a viscous droplet with dynamic viscosity contrast as provided by Feng [92] (see section S1.3 in the SI). Re is the Reynolds number

$$Re(|\vec{u}|, R) = \frac{2\,R\,\rho_{\mathrm{a}}\,|\vec{u}|}{\mu_{\mathrm{a}}} \qquad (4)$$

and μ_a is the dynamic viscosity of air. Combining drag and gravity forces, the total force $\vec{F}$ on an airborne droplet is

$$\begin{aligned} m\,\dot{\vec{u}} = \vec{F}(t) &= \vec{F}_g + \vec{F}_\mathrm{drag} \\ &= -(m - m_\mathrm{a})\,g\,\vec{e}_z - \beta\,C_D(Re(|\vec{u}|))\,|\vec{u}|\,\vec{u} \end{aligned} \quad (5)$$

with m_a being the mass of air the droplet displaces (buoyancy) and g being the gravitational acceleration. The equations of motion for an airborne droplet are

$$\begin{aligned} \frac{dx(t)}{dt} &= u_x(t), & \frac{du_x(t)}{dt} &= -k\,C_D\,|\vec{u}|\,u_x, \\ \frac{dy(t)}{dt} &= u_y(t), & \frac{du_y(t)}{dt} &= -k\,C_D\,|\vec{u}|\,u_y, \\ \frac{dz(t)}{dt} &= u_z(t), & \frac{du_z(t)}{dt} &= -k\,C_D\,|\vec{u}|\,u_z - g_r \end{aligned} \quad (6)$$

with

$$g_r = \left(1 - \frac{m_\mathrm{a}}{m}\right) g = \left(1 - \frac{\rho_\mathrm{a}}{\rho}\right) g \quad (7)$$

$$k(R) = \frac{\beta(R)}{m(R)} = \frac{3}{8}\,\frac{\rho_\mathrm{a}}{\rho}\,\frac{1}{R}. \quad (8)$$

On top of this basic trajectory model, we consider two additional effects: (i) evaporation and (ii) updraft. While the droplet is airborne, its radius decreases due to evaporation. To include this effect in our model, we assume [90, eq. (36)]

$$R(t) \approx \sqrt{R_0^2 - \frac{K\,t}{16}} \quad (9)$$

with R_0 being the initial radius and

$$K = q_0\,\Delta T\,(1 + 2\,q_1\,R_0) = 2.628 \cdot 10^{-10}\,\frac{\mathrm{m}^2}{\mathrm{s}} \quad (10)$$

being the evaporation constant [90, eq. (46)]. The parameters $q_0 = 90.63 \cdot 10^{-12}\,\frac{\mathrm{m}^2}{\mathrm{s\,K}}$, $q_1 = 0.004225 \cdot 10^6\,\frac{1}{\mathrm{m}}$ and $\Delta T = 2.9\,\mathrm{K}$ are interpolated from [90, table 5] at $20°C$ and 75% relative humidity, the approximate conditions at the ocean surface [93]. This model is a simplification of the model presented in [90] in that it does not account for the evaporation rate depending on droplet velocity. It directly yields the lifetime [90, eq. (52)]

$$t_\mathrm{life} = \frac{2}{q_0 q_1^2 \Delta T}\,(2q_1R_0 - \log\,(1 + 2q_1R_0))\,. \quad (11)$$

Note that we also ignore solute effects on the evaporation rate in this simplified equation.
In order to include the effect of updraft into our model, an additional vertical wind velocity offset u_updraft is introduced in equation (1)

$$\vec{F}_\mathrm{drag} = -\beta\,C_D\,|\vec{u} - u_\mathrm{updraft}\,\vec{e}_z|\,(\vec{u} - u_\mathrm{updraft}\,\vec{e}_z) \quad (12)$$

with $C_D = C_D(Re(|\vec{u} - u_\mathrm{updraft}\,\vec{e}_z|))$. Assuming constant updraft velocity instead of considering turbulent air movement of course is a simplification and in nature the process is more complicated.
We integrate (6) together with the effects of evaporation and updraft numerically using Runge-Kutta-4 with a stepsize of $\Delta t = 0.1\,\mathrm{ms}$ (without updraft) $\Delta t = 10\,\mathrm{ms}$ (with updraft) for each one of the approximately 17000 spray droplets per 100-impact-dataset. As initial conditions, we use the position and velocity from the droplet detection algorithm in section 2.3.1. From the integration we obtain the maximum altitude and airborne time. From the airborne time we derive a clear criterion of microplastic uptake into the atmosphere. Two uptake scenarios are possible: either the airborne time diverges due to high updraft in combination with small droplet size or the airborne time remains finite (i.e. the droplet would eventually fall back onto the surface) but larger than the droplet life time. In this latter case, we also consider the microplastic particles to be taken up by the atmosphere.
For our setup (sea water and air, temperature $T = 20°C$) we have $g = 9.81\,\frac{\mathrm{m}}{\mathrm{s}^2}$, $\rho = 1024.8103\,\frac{\mathrm{kg}}{\mathrm{m}^3}$, $\rho_\mathrm{a} = 1.204\,\frac{\mathrm{kg}}{\mathrm{m}^3}$ [94] and $\mu_\mathrm{a} = 1.813 \cdot 10^{-5}\,\frac{\mathrm{kg}}{\mathrm{m\,s}}$ [95].

2.4 Simulation Setup and Parameters

Raindrops in nature are limited in diameter d to approximately $1 - 7\,\mathrm{mm}$. If they are too small, they are mainly advected by winds, and if they are too large, they split into smaller droplets [33]. Depending on diameter and local air pressure, the terminal velocity (table 1) is fixed [29, 30, 32].
For simulating sea water, we use the kinematic shear viscosity $\nu = 1.0508 \cdot 10^{-6}\,\frac{\mathrm{m}^2}{\mathrm{s}}$, density $\rho = 1024.8103\,\frac{\mathrm{kg}}{\mathrm{m}^3}$ and surface tension $\sigma = 7.381 \cdot 10^{-2}\,\frac{\mathrm{kg}}{\mathrm{s}^2}$ at standard temperature $T = 20°C$ and standard absolute salinity $S = 35\,\frac{\mathrm{g}}{\mathrm{kg}}$ [96, 97]. The standard gravitational acceleration is $g = 9.81\,\frac{\mathrm{m}}{\mathrm{s}^2}$. The raindrop fluid possesses the same parameters as sea water.
The depth of the liquid pool must be sufficient to approximate a 'deep' pool in that, during impact, the cavity expansion is not limited by the bottom wall of the simulation box. For a box deeper than a few times the drop diameter, there is no significant change in impact dynamics. In the simulation, the overall box size is limited by (video) memory, so a larger box size compared to the drop size will lower the resolution. We use a box of the size $10\,d$ by $10\,d$ by $8.5\,d$ with a pool depth of $4\,d$ as a compromise. The lateral boundaries are periodic and the very bottom lattice layer is a no-slip bounce-back boundary.
The microplastic particles are initialized at (reproducible) pseudo-random positions in the liquid pool with a concentration of 5000 particles per cubic cen-

d/mm	$u/\frac{\mathrm{m}}{\mathrm{s}}$	Re	We	Fr	Ca	Bo
1	4.50	4282	281	45.4	0.0657	0.136
2	6.80	12943	1284	48.6	0.0992	0.545
3	8.10	23125	2733	47.2	0.1182	1.226
4	8.80	33498	4301	44.4	0.1284	2.179
5	9.20	43776	5876	41.5	0.1342	3.405
6	9.40	53673	7361	38.8	0.1371	4.903
7	9.55	63618	8864	36.4	0.1393	6.674

Table 1: Terminal velocity curve for raindrops at mean sea level pressure [29]. The terminal velocity u is a function of the drop diameter d. The dimensionless numbers Reynolds $Re = \frac{d\,u}{\nu}$, Weber $We = \frac{d\,u^2\,\rho}{\sigma}$, Froude $Fr = \frac{u}{\sqrt{d\,g}}$, Capillary $Ca = \frac{u\,\rho\,\nu}{\sigma}$ and Bond $Bo = \frac{d^2\,\rho\,g}{\sigma}$ are also given.

timeter – several orders of magnitude higher than typically found in nature ($\approx 1-7$ particles per liter) [1]. If we chose typical particle concentration as found in contaminated sea water, there would be a single or no particle at all in the volume of the simulated domain. A particle concentration much higher would severely decrease computational efficiency as the buoyancy force of multiple nearby IBM particles would have to be atomic-added to each lattice point. Since our particles do not directly interact with each other and buoyancy effects are negligible at short time scales, the number of ejected particles per raindrop impact can be scaled down linearly with the initial particle concentration. The raindrop initially is devoid of particles.

During unit conversion from SI-units to lattice units, the density $\rho = 1$ in lattice units is fixed and the length scale L_x is limited by memory capacity. The impact velocity u in lattice units however is a free parameter that does not change physics, but has large impact on both compute time (proportional to $\frac{1}{u}$) and accuracy. Tests with Poiseuille flow in a cylindrical channel showed that for FP32 floating-point accuracy, u should be in the range $0.0003 \leq u \leq 0.5$ and especially $u \leq 0.5$ anywhere in the simulation box [51]. In figure S12 in S3.1 we show that simulations with $u \in [0.005, 0.15]$ run stable. Outside of this interval, instabilities propagate from the first unstable lattice point through the simulation box. We thus use $u = 0.05$ for all of our simulations. We also show simulations for varying lattice resolution $L_x \in \{64, 128, 256, 464, 636, 748\}$ in S3.2 and observe that $L_x < 400$ is insufficient to resolve details.

[1] Measurements of microplastic concentration on sea surface water greatly vary with time, location and measurement method, ranging from 0.0018 [20] over 0.04 [21] to 0.406 [22] microplastic particles per m^2 in trawl measurements. [54] provide an average volumetric value of 2.9 particles per liter at 5 m depth and [3] provide measurements of between 0.99–7.00 particles per liter across the Atlantic Ocean at 10 m depth.

Simulations are run for a time span of 10 ms. In nature, around 10 ms after impact low air pressure behind the falling droplet contracts the crown to a canopy or surface seal [46]. Since in our simulation we do not model the gas phase, we do not observe surface seal formation and thus for longer simulation periods our results would significantly divert from experimental findings. However only in the initial splash phase of the impact do ejected droplets have sufficiently small size and high velocity to be relevant for atmospheric pickup. Droplets ejected later are slower and more likely to fall back to the surface instead of contributing to transport across the interface. The later occurring jet in the experiment also does not significantly contribute to ejected high-velocity droplets [41, 46].

3 Simulation Results

3.1 Validating our Simulations by Comparison with Experiment

To begin our investigations, we validate the employed simulation model by comparing it to high-speed images of a raindrop impact in sea water. Further validation is provided in S2. Murphy et al. [46] studied oily marine aerosol production when raindrops impact oil slicks on the ocean. As a control, they conducted and documented a 4.1 mm diameter raindrop impact on pure seawater at an impact velocity of 7.2 $\frac{\mathrm{m}}{\mathrm{s}}$ – a bit less than terminal velocity. The fluid properties used in [46] (see table S2 of the SI) coincide almost exactly with the ones used in our microplastic simulations as detailed below. This allows for a direct experimental validation of our simulation setup in the relevant parameter range. In figure 1, we compare the simulation results to the experiment finding excellent agreement for the cavity size and crown breakup. A deviation between experiment and simulation is only seen at $t = 8$ ms where the crown in the experiment contracts and begins to form a canopy. This is due to the lower air pressure behind the falling raindrop (Bernoulli effect) that pulls the crown inwards in what is commonly called a surface seal [98, 99]. Since air flow is not explicitly included in our simulations, we do not see the crown contracting. In this work, however, we focus on the generation and ejection of small droplets which primarily happens in the beginning of splash formation and thus will not be significantly affected by the entraining air flow.

	SI-units	LBM units
drop diameter d	$1-7\,\mathrm{mm}$	46.4
impact velocity u	$4.50-9.55\,\frac{\mathrm{m}}{s}$	0.05 (fixed)
impact angle α	$0°-40°$	$0°-40°$
simulation box dimensions L_x, L_y, L_z	$10\,d$, $10\,d$, $8.5\,d$	464, 464, 394 (fixed)
pool height h	$4\,d$	185.6
kinematic shear viscosity ν	$1.0508\cdot10^{-6}\,\frac{\mathrm{m}^2}{\mathrm{s}}$	$5.417\cdot10^{-4}\,(1\,\mathrm{mm})-3.647\cdot10^{-5}\,(7\,\mathrm{mm})$
water density ρ	$1024.8103\,\frac{\mathrm{kg}}{\mathrm{m}^3}$	1 (fixed)
surface tension σ	$7.381\cdot10^{-2}\,\frac{\mathrm{kg}}{\mathrm{s}^2}$	$4.126\cdot10^{-4}\,(1\,\mathrm{mm})-1.309\cdot10^{-5}\,(7\,\mathrm{mm})$
gravitational acceleration g	$9.81\,\frac{\mathrm{m}}{\mathrm{s}^2}$	$2.610\cdot10^{-8}\,(1\,\mathrm{mm})-4.057\cdot10^{-8}\,(7\,\mathrm{mm})$
(hydrodynamic) particle diameter d_p	$\lessapprox 22-151\,\mu\mathrm{m}$	$\approx 0.6-1.0$ (fixed)
number of particles N_p	$2000-686000$	$2000-686000$

Table 2: Overview on the simulation parameters before and after unit conversion.

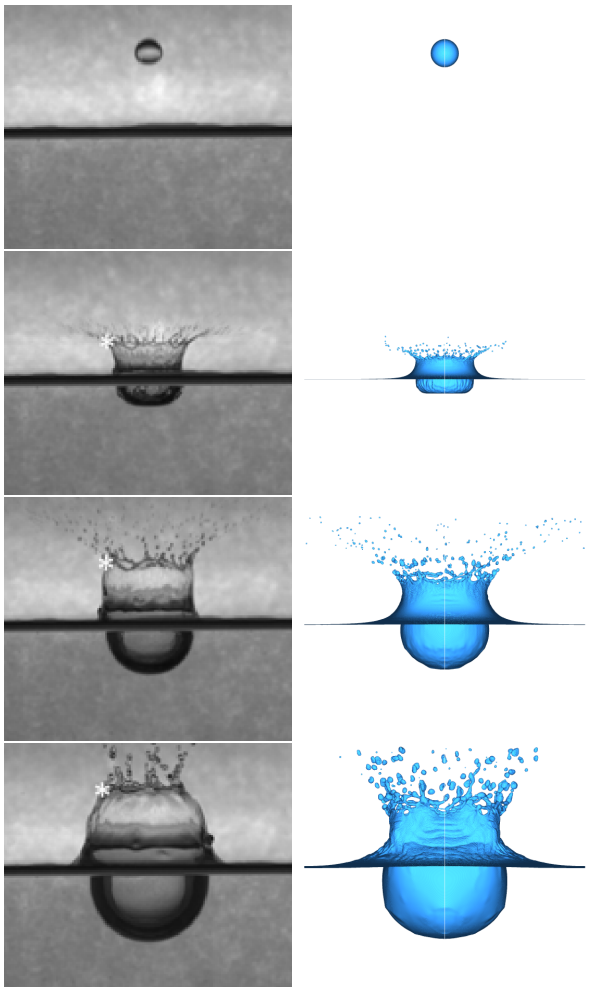
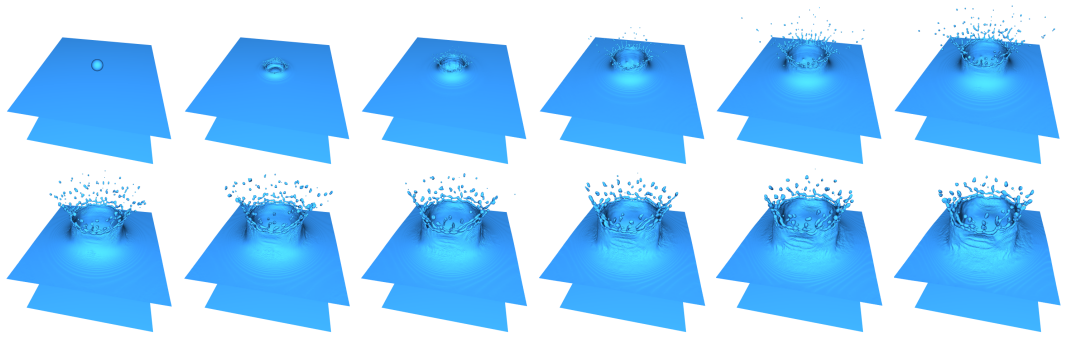


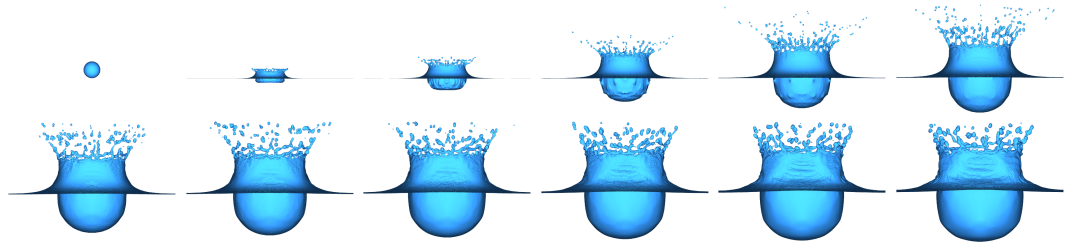
Figure 1: Our simulation compared to the experiment from Murphy et al. [46], figure 3 (a)-(d), at times $t \in \{-2,\ 1,\ 3,\ 8\}$ ms. In both experiment and simulation, $t = 0$ ms is defined as the time the droplet touches the water surface. The raindrop diameter is 4.1 mm. The asterisk marks the tracked crown rim position in the experiment.

3.2 Simulation of Raindrop Impacts on Pure Sea Water

We next illustrate our 4 mm diameter raindrop impact reference system in figure 2 for pure sea water without microplastic particles. The corresponding parameters are given in table 2. Shortly after impact, the perimeter of the impact site shoots upward. It forms a thin wall of fluid – called crown – that quickly breaks up into lots of tiny droplets due to surface tension. Crown droplets initially are small and fast with a velocity inclined by approximately 53° from the vertical axis radially outward. As time progresses, they gradually become larger and slower (with less velocity inclination). Shortly after the raindrop and pool surfaces touch (time of impact), the upper half of the raindrop initially retains its convex shape which now forms the bottom surface of the impact cavity while displaced fluid exits upward at the perimeter. Once the raindrop fluid plunges further in – about 1 ms after impact – the cavity center flattens and becomes concave thereafter, steadily expanding while more and more fluid is pushed into the crown. The fluid of the raindrop spreads into a thin sheet around the cavity. During the simulated 10 ms, the cavity and crown continuously expand while the gravitation-driven cavity collapse followed by the well-known vertical fluid jet [41, 46, 100] happen only at a later stage. The fastest droplets that reach the highest altitude in the air are ejected within the first few milliseconds after impact. Previous research has suggested that the secondary jet after cavity collapse does not contribute greatly to ejection of droplets [41, 46] so for this study we have not included it.



(a) Perspective view.



(b) Orthogonal side view.

Figure 2: Visual representation of the 4 mm diameter raindrop impact simulation without microplastic particles. Time stamps (left to right) are $t \in \{0.0, 0.5, 1.0, 2.0, 3.0, ..., 10.0\}$ ms and lattice resolution is $L_x = 464$.

3.3 Impacts of Raindrops on Sea Water with Microplastic Particles

We now include microplastic particles in the simulation. To obtain sufficient statistics of ejected droplets and particles, we run the simulation 100 times for each set of parameters, with the microplastic particles each time being initialized at different random positions, resulting in slightly different random crown breakup. We choose the highest possible lattice resolution for our IBM-LBM simulations ($L_x = 464$) and run each simulation for 10 ms. Droplets that touch the top of the simulation box are measured and then deleted from the simulation box as described in section 2.3.1 above. The upper bound for the microplastic diameter is 86 μm as determined in S1.1 and their density is 1.05 $\frac{\text{g}}{\text{cm}^3}$ (polystyrene). In figure S23 we provide data on the velocity inclination from the vertical axis of the ejected droplets for the 4 mm diameter raindrop reference data set. In S4.7 we provide simulations over the entire range of common plastics densities from 0.92 $\frac{\text{g}}{\text{cm}^3}$ (polypropylene) to 2.17 $\frac{\text{g}}{\text{cm}^3}$ (polytetrafluoroethylene) and demonstrate that buoyancy effects are negligible during the simulated time frame. Figure S27 demonstrates how small the run-to-run variation is in the simulation results, giving us confidence about the accuracy of our data. In section S3.2 we provide data on the influence of lattice resolution on the droplet distribution.

3.3.1 Microplastic Transport during 4 mm Diameter Raindrop Impacts

Figure 3 shows how the ejected droplets of 100 raindrop impact simulations are distributed in diameter (x-axis), ejection altitude (y-axis) and time of detection (color). We observe that droplets at later points in time have more volume, but less velocity, so maximum altitude is lower. After $t = 5$ ms almost no droplets are ejected beyond half a meter in amplitude. This gives us confidence that our simulations capture all droplets with a high ejection velocity which are relevant to microplastic transport. The vertical velocity of the small droplets ejected at an early stage reaches up to about 10 $\frac{\text{m}}{\text{s}}$, while the velocities of larger droplets ejected at later times are below 3 $\frac{\text{m}}{\text{s}}$. In figures 4 and 5, we present histograms to reveal the size distribution of droplets and the cumulative ejected fluid volume as well as the distribution of how many particles are present in droplets and the particle concentration depending on droplet diameter. The bin width in all histograms is 0.02 mm.

We observe that, by number, the majority of ejected droplets has a diameter below 0.6 mm with the peak centered at 0.4 mm. In reality, there are likely to be more smaller droplets because we cannot resolve droplets below 0.23 mm diameter in our simulation. Between 0.6 and 1.0 mm, we observe a relatively flat plateau. These larger droplets are less numerous, but contribute more than double to cumulative

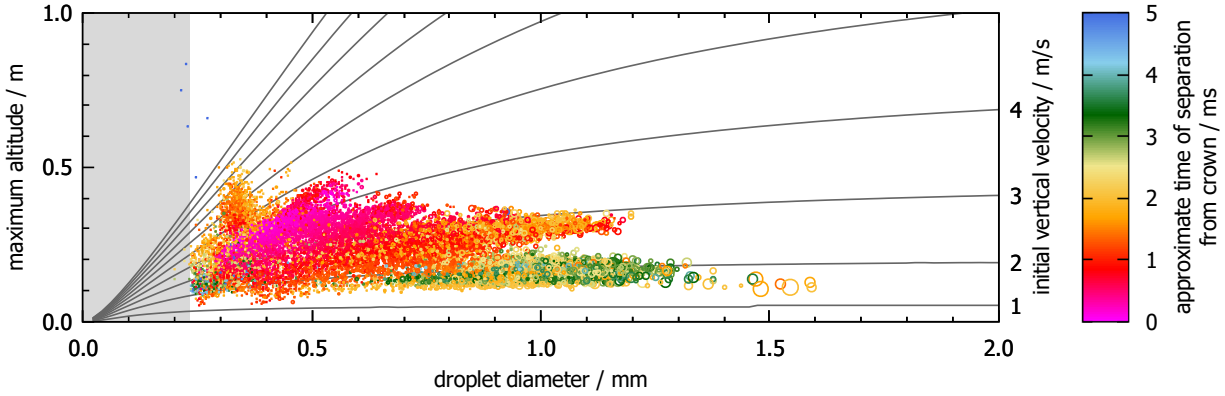


Figure 3: Diameter and maximum altitude of droplets detected within 10 ms after impact of 100 4 mm diameter raindrops. The size of the circles indicates the number of microplastic particles enclosed in each droplet. The lines indicate constant initial vertical velocity. Tiniest droplets below 0.23 mm diameter (gray area) cannot be resolved in the simulation. The color represents the approximate time the droplet separated from the crown rim. The maximum altitude of ejected droplets decreases over time.

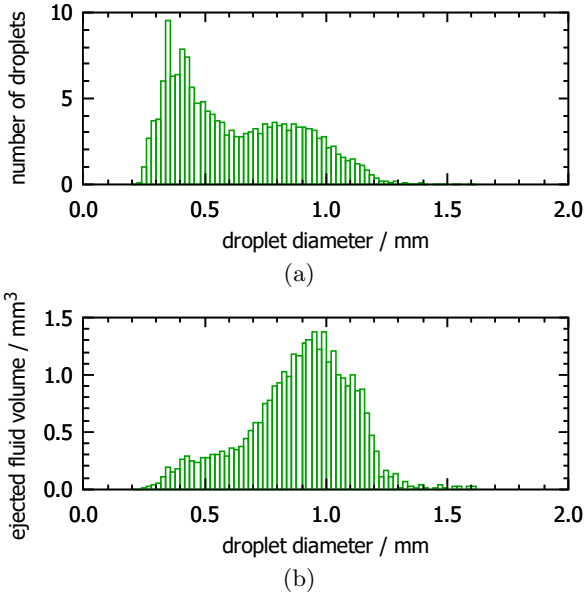


Figure 4: Properties of ejected droplets: (a) The size distribution and (b) the distribution of ejected fluid volume by droplet diameter.

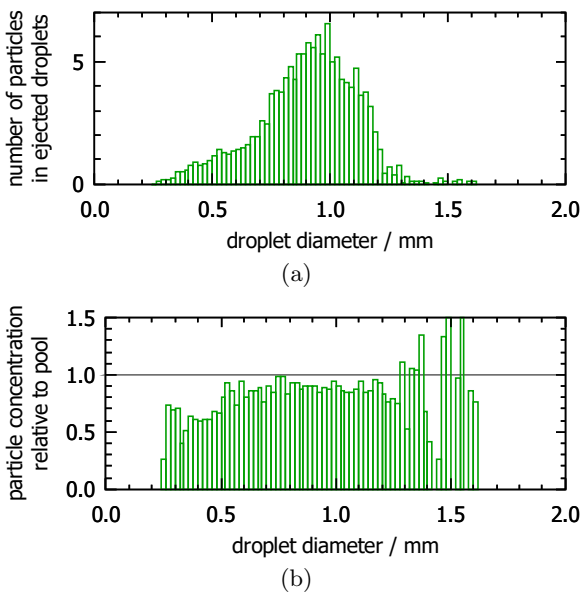


Figure 5: Properties of microplastic particles in ejected droplets: (a) The distribution of particles in droplets and (b) particle concentration depending on droplet diameter.

ejected fluid volume than the small droplets as can be seen in figure 4 (b). Comparing figures 4 (b) and 5 (a), we find good proportionality between cumulative ejected fluid volume and the number of particles in the ejected droplets. Larger droplets with more fluid carry more particles. The particle concentration in ejected droplets is on average 87% compared to the initial concentration in the pool, meaning that the raindrop fluid makes up approximately 13% of the ejected fluid. Only for the smallest droplets, the particle concentration appears to be a bit lower, closer to 70% which most likely is an effect of numerical resolution and a larger uncertainty, as in this region there is very little total fluid volume to divide the particle number by.

To conclude this section, we find that a single 4 mm diameter raindrop ejects about 167 droplets (> 0.23 mm diameter[2]) during the first 10 ms after impact. These droplets contain a total of 136 microplastic particles for an initial concentration of 5000 microplastic particles per cm^3 in the sea water.

[2]smaller droplets cannot be resolved in the simulation

3.3.2 How Raindrop Diameter affects Microplastics Transport

The diameter of raindrops, which determines their terminal velocity according to table 1, strongly affects the number of ejected droplets and microplastic particles. Figure 6 shows the distribution of size, altitude and microplastic load of ejected droplets similar to our 4mm reference system in figure 3, when the raindrop diameter varies between 2 and 7 mm. Snapshots of the corresponding simulations are shown in figure S28. From these visualizations we conclude that raindrops with diameters below 2 mm, although frequently occuring in nature, do not eject a significant number of droplets into the air. Raindrops with diameters above 7 mm are very rare in natural rain events and are therefore not considered further. Note that the minimum resolvable droplet size is larger for simulations of larger raindrops and thus the gray area in figure 6 where no droplets can be resolved, increases.

For a more quantitative description, figure 7 (a) provides histograms of the droplet size distribution for different raindrop diameters. We observe that with increasing raindrop diameter, the average size of produced droplets increases simultaneously. Figure 7 (b) shows a similar trend for the number of microplastic particles ejected by droplets of various sizes: for larger raindrops the contribution of the larger droplets increases. We furthermore note that similar to the 4 mm reference case of the previous section, within the droplets we find a constant particle concentration of on average 85% the bulk value regardless of raindrop diameter (data not shown).

In the SI in figure S3 we provide the maximum altitude of ejected droplets as function of the time the droplets separate from the crown rim. We see that – despite the impact velocity being larger for bigger raindrops – the overall time scale of the impact increases with droplet size. Figure S3 furthermore shows that the maximum altitude of the ejected droplets decreases monotonically over time. This illustrates an important limitation of our simulations: since we only consider the first 10 ms after impact, droplets after this time are missing in our statistics. For small raindrops with sizes up to 4 mm, figure S3 shows that the amount of missed droplets can be expected to be small and, more importantly, the maximum altitude of the missing droplets will not be larger than a cut-off of around $h_{\mathrm{cut}} = 0.19\,\mathrm{m}$. This picture changes for raindrops with diameters above 4 mm, where a substantial amount of droplets are missed. Nevertheless, figure S3 allows us to determine h_{cut} depending on raindrop size. For the largest raindrops of 7 mm, $h_{\mathrm{cut}} = 0.48\,\mathrm{m}$, meaning that the vast majority of droplets ejected to altitudes above

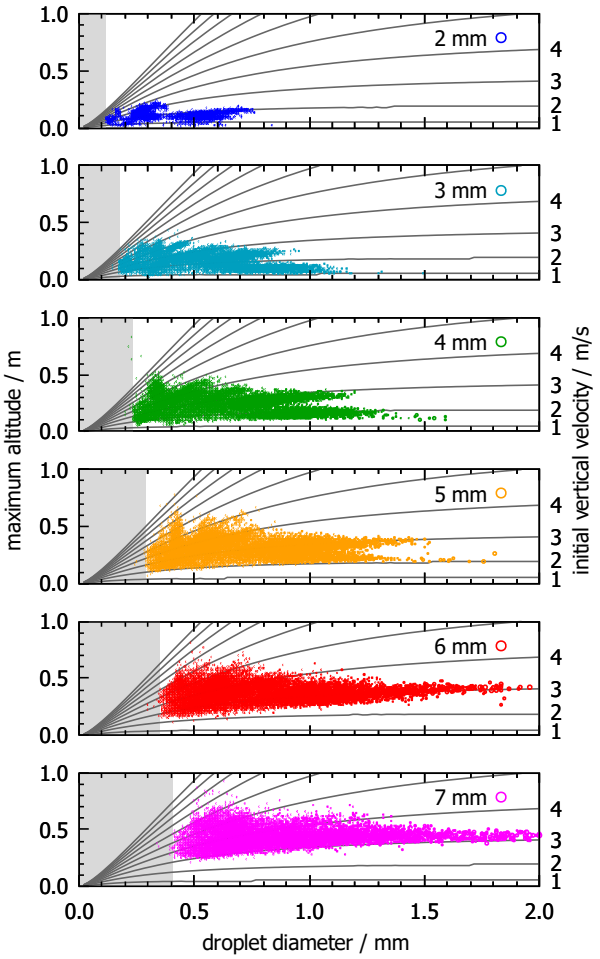


Figure 6: Diameter and maximum altitude of ejected droplets after impact of terminal velocity raindrops of various diameters. The lines indicate constant initial vertical velocity. The circle size indicates the number of microplastic particles in each droplet. Tiniest droplets in the gray marked areas on the left cannot be resolved in simulation.

0.48 m are counted, and some droplets with ejection altitudes below 0.48 m are missed.
In figure 8 we show the number of ejected droplets and number of particles in ejected droplets as a function of maximum altitude. The droplets from the initial phase of the impact are very small and therefore carry less particles than droplets with larger volume from the later phase of the impact, so the cumulative distribution of maximum altitude of the particles drops steeper than that of the droplets.

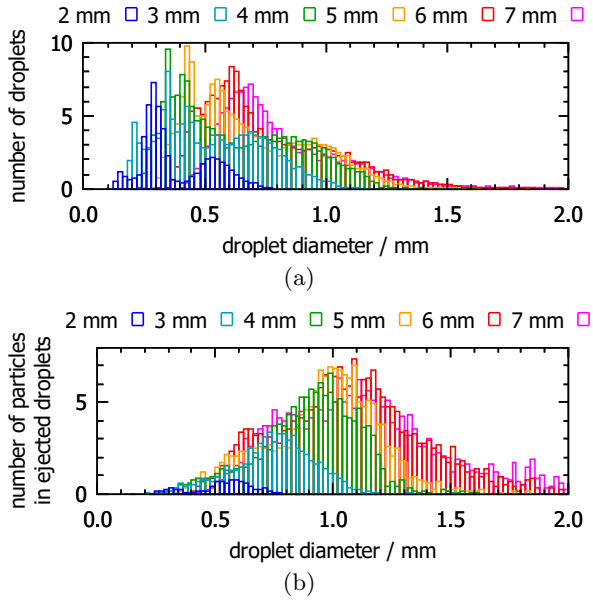


Figure 7: (a) Size distribution of ejected droplets and (b) distribution of particles in ejected droplets depending on droplet size for various raindrop diameters. In the SI, we show these histograms as separate plots for each raindrop diameter in figures S21 and S22.

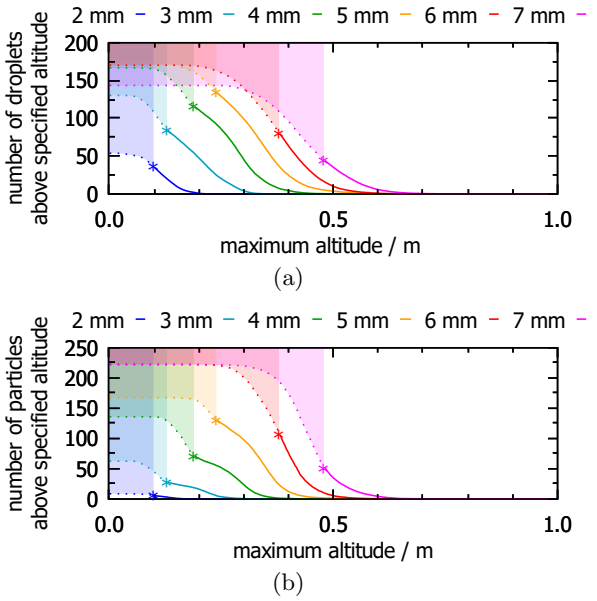


Figure 8: Cumulative distribution of maximum altitude of resolvable droplets (a) and microplastic particles in droplets (b). The graphs indicate the number of droplets / particles ejected above a specified altitude depending on raindrop diameter. Above the cut-off altitude h_{cut} marked by the asterisk, all droplets are detected by our simulations (solid line). Below h_{cut}, our simulations only provide a lower bound for the number ejected droplets (dashed line) due to limited simulation time as explained in section 2.3.2.

3.3.3 How Oblique Impacts affect Microplastics Transport

Due to influence of wind, it may be expected that most raindrops will not fall perfectly straight and will impact the water surface at an angle. Such oblique impacts have an inclined and asymmetric crown geometry as illustrated in figure 9 and in figures S29 and S30 in the SI. This affects how the fluid of the raindrop, which is initially devoid of microplastics, is distributed in the bulk fluid during impact. For the straight impact, the raindrop water is spread around the bottom of the cavity as a thin film and not ejected into the air as demonstrated by the high microparticle concentration in the droplets in figure 5 (b). This behavior changes for oblique impacts. As shown in figure 10 (a), here the raindrop fluid is partly redirected into the crown, thus slightly reducing the overall particle concentration in the ejected droplets from 87% of the pool concentration ($\alpha = 0°$) to 77% ($\alpha = 40°$).

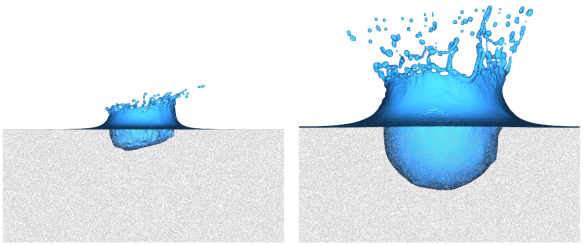


Figure 9: Illustration of a $\alpha = 20°$ inclined raindrop impact at times $t \in \{1, 5\}$ ms.

Besides this, there is a second, equally interesting effect emerging: For the straight impact, the fast droplets directly after impact are ejected radially outward with their velocity inclined from the vertical axis as can be seen in the first frames of figure 2. Thus, on one side the trajectory of the small droplets becomes more vertical, resulting in more droplets being ejected to higher altitude as shown in figure 10 (b). In addition, the total amount of ejected droplets during our simulated 10 ms time frame also increases. For this reason, the initial fast droplets carry microplastic particles to higher altitudes for oblique impacts. Nevertheless, since the water from the raindrop itself is partly deflected into the crown later on, the total amount of ejected particles is actually smaller for oblique impacts as shown in figure 10 (c).

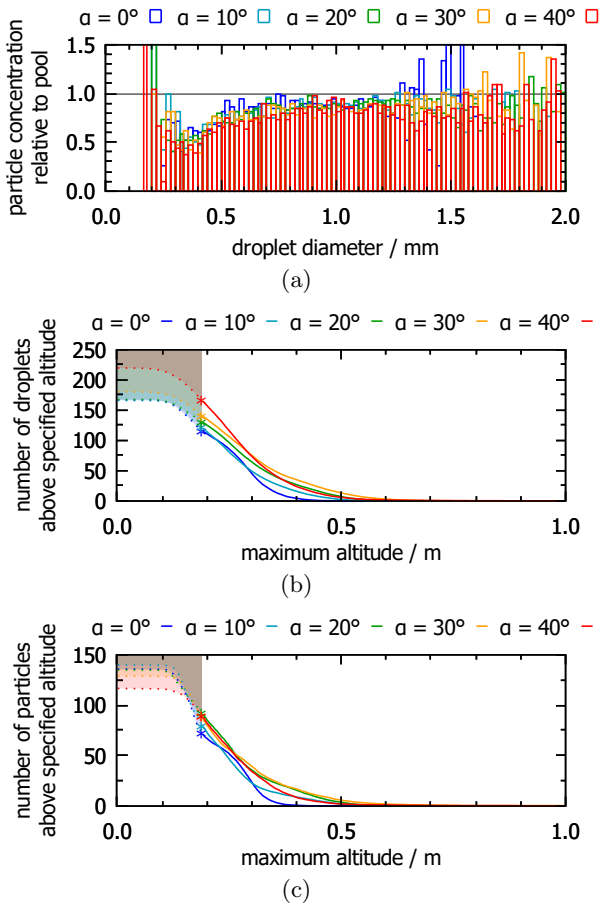


Figure 10: Simulation results for oblique impacts: (a) Distribution of particle concentration in ejected droplets depending on droplet size and impact angle α as well as (b) number of resolvable droplets and (c) particles in droplets ejected above a specified altitude depending on impact angle α. In (b) and (c), for altitudes lower than h_{cut} (asterisks), the distributions are considered the lower bound (dashed lines) and the upper bound is undefined as indicated by the shaded areas (see section 2.3.2).

3.3.4 Origin of ejected Microplastic Particles

Using our simulations, we can also detect from which region of the bulk fluid (relative to the impact location) the microplastic particles originate that are ejected into the air. For this, we assign a unique ID number to each microplastic particle in the bulk fluid and then store the IDs of those particles that are ejected into the air during the impact simulation of 10 ms. The initial position of those particles is marked in red in figure 11. We find that only particles from the very top layer of the pool are ejected in the crown. The initial positions furthermore form a ring around the impact site.

Figures S28 and S30 illustrate particle origin for various raindrop diameters and impact angles. In all cases only particles directly on the water surface or a few millimeters below the surface are relevant for consideration.

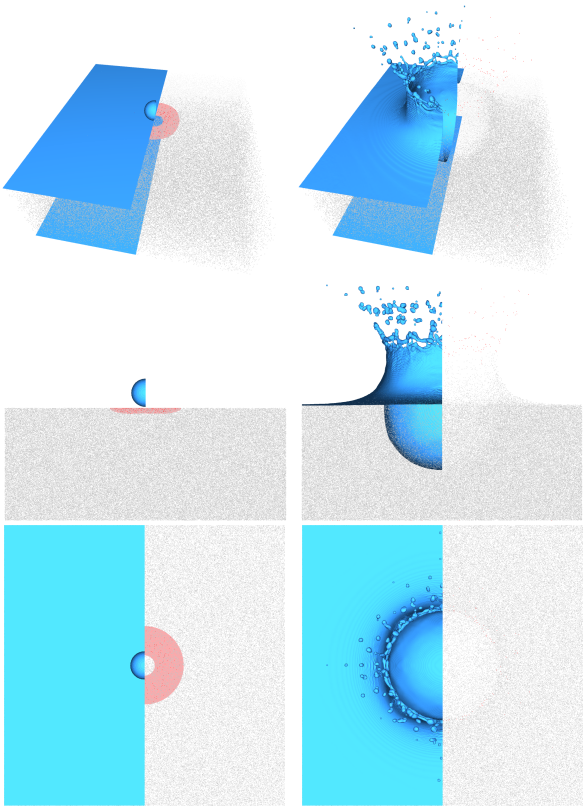


Figure 11: Illustration of the origin of ejected particles. All particles in ejected droplets by $t = 10$ ms are colored in red. At $t = 0$ ms (left column) these particles are located on a ring around the impact site directly under the surface. They are the first to enter the crown and to get into separated droplets during crown breakup (right column at $t = 5$ ms).

4 Experimental Demonstration of Microplastic Transport

In addition to the detailed simulations, we conduct a series of laboratory experiments demonstrating microplastic transport by impacting drops. An image of our experimental setup is shown in figure 12 (a). The setup consists of a timer-controlled magnetic valve (eltima electronic) which allows us to adjust the size of the falling drop by changing the opening time of the valve. A Mariotte's bottle containing desalinated water is connected to the valve with a nozzle below. The surface tension of desalinated water ($72.75\,\frac{\mathrm{mN}}{\mathrm{m}}$) only insignificantly differs from that of sea water ($73.39 - 76.67\,\frac{\mathrm{mN}}{\mathrm{m}}$ depending on salinity) at $20°C$. We note that natural surfactants in the ocean surface microlayer, enriched by for example bubble scavenging [24], may slightly decrease surface tension compared to pure salt water. In our experiments, microplastic particles could, in principle, reduce surface tension in a similar fashion. However, the volumetric ratio of microplastics to water is $\approx 10^{-6}$ and thus small enough that this effect can safely be neglected regarding both viscosity[3] and surface tension. The drop falls into a water reservoir which is filled with desalinated water and spherical polystyrene particles with a diameter of $6\,\mu$m (Polysciences). The particle concentration in the reservoir is $25000\,\frac{1}{\mathrm{cm}^3}$. When opening the magnetic valve for 20 ms, we obtain falling drops with a diameter of approximately $5.0 - 5.5$ mm. Similar to the simulation, the water height in the reservoir is about 4 times the drop diameter. The distance between the water body and the nozzle exit is approximately 2.5 m, based on which we estimate the impact velocity to be $6.8\,\frac{\mathrm{m}}{\mathrm{s}}$ [49], i.e. about 74% of terminal velocity [29]. The splash droplets released after impact are caught on a 76 mm × 26 mm glass slide mounted 20 cm above the reservoir. The glass slide is placed such that the inner 26 mm edge is 65 mm (position A) and 141 mm (position B) radially outward from the impact center (figure 12 (c)).

After the impact, the glass slide is transferred to a microscope immediately. Although the glass slides are carried openly to the microscope located in the same room and thus a small contamination by particles from room air is possible, it would be highly unlikely that contaminant particles coincidentally possess the same uniform diameter ($6\,\mu$m) as our microplastic particles. The size and spherical shape of the counted particles thus clearly shows that the observed particles on the glass side are indeed microplastic ejected from the liquid reservoir. Independent of the rain drop experiments we control the shape and size of our polystyrene particles by attenuating the suspension containing the particles and assessing them under the microscope to confirm the spherical shape and the diameter of $6\,\mu$m.

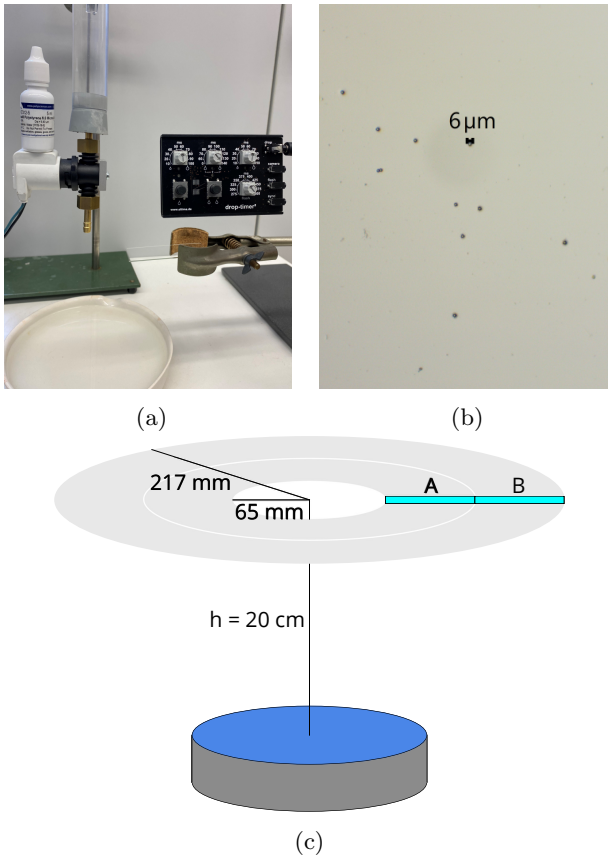


Figure 12: (a) The experimental setup with timer-controlled magnetic valve for the creation of individual drops, connected to a Mariotte's bottle above and a nozzle below. Released drops impact on a reservoir filled with desalinated water and $6\,\mu$m spherical polystyrene particles. Ejected droplets are captured on a glass slide. Note that for illustration purposes, valve and reservoir are shown together, while in our experiments the falling distance is 2.5 m. (b) Splash droplets carry particles to a glass slide placed above the reservoir. The droplets evaporate, leaving the microplastic particles behind. (c) Sketch of the glass slide placement (positions A and B) above the reservoir. The radial symmetry of the impact splash allows to only cover a circle segment with the glass slides and extrapolate to the annular cross-section areas around the A and B placements (gray rings) through which ejected droplets may pass.

Using the microscope we clearly observe the presence of microplastic particles in the splash droplets (figure 12 (b)). For positions A and B we conduct 10 drop impacts each. We then count the number of droplets

[3] Einstein suspension viscosity [101, 102] is 0.0007% larger than without particles.

on the glass slide that have not yet evaporated, measure their approximate diameter, their position on the slide and the number of particles inside. The number of tested splash drops depends on the number of drops found on the glass slide and how many of the drop positions could be clearly identified under the microscope. Some of them evaporated before their location on the slide was noted down. Some of the ejected droplets do not contain any plastic particles. In some cases, we could test all droplets for microplastic particles if there were only between 1 and 3 droplets on the slide.

We find that for a single drop impact, the average number of droplets on the glass slide at position A is 4.2 and on position B is 2.4. The average number of particles per detected droplet is 16 and 3 for positions A and B, respectively. The difference in particles per droplet for positions A and B is due to the smaller average size of the droplets at B. This is in full agreement with the numerical simulations: as can be seen in figure S23, the initially released small droplets have a higher velocity inclination from the vertical axis and thus are more likely to land on the glass slide at position B, whereas larger droplets released at later times have a smaller velocity inclination and are more likely to land at position A.

In order to obtain the total number of ejected droplets by a single drop impact, we first extrapolate the area of the glass slides at A and B to the two cross-sectional rings around positions A and B (figure 12 (c)). In addition we re-scale the particle concentration to 5000 $\frac{1}{\mathrm{cm}^3}$ to be able to directly compare with the simulation. Adding the two cross-sectional rings A and B, we find that 208 droplets containing 398 particles are ejected to an altitude of 20 cm or higher. The number of droplets is in very good agreement with the simulations (approximately 1.3 times larger) as can be seen in figure 8 (a) when looking at the lower bound of the shaded area for the 5 mm curve at 0.2 m altitude. The number of particles, on the other hand, is approximately 2.5 times larger in the experiment than in our simulations (figure 8 (b), lower bound of the shaded area for the 5 mm curve at 0.2 m altitude). This points to a certain surface-activity of the employed microplastic particles: particles sticking to the reservoir surface would enrich the local concentration at the surface, thus resulting in a higher particle concentration within the splash droplets as well.

5 Discussion: Estimating the Annual Amount of Microplastics Transitioning from Global Oceans into the Atmosphere due to Impacting Raindrops

Based on our simulation results, we can provide an order-of-magnitude estimate for the global annual amount of microplastics transitioning from the oceans into the atmosphere due to raindrops. To guarantee reproducibility, our model assumes single, isolated raindrop impacts on a perfectly flat ocean surface. For this approximation, we use straight impacts only. We are aware that this represents an idealized scenario compared to what would be observed in field experiments. Given the robustness of our results even for oblique impacts as shown in figure 10, we are confident that this does not represent a major limitation.

We will proceed in three steps: in the first step, we obtain the number of raindrops as function of their size depending on the rain rate based on known experimental data. In the second step, using our simulations, we can predict the number of microplastic particles ejected into the air per square kilometer per hour, again depending on the rain fall rate and wind speed. In the final step, we provide an estimate of the global annual amount of microplastic transported into the atmosphere due to impacting raindrops.

5.1 How many raindrops of which size are present in nature?

As shown in section 3.3.2, the amount of transported microplastic strongly depends on the diameter of the impacting raindrop. As a first step, it is therefore necessary to understand how the total amount of precipitation water is distributed across different raindrop sizes for different rain rates. We approximate the distribution by the Marshall-Palmer model [31, 33]: By number, small raindrops are exponentially more frequent than large raindrops, following

$$N(d) = \Lambda\, e^{-\Lambda\, d} \tag{13}$$

whereby $\Lambda = 4.1\,(\frac{\mathcal{R}}{\frac{\mathrm{mm}}{\mathrm{h}}})^{-0.21}\,\frac{1}{\mathrm{mm}}$ is a parameter depending on the rainfall rate $\mathcal{R}$ in $\frac{\mathrm{mm}}{\mathrm{h}}$ and d is the raindrop diameter in mm. Eq. (13) is normalized. In order to reasonably estimate how the total precipitation volume splits up upon differently sized raindrops, we discretize the raindrop size distribution (13) to discrete raindrop diameter ranges. The probability to find a raindrop in the interval $[d-0.5\,\mathrm{mm},\ d+0.5\,\mathrm{mm}]$

then is:

$$p(d) = \int_{d-0.5\,\mathrm{mm}}^{d+0.5\,\mathrm{mm}} N(d')\,\mathrm{d}d' \tag{14}$$

$$= \left[-e^{-\Lambda\, d'}\right]_{d-0.5\,\mathrm{mm}}^{d+0.5\,\mathrm{mm}} \tag{15}$$

$$= 2\, e^{-\Lambda\, d} \sinh\left(\frac{\Lambda}{2}\,\mathrm{mm}\right) \tag{16}$$

Using $p(d)$, we estimate how a given precipitation volume V_{pre} splits up upon differently sized raindrops with $d \in \{1, 2, 3, ..., 7\}$ mm. The volume of a single raindrop is denoted as $V(d) = \frac{\pi}{6}\, d^3$. Computing the products $p(d) \cdot V(d)$ and normalizing results yields the volume fractions associated with the raindrop diameters[4]:

$$p_V(d) \approx \frac{p(d) \cdot V(d)}{\sum_{d=1,2,...,7\,\mathrm{mm}} p(d)\, V(d)} \tag{17}$$

$p_V(d)$ is plotted in figure 13 and used to calculate

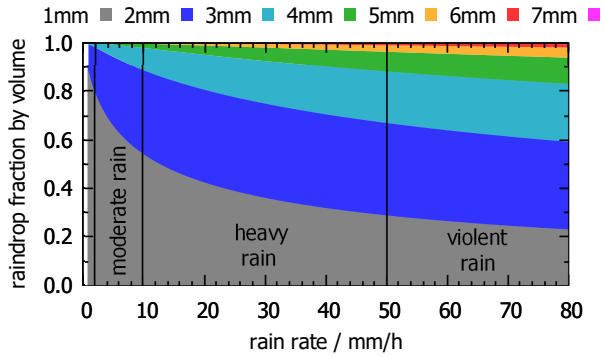


Figure 13: The (normalized) raindrop size distribution by volume, depending on the rain rate $\mathcal{R}$. For moderate rain, 1 mm diameter raindrops make up the largest part while 2 mm diameter raindrops dominate for heavier rain. $3-5$ mm raindrops only become important for very heavy rain, while $6-7$ mm raindrops only carry a tiny fraction of precipitation water even for violent rain rate.

the precipitation volume for every raindrop diameter interval. We then divide by the volume $V(d)$ of a single raindrop and this way obtain the number of raindrops $N(d)$ for each raindrop diameter interval separately. The size distribution of raindrops is key for the estimate because raindrops of different size have different impact dynamics and generate vastly different amounts of spray droplets.

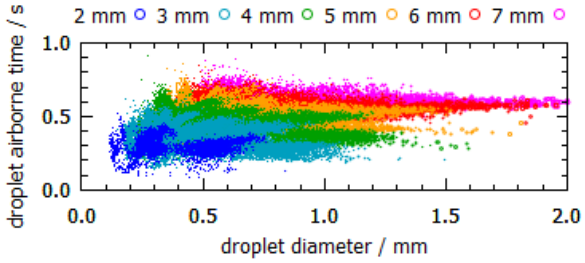
[4]This assumes that the entire precipitation volume V_{pre} is within the diameter range $d \in [0.5\,\mathrm{mm}, 7.5\,\mathrm{mm}]$.

5.2 Local estimate for the number of transitioning microplastic particles

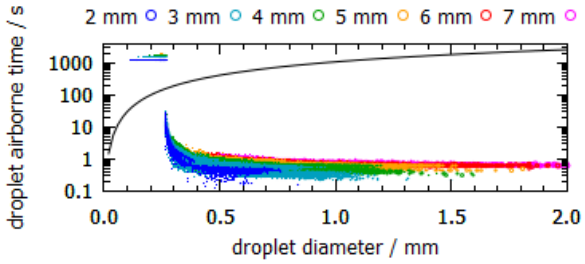
In the second step, we determine the airborne time of spray droplets based on atmospheric updraft by integrating their 3D trajectories with an additional vertical wind velocity offset u_{updraft} as detailed in section 2.3.3 above. Figure 14 (a) shows that without updrafts, airborne time is less than 1 s and atmospheric uptake is considered negligible. Although the typical mean surface wind speed over the ocean is in the order of $8 \pm 4\,\frac{\mathrm{m}}{\mathrm{s}}$ [103, figure 2], the vertical updraft velocity close to the ground is $\lessapprox 1\,\frac{\mathrm{m}}{\mathrm{s}}$ [57]. At this vertical wind velocity, all droplets smaller than 0.26 mm diameter have diverging airborne time (figure 14 (b)), i.e. they are taken up by the atmosphere. Alternatively, if the airborne time remains finite, but droplets evaporate before falling back to the surface, we also consider the contained microplastic particles as being taken up by the atmosphere. Together, these two criteria provide a very clear threshold for which droplets are picked up or fall back down, because all droplets above the lifetime curve have diverging airborne time as a function of updraft velocity (figure 14 (c)). Again, we emphasize that assuming constant updraft velocity is a simplified model and in nature the turbulent air movement is much more complicated.

The number of ejected particles per raindrop impact is rescaled with the concentration of microplastics at the surface, from the 5000 particles per cm^3 in our simulations to a realistic value for the global average microplastic concentration in ocean surface waters, for which we choose 0.0029 particles per cm^3 (2.9 particles per liter) as measured at a depth of 5 m [54]. This value is in line with other measurements across the Atlantic Ocean which gave concentrations between $0.99 - 7.00$ particles per liter at a depth of 10 m [3]. Coincidentally, the size of the microplastics detected by [54] is in the range of $10 - 600\,\mu$m and the size detected by [3] is between $32 - 651\,\mu$m with a mean of 81 μm, very close to the particle sizes used in our simulations. However these measurements likely undercount particles smaller than 10 μm as these are increasingly difficult to detect.

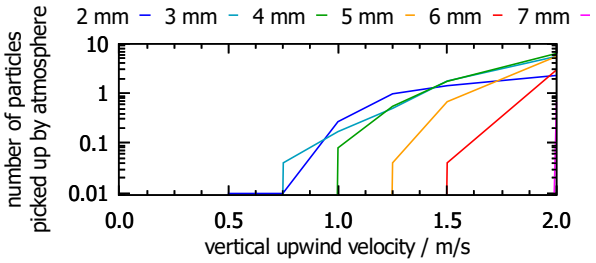
Direct trawl measurements of the microplastic concenteration at the sea surface [20–22] are not suitable here for two reasons: Firstly, these are measurements per area and to convert to a volumetric concentration, one would have to make an assumption about the mean submersion depth. Secondly and more importantly, the trawls have a mesh size of $\approx 330\,\mu$m, which is insufficient to detect the much more numerous tiny particles. However these trawl measurements give a good indication about the large local and temporal



(a) Droplet airborne time without updraft.

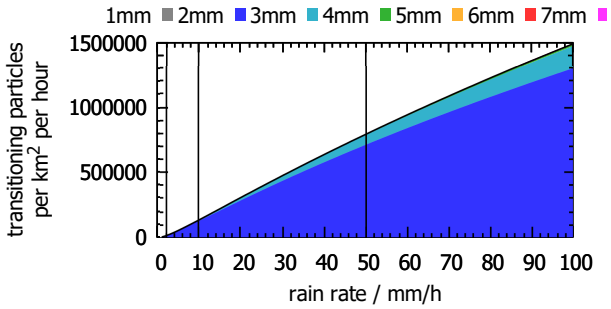


(b) Droplet airborne time with 1 $\frac{\text{m}}{\text{s}}$ vertical updraft.

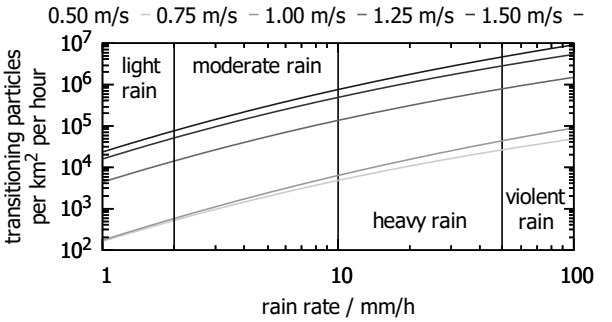


(c) Number of particles with diverging airborne time which are consequently picked up by wind and transfered to the atmosphere.

Figure 14: (a) Airborne time of spray droplets without updraft is in the order of half a second. No droplets are picked up by the atmosphere. (b) With 1 $\frac{\text{m}}{\text{s}}$ vertical updraft velocity, all droplets smaller than 0.26 mm diameter have diverging airborne time (capped at finite values so that the data points are visible in the diagram). The black curve represents the lifetime (time until full evaporation, eq. (11)) of droplets depending on diameter. If the airborne time is larger than the lifetime, the droplets are considered picked up by the atmosphere. (c) The number of picked up microplastic particles per raindrop impact increases with vertical updraft velocity. Numbers are given for an initial concentration of 5000 microplastic particles per cm^3 in the sea water as used in our simulations. Additional figures and data in figure S24 and table S3.



(a) Contribution of different raindrop diameters to particle uptake for 1 $\frac{\text{m}}{\text{s}}$ vertical updraft velocity.



(b) Number of microplastic particles depending on rain rate for various vertical updraft velocities.

Figure 15: (a) The number of microplastic particles transitioning from the oceans into the atmosphere per km^2 per hour depending on rain rate for 1 $\frac{\text{m}}{\text{s}}$ vertical updraft velocity. The contribution of different raindrop sizes to particle transport is illustrated by the colored areas. With rising rain rate, the number of transitioning particles increases as both the number of raindrops and the fraction of larger raindrops become larger. (b) Different vertical updraft velocity strongly affects the number of transitioning particles. For less than 0.5 $\frac{\text{m}}{\text{s}}$, no particles get picked up by the atmosphere.

variation in concentration, spanning across more than two orders of magnitude [3, 20–22].
Next, we multiply the number of raindrops $N(d)$ by the number of ejected particles per raindrop impact $N_p(d)$ to obtain the number of ejected particles for every raindrop diameter. Summing over all diameters between 1 and 7 mm, we obtain the amount of transitioning particles per square kilometer per hour as function of the rain rate which is shown in figure 15. We find that both local rain rate and wind speed strongly affect particle uptake (figure 15). It is interesting that $2-3$ mm diameter raindrops contribute the vast majority to the transitioning particles with larger raindrops only having very minor contribution at violent rain rate (figure 15 (a)). During heavy storms, anything detached from the sea surface may be carried across vast distances by wind, including droplets, particles [17, 18] and even various sea creatures [104, 105]. During mild weather conditions on the other hand, most spray droplets may fall back to the surface before significant atmospheric transport has happened.

5.3 Global estimate for the number of transitioning microplastic particles

In the final step, we use the above considerations to provide a rough estimate for the global annual amount of microplastic transferred to the atmosphere by impacting raindrops.
We first observe that the rain rate itself follows an exponential distribution, where low rain rate is much more frequent and high rain rate is much less frequent. We use the (normalized) Rice-Holmberg model [56]

$$P_{\mathrm{RH}}(\mathcal{R}) = \frac{1}{1.80273}\left(0.03\, e^{-0.03\,\mathcal{R}} + \frac{1-\beta}{5}\left(e^{-0.258\,\mathcal{R}} + 1.86\, e^{-1.63\,\mathcal{R}}\right)\right) \tag{18}$$

with $\beta = 0.2$. According to [55], the average precipitation on the ocean is $2.89 \pm 0.29\,\frac{\mathrm{mm}}{\mathrm{day}}$. Global oceans cover a surface of $3.619 \cdot 10^{14}\,\mathrm{m}^2$, so the total annual precipitation volume on the oceans is $V_{\mathrm{pre,total}} = 3.82 \cdot 10^{14}\,\mathrm{m}^3$. For estimating the annual global amount, we set this value as V_{pre} in our calculations detailed above (figure S25). We then weight the curves in figure S25 with the Rice-Holmberg rain rate distribution (eq. (18), numeric integration with trapezoidal rule). Finally, we assume that vertical wind direction is upward and downward half of the time and no particles transition for downward wind velocity, giving us another factor 1/2. We then find that, depending on which vertical updraft velocity we assume, $7.0{\cdot}10^{13}$ ($0.50\,\frac{\mathrm{m}}{\mathrm{s}}$), $1.0{\cdot}10^{14}$ ($0.75\,\frac{\mathrm{m}}{\mathrm{s}}$), $2.0{\cdot}10^{15}$ ($1.00\,\frac{\mathrm{m}}{\mathrm{s}}$), $7.3 \cdot 10^{15}$ ($1.25\,\frac{\mathrm{m}}{\mathrm{s}}$) or $1.2 \cdot 10^{16}$ ($1.50\,\frac{\mathrm{m}}{\mathrm{s}}$) microplastic particles may transition from global oceans into the atmosphere every year. Because $0.50{-}0.75\,\frac{\mathrm{m}}{\mathrm{s}}$ is already rather large for vertical updraft velocity close to the ground [57], our estimation gives us a clear value for the upper bound, a hundred trillion (10^{14}) microplastic particles per year. For vertical updraft velocities below $0.50\,\frac{\mathrm{m}}{\mathrm{s}}$, based on our definition of atmospheric suspension, no particles transition, so we cannot confidently provide a lower bound for the estimate.

The uncertainties of this estimate are located in the local sea surface microplastic concentration, but also in other simplifications of our model such as assuming mean updraft velocity instead of turbulence. In future research, we envision to use more precise spatially resolved wind and microplastic distribution data using for example satellite measurements [106], such as currently investigated in the TOPIOS project [5, 107, 108], to further narrow down this estimate. This refinement should also include the effect that microplastic particles right after transition to the atmosphere may collide with subsequent raindrops and therefore may be washed out of the atmosphere again leading to a reduction of total transport.

6 Conclusions

We investigated and quantified microplastic particle transport across the water-air interface during raindrop impacts on sea water in great detail using numerical simulations and laboratory experiments.

Depending on raindrop size, each impact ejects in the order of 100 splash droplets into the air with typical vertical velocities of up to $10\,\frac{\mathrm{m}}{\mathrm{s}}$, allowing them to reach altitudes of up to about 80 cm above the sea surface.

Our key result is that the particle concentration in these ejected droplets is about 85% of that in the sea water. This means that there is no 'filter effect' holding particles back and that the fluid from the raindrop, even if it is devoid of particles, is mainly engulfed into the sea while the ejected fluid mainly consists of sea water. We further found that the particle density has negligible influence on the impact dynamics, since gravity-related effects only play a minor role during the short time scale of the impact. The origin of the ejected particles has been identified as a circular region around the impact site very close to the sea surface. No particles are ejected from regions deeper than approximately half the radius of the raindrop. For raindrops impacting the surface at an angle, e.g. due to wind, part of the raindrop fluid is redirected into the spray and droplets can reach slightly higher altitudes. The overall mechanism of microplastic transport is nevertheless operative also for oblique drop impacts.

Our laboratory experiments of artificially produced raindrops with a well-defined size and velocity on a reservoir filled with high concentrations of microplastic particles are in good agreement with the simulation predictions. The experiments clearly confirm that microplastic particles are contained in the ejected spray droplets.

Based on our simulations of a single raindrop event, assuming an average microplastic particle concentration of 2.9 particles per liter at the ocean surface, we estimate that a realistic upper bound for the annual number of microplastic particles transitioning during rainfall from global oceans into the atmosphere is a hundred trillion (10^{14}). This estimate contains a number of uncertainties such as microplastic concentration at the ocean surface, calling for additional research to narrow this range further down.

Acknowledgements

We acknowledge support through the computational resources provided by the Bavarian Polymer Institute. We acknowledge the NVIDIA Corporation for donating a Titan Xp GPU for our research.

Funding

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 391977956 – SFB1357.

Availability of data and materials

The data sets used and/or analyzed during the current study are publicly available on Zenodo (https://zenodo.org/record/5683801) and the simulation source code is available from the corresponding author upon reasonable request.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

ML and FH contributed the software used for the simulations. ML and SG contributed design of the study. ML contributed to data analysis and evaluation. ML, SG, LMO and AH contributed writing and literature research. ML conducted the simulations. LMO and AH conducted the experiments.

7 References

[1] Chelsea M Rochman. "Microplastics research—from sink to source". In: *Science* 360.6384 (2018), pp. 28–29.

[2] Andrés Cózar et al. "Plastic debris in the open ocean". In: *Proceedings of the National Academy of Sciences* 111.28 (2014), pp. 10239–10244.

[3] Katsiaryna Pabortsava and Richard S Lampitt. "High concentrations of plastic hidden beneath the surface of the Atlantic Ocean". In: *Nature communications* 11.1 (2020), pp. 1–11.

[4] Lucy C Woodall et al. "The deep sea is a major sink for microplastic debris". In: *Royal Society open science* 1.4 (2014), p. 140317.

[5] Victor Onink et al. "Global simulations of marine plastic transport show plastic trapping in coastal zones". In: *Environmental Research Letters* 16.6 (2021), p. 064053.

[6] Steve Allen et al. "Atmospheric transport and deposition of microplastics in a remote mountain catchment". In: *Nature Geoscience* 12.5 (2019), pp. 339–344.

[7] Miri Trainic et al. "Airborne microplastic particles detected in the remote marine atmosphere". In: *Communications Earth & Environment* 1.1 (2020), pp. 1–9.

[8] V. Tirelli, G. Suaria, and A. L. Lusher. "Microplastics in Polar Samples". In: *Handbook of Microplastics in the Environment*. Ed. by Teresa Rocha-Santos, Mónica Costa, and Catherine Mouneyrac. Cham: Springer International Publishing, 2020, pp. 1–42. ISBN: 978-3-030-10618-8. DOI: 10.1007/978-3-030-10618-8_4-1.

[9] Imogen E Napper et al. "Reaching new heights in plastic pollution—preliminary findings of microplastics on Mount Everest". In: *One Earth* 3.5 (2020), pp. 621–630.

[10] Henri Lhuissier and Emmanuel Villermaux. "Bursting bubble aerosols". In: *Journal of Fluid Mechanics* 696 (2012), p. 5.

[11] Elisabeth Ghabache and Thomas Séon. "Size of the top jet drop produced by bubble bursting". In: *Physical Review Fluids* 1.5 (2016), p. 051901.

[12] Alexis Berny et al. "Role of all jet drops in mass transfer from bursting bubbles". In: *Physical Review Fluids* 5.3 (2020), p. 033605.

[13] Maria Masry et al. "Experimental evidence of plastic particles transfer at the water-air interface through bubble bursting". In: *Environmental Pollution* 280 (2021), p. 116949.

[14] Fabrice Veron. "Ocean spray". In: *Annual Review of Fluid Mechanics* 47 (2015), pp. 507–538.

[15] Duncan C Blanchard. "Sea-to-air transport of surface active material". In: *Science* 146.3642 (1964), pp. 396–397.

[16] Duncan C Blanchard. "Jet drop enrichment of bacteria, virus, and dissolved organic material". In: *pure and applied geophysics* 116.2-3 (1978), pp. 302–308.

[17] John A Quinn, Richard A Steinbrook, and John L Anderson. “Breaking bubbles and the water-to-air transport of particulate matter”. In: *Chemical Engineering Science* 30.9 (1975), pp. 1177–1184.

[18] Gerrit de Leeuw et al. “Production flux of sea spray aerosol”. In: *Rev. Geophys.* 49.2 (May 2011), pp. 13–39.

[19] Colin D O’Dowd et al. “Marine aerosol, sea-salt, and the marine sulphur cycle: A short review”. In: *Atmospheric Environment* 31.1 (1997), pp. 73–80.

[20] Ana L d F Lacerda et al. “Plastics in sea surface waters around the Antarctic Peninsula”. In: *Scientific reports* 9.1 (2019), pp. 1–12.

[21] Trishan Naidoo and David Glassom. “Sea-surface microplastic concentrations along the coastal shelf of KwaZulu–Natal, South Africa”. In: *Marine pollution bulletin* 149 (2019), p. 110514.

[22] Tamara Gajšt et al. “Sea surface microplastics in Slovenian part of the Northern Adriatic”. In: *Marine pollution bulletin* 113.1-2 (2016), pp. 392–399.

[23] Giuseppe Suaria et al. “Microfibers in oceanic surface waters: A global characterization”. In: *Science Advances* 6.23 (2020), eaay8493.

[24] Tiera-Brandy Robinson, Helge-Ansgar Giebel, and Oliver Wurl. “Riding the plumes: characterizing bubble scavenging conditions for the enrichment of the sea-surface microlayer by transparent exopolymer particles”. In: *Atmosphere* 10.8 (2019), p. 454.

[25] Zachary T Anderson et al. “A rapid method for assessing the accumulation of microplastics in the sea surface microlayer (SML) of estuarine systems”. In: *Scientific reports* 8.1 (2018), pp. 1–11.

[26] Jessica L Stead et al. “Identification of tidal trapping of microplastics in a temperate salt marsh system using sea surface microlayer sampling”. In: *Scientific reports* 10.1 (2020), pp. 1–10.

[27] Young Kyoung Song et al. “Large accumulation of micro-sized synthetic polymer particles in the sea surface microlayer”. In: *Environmental science & technology* 48.16 (2014), pp. 9014–9021.

[28] Steve Allen et al. “Examination of the ocean as a source for atmospheric microplastics”. In: *PloS one* 15.5 (2020), e0232746.

[29] Federico Porcù et al. “Effects of altitude on maximum raindrop size and fall velocity as limited by collisional breakup”. In: *Journal of the atmospheric sciences* 70.4 (2013), pp. 1129–1134.

[30] John H van Boxel et al. “Numerical model for the fall speed of rain drops in a rain fall simulator”. In: *Workshop on wind and water erosion.* 1997, pp. 77–85.

[31] John S Marshall and W Mc K Palmer. “The distribution of raindrops with size”. In: *Journal of meteorology* 5.4 (1948), pp. 165–166.

[32] Athelstan F Spilhaus. “Raindrop size, shape and falling speed”. In: *Journal of Meteorology* 5.3 (1948), pp. 108–110.

[33] Emmanuel Villermaux and Benjamin Bossa. “Single-drop fragmentation determines size distribution of raindrops”. In: *Nature Physics* 5.9 (2009), pp. 697–702.

[34] Martin Rein. “Phenomena of liquid drop impact on solid and liquid surfaces”. In: *Fluid Dynamics Research* 12.2 (1993), p. 61.

[35] Martin Rein. “The transitional regime between coalescing and splashing drops”. In: *Journal of Fluid Mechanics* 306 (1996), pp. 145–165.

[36] Alexander I Fedorchenko and An-Bang Wang. “On some common features of drop impact on liquid surfaces”. In: *Physics of Fluids* 16.5 (2004), pp. 1349–1365.

[37] Alfio Bisighini and Gianpietro Elvio Cossali. “High-speed visualization of interface phenomena: single and double drop impacts onto a deep liquid layer”. In: *Journal of visualization* 14.2 (2011), pp. 103–110.

[38] NN Myagkov and TA Shumikhin. “Modeling of high-velocity impact ejecta by experiments with a water drop impacting on a water surface”. In: *Acta Mechanica* 227.10 (2016), pp. 2911–2924.

[39] Liow Jong Leng. “Splash formation by spherical drops”. In: *Journal of Fluid Mechanics* 427 (2001), pp. 73–105.

[40] Bahni Ray, Gautam Biswas, and Ashutosh Sharma. “Regimes during liquid drop impact on a liquid pool”. In: *Journal of Fluid Mechanics* 768 (2015), pp. 492–523.

[41] Guy-Jean Michon, Christophe Josserand, and Thomas Séon. “Jet dynamics post drop impact on a deep pool”. In: *Physical Review Fluids* 2.2 (2017), p. 023601.

[42] Quan Ding, Tianyou Wang, and Zhizhao Che. "Two jets during the impact of viscous droplets onto a less-viscous liquid pool". In: *Physical Review E* 100.5 (2019), p. 053108.

[43] Marise V Gielen et al. "Oblique drop impact onto a deep liquid pool". In: *Physical review fluids* 2.8 (2017), p. 083602.

[44] Hasan N Oguz and Andrea Prosperetti. "Bubble entrainment by the impact of drops on liquid surfaces". In: *Journal of Fluid Mechanics* 219 (1990), pp. 143–179.

[45] C Motzkus, E Géhin, and F Gensdarmes. "Study of airborne particles produced by normal impact of millimetric droplets onto a liquid film". In: *Experiments in fluids* 45.5 (2008), p. 797.

[46] David W Murphy et al. "Splash behaviour and oily marine aerosol production by raindrops impacting oil slicks". In: *Journal of Fluid Mechanics* 780 (2015), p. 536.

[47] Herman Medwin et al. "The anatomy of underwater rain noise". In: *The Journal of the Acoustical Society of America* 92.3 (1992), pp. 1613–1623.

[48] Andrea Prosperetti and Hasan N Oguz. "The impact of drops on liquid surfaces and the underwater noise of rain". In: *Annual Review of Fluid Mechanics* 25.1 (1993), pp. 577–602.

[49] PK Wang and HR Pruppacher. "Acceleration to terminal velocity of cloud and raindrops". In: *Journal of Applied Meteorology* 16.3 (1977), pp. 275–280.

[50] Yisen Guo and Yongsheng Lian. "High-speed oblique drop impact on thin liquid films". In: *Physics of Fluids* 29.8 (2017), p. 082108.

[51] Moritz Lehmann. *High Performance Free Surface LBM on GPUs*. 2019. URL: https://epub.uni-bayreuth.de/5400/.

[52] Moritz Lehmann and Stephan Gekle. "Analytic Solution to the Piecewise Linear Interface Construction Problem and its Application in Curvature Calculation for Volume-of-Fluid Simulation Codes". In: *arXiv preprint arXiv:2006.12838* (2020).

[53] Fabian Häusl. *MPI-based multi-GPU extension of the Lattice Boltzmann Method*. 2019. URL: https://epub.uni-bayreuth.de/5689/.

[54] C Anela Choy et al. "The vertical distribution and biological transport of marine microplastics across the epipelagic and mesopelagic water column". In: *Scientific reports* 9.1 (2019), pp. 1–9.

[55] Robert F Adler et al. "Global precipitation: Means, variations and trends during the satellite era (1979–2014)". In: *Surveys in Geophysics* 38.4 (2017), pp. 679–699.

[56] Philip Rice and Nettie Holmberg. "Cumulative time statistics of surface-point rainfall rates". In: *IEEE Transactions on Communications* 21.10 (1973), pp. 1131–1136.

[57] Huug G Ouwersloot et al. "Vertical wind velocity observations at the Cabauw tower". In: *19th Symposium on Boundary Layers and Turbulence, American Meteorological Society, 2-6 August 2010, Keystone, Colorado, USA*. 2010, P1–2.

[58] Christian Obrecht et al. "A new approach to the lattice Boltzmann method for graphics processing units". In: *Computers & Mathematics with Applications* 61.12 (2011), pp. 3628–3638.

[59] Markus Wittmann. "Hardware-effiziente, hochparallele Implementierungen von Lattice-Boltzmann-Verfahren für komplexe Geometrien". In: (2016).

[60] Nicolas Delbosc et al. "Optimized implementation of the Lattice Boltzmann Method on a graphics processing unit towards real-time fluid simulation". In: *Computers & Mathematics with Applications* 67.2 (2014), pp. 462–475.

[61] Gregory Herschlag et al. "Gpu data access on complex geometries for d3q19 lattice boltzmann method". In: *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2018, pp. 825–834.

[62] Mark J Mawson and Alistair J Revell. "Memory transfer optimization for a lattice Boltzmann solver on Kepler architecture nVidia GPUs". In: *Computer Physics Communications* 185.10 (2014), pp. 2566–2574.

[63] Markus Wittmann et al. "Comparison of different propagation steps for lattice Boltzmann methods". In: *Computers & Mathematics with Applications* 65.6 (2013), pp. 924–935.

[64] Frédéric Kuznik et al. "LBM based flow simulation using GPU computing processor". In: *Computers & Mathematics with Applications* 59.7 (2010), pp. 2380–2392.

[65] Christian Obrecht et al. "Multi-GPU implementation of the lattice Boltzmann method". In: *Computers & Mathematics with Applications* 65.2 (2013), pp. 252–261.

[66] Timm Krüger et al. "The lattice Boltzmann method". In: *Springer International Publishing* 10 (2017), pp. 978–3.

[67] Sydney Chapman, Thomas George Cowling, and David Burnett. *The mathematical theory of non-uniform gases: an account of the kinetic theory of viscosity, thermal conduction and diffusion in gases*. Cambridge, UK: Cambridge university press, 1990.

[68] Acep Purqon et al. "Accuracy and Numerical Stabilty Analysis of Lattice Boltzmann Method with Multiple Relaxation Time for Incompressible Flows". In: *Journal of Physics: Conference Series*. Vol. 877. 1. IOP Publishing. 2017, p. 012035.

[69] Xiongwei Cui et al. "A Coupled Two-relaxation-time Lattice Boltzmann-Volume Penalization method for Flows Past Obstacles". In: *arXiv preprint arXiv:1901.08766* (2019).

[70] Zhaoli Guo and Chang Shu. *Lattice Boltzmann method and its applications in engineering*. Vol. 3. Singapore: World Scientific, 2013.

[71] Shimpei Saito, Yutaka Abe, and Kazuya Koyama. "Lattice Boltzmann modeling and simulation of liquid jet breakup". In: *Physical Review E* 96.1 (2017), p. 013317.

[72] A Kuzmin, ZL Guo, and AA Mohamad. "Simultaneous incorporation of mass and force terms in the multi-relaxation-time framework for lattice Boltzmann schemes". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 369.1944 (2011), pp. 2219–2227.

[73] Zhaoli Guo, Chuguang Zheng, and Baochang Shi. "Discrete lattice effects on the forcing term in the lattice Boltzmann method". In: *Physical Review E* 65.4 (2002), p. 046308.

[74] Thomas Pohl. *High performance simulation of free surface flows using the lattice Boltzmann method*. Erlangen, Germany: Verlag Dr. Hut, 2008.

[75] Stefan Donath. *Wetting Models for a Parallel High-performance Free Surface Lattice Boltzmann Method: Benetzungsmodelle Für Eine Parallele Lattice-Boltzmann-Methode Mit Freien Oberflächen*. Erlangen, Germany: Verlag Dr. Hut, 2011.

[76] Carolin Körner et al. "Lattice Boltzmann model for free surface flow for modeling foaming". In: *Journal of Statistical Physics* 121.1-2 (2005), pp. 179–196.

[77] Nils Thürey, C Körner, and U Rüde. "Interactive free surface fluids with the lattice Boltzmann method". In: *Technical Report05-4. University of Erlangen-Nuremberg, Germany* (2005).

[78] Martin Schreiber and DTMP Neumann. "GPU based simulation and visualization of fluids with free surfaces". PhD thesis. Diploma Thesis, Technische Universität München, 2010.

[79] Christian Janßen and Manfred Krafczyk. "Free surface flow simulations on GPGPUs using the LBM". In: *Computers & Mathematics with Applications* 61.12 (2011), pp. 3549–3563.

[80] Simon Bogner, Ulrich Rüde, and Jens Harting. "Curvature estimation from a volume-of-fluid indicator function for the simulation of surface tension and wetting with a free-surface lattice Boltzmann method". In: *Physical Review E* 93.4 (2016), p. 043302.

[81] David L Youngs. "An interface tracking method for a 3D Eulerian hydrodynamics code". In: *Atomic Weapons Research Establishment (AWRE) Technical Report* 44.92 (1984), p. 35.

[82] Ruben Scardovelli and Stephane Zaleski. "Analytical relations connecting linear interfaces and volume fractions in rectangular grids". In: *Journal of Computational Physics* 164.1 (2000), pp. 228–237.

[83] Akio Kawano. "A simple volume-of-fluid reconstruction method for three-dimensional two-phase flows". In: *Computers & Fluids* 134 (2016), pp. 130–145.

[84] Paul Bourke. *Polygonising a scalar field*. 1994.

[85] Charles S Peskin. "The immersed boundary method". In: *ANU* 11 (July 2003), pp. 479–517.

[86] Timm Krüger. "Introduction to the immersed boundary method". In: *LBM Workshop, Edmonton*. 2011.

[87] Paul Bourke. "Interpolation methods". In: *Miscellaneous: projection, modelling, rendering*. 1 (1999).

[88] Anca Hamuraru. *Atomic operations for floats in OpenCL – improved*. 2016. URL: https://streamhpc.com/blog/2016-02-09/atomic-operations-for-floats-in-opencl-improved/ (visited on 11/15/2019).

[89] Stefan Frijters, Timm Krüger, and Jens Harting. "Parallelised Hoshen–Kopelman algorithm for lattice-Boltzmann simulations". In: *Computer Physics Communications* 189 (2015), pp. 92–98.

[90] HJ Holterman. *Kinetics and evaporation of water drops in air*. Vol. 2012. Wageningen, Netherlands: Citeseer, 2003.

[91] Faith A Morrison. *An introduction to fluid mechanics*. New York, USA: Cambridge University Press, 2013.

[92] Zhi-Gang Feng and Efstathios E Michaelides. "Drag coefficients of viscous spheres at intermediate and high Reynolds numbers". In: *J. Fluids Eng.* 123.4 (2001), pp. 841–849.

[93] Cameron Beccario. *earth.nullschool.net*. 2021. URL: https://earth.nullschool.net/ (visited on 06/25/2021).

[94] The Engineering ToolBox. *Air - Density, Specific Weight and Thermal Expansion Coefficient at Varying Temperature and Constant Pressures*. 2020. URL: https://www.engineeringtoolbox.com/air-absolute-kinematic-viscosity-d_601.html (visited on 08/23/2020).

[95] The Engineering ToolBox. *Air - Dynamic and Kinematic Viscosity*. 2020. URL: https://www.engineeringtoolbox.com/air-absolute-kinematic-viscosity-d_601.html (visited on 07/29/2020).

[96] ITTC Specialist Committee et al. "ITTC–Recommended Procedures Fresh Water and Seawater Properties". In: IITC. 2011.

[97] The International Association for the Properties of Water and Steam. "Guideline on the Surface Tension of Seawater". In: (2019).

[98] Raymond Bergmann et al. "Controlled impact of a disk on a water surface: cavity dynamics". In: *J. Fluid Mech.* 633 (Aug. 2009), p. 381.

[99] Javad Eshraghi, Sunghwan Jung, and Pavlos P Vlachos. "To seal or not to seal: The closure dynamics of a splash curtain". In: *Phys. Rev. Fluids* 5.10 (Sept. 2020), p. 104001.

[100] Stephan Gekle et al. "High-speed jet formation after solid object impact." In: *Phys. Rev. Lett.* 102.3 (Jan. 2009), p. 034502.

[101] Albert Einstein. "Eine neue Bestimmung der Moleküldimensionen". PhD thesis. ETH Zurich, 1905.

[102] Brian M Haines and Anna L Mazzucato. "A proof of Einstein's effective viscosity for a dilute suspension of spheres". In: *SIAM Journal on Mathematical Analysis* 44.3 (2012), pp. 2120–2145.

[103] Adam Hugh Monahan. "The probability distribution of sea surface wind speeds. Part I: Theory and SeaWinds observations". In: *Journal of climate* 19.4 (2006), pp. 497–520.

[104] WALDOL McATEE. "Showers of organic matter". In: *Monthly Weather Review* 45.5 (1917), pp. 217–224.

[105] GILBERT P Whitley. "Rains of fishes in Australia". In: *Australian Natural History* 17 (1972), pp. 154–159.

[106] Madeline C Evans and Christopher S Ruf. "Toward the Detection and Imaging of Ocean Microplastics With a Spaceborne Radar". In: *IEEE Transactions on Geoscience and Remote Sensing* (2021).

[107] David Wichmann, Philippe Delandmeter, and Erik van Sebille. "Influence of near-surface currents on the global dispersal of marine microplastic". In: *Journal of Geophysical Research: Oceans* 124.8 (2019), pp. 6086–6096.

[108] David Wichmann et al. "Mixing of passive tracers at the ocean surface and its implications for plastic transport modelling". In: *Environmental Research Communications* 1.11 (2019), p. 115001.

SI – Ejection of marine microplastics by raindrops: a computational and experimental study

Moritz Lehmann[1*], Lisa Marie Oehlschlägel[2], Fabian Häusl[1], Andreas Held[2] and Stephan Gekle[1]

November 12, 2021

*Correspondence:
moritz.lehmann@uni-bayreuth.de
[1]Biofluid Simulation and Modeling – Theoretische Physik VI, University of Bayreuth
[2]Technischer Umweltschutz, Fachgebiet Umweltchemie und Luftreinhaltung, Technische Universität Berlin

S1 Model Details

S1.1 Estimating Hydrodynamic Diameter of IBM Particles by Sedimentation

Although the IBM particles have no intrinsic size, they interact with the LBM lattice at the length scale of the distance between two neighboring lattice points, which in LBM units is 1. Here we check the hydrodynamic particle diameter with the Stokes drag. The entire test is in LBM units. We let a single IBM particle float vertically upwards or downwards and measure its velocity u_z to determine its diameter $d_p = 2\,R_p$ via the force balance of buoyant force F_g and drag force F_{Stokes}. m_p and ρ_p are the mass and density of the particle, m is the mass of the displaced fluid, g is the gravitational constant and ν and $\rho = 1$ are the kinematic shear viscosity and density of the fluid.

$$(m_p - m)\,g = F_g = F_{\text{Stokes}} = 6\,\pi\,\nu\,\rho\,R_p\,u_z \tag{S1}$$

$$\left(\frac{\rho_p}{\rho} - 1\right) V\,f = 6\,\pi\,\nu\,\rho\,R_p\,u_z \tag{S2}$$

$$\left(\frac{\rho_p}{\rho} - 1\right) \frac{4}{3}\,\pi\,R_p^3\,f = 6\,\pi\,\nu\,\rho\,R_p\,u_z \tag{S3}$$

$$d_p = 2\,R_p = \sqrt{\frac{18\,\nu\,\rho^2\,u_z}{(\rho_p - \rho)\,f}} \tag{S4}$$

Here f is the force F per volume $\tilde{V}$:

$$f = \frac{F}{\tilde{V}} = \frac{m\,g}{\tilde{V}} = \frac{\rho\,\tilde{V}\,g}{\tilde{V}} = \rho\,g \tag{S5}$$

We expect $d_p \approx 1$ no matter what. To test this, we place a single IBM particle in the middle of a 128^3 simulation box with the outmost layer of lattice points being (non-moving bounce-back) boundary points. We vary ρ_p and ν. To have consistent tests, we adjust f such that the target sedimentation velocity of a particle with hydrodynamic diameter $d_{p,\text{target}} = 1$ is constant at $u_{z,\text{target}}$:

$$f = f(\rho_p, \nu) = \frac{18\,\nu\,\rho^2\,u_{z,\text{target}}}{(\rho_p - \rho)\,d_{p,\text{target}}^2} \tag{S6}$$

To also make sure the measurement is generally independent of f, we choose two different velocities $u_{z,\text{target}} \in \{10^{-4}, 10^{-5}\}$. With these settings, we first

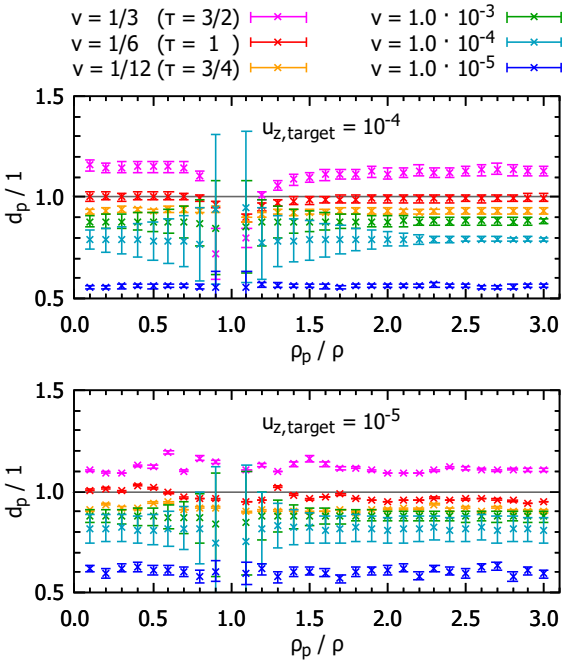


Figure S1: The hydrodynamic diameter of IBM particles for varying particle material density ρ_p and kinematic shear viscosity ν at varying volume force f such that the target sedimentation velocity is $u_{z,\text{target}} = 10^{-4}$ (top) and $u_{z,\text{target}} = 10^{-5}$ (bottom).

simulate 100000 LBM time steps so that the particle velocity reaches equilibrium. In this time, the particle moves approximately 10 ($u_{z,\text{target}} = 10^{-4}$) or 1 ($u_{z,\text{target}} = 10^{-5}$) lattice points up or down from the box center depending on $\rho_p \lessgtr \rho$. We then measure the z-velocity 100 times at 100 LBM time steps apart, calculate the corresponding hydrodynamic diameter and take the average as one data point and the standard deviation as its uncertainty. The results are presented in figure S1. We observe almost no variation of d_p when varying ρ_p except for the region around $\rho_p \approx \rho$, where due to the nature of the test errors increase when approaching neutral buoyancy. However there is some variation in d_p when varying ν: when decreasing ν by several orders of magnitude, d_p becomes slightly smaller.

S1.2 Estimating Time of Droplet Separation from Crown

The droplet tracking algorithm only delivers the droplet radius R_0, velocity $\vec{u}_0$ (with vertical component u_z) and detection time t_d when a droplet touches the ceiling of the simulation box at altitude h_air over the undisturbed pool surface. From this, we reconstruct the approximate time of separation from the crown rim based on the measured time of detection, droplet radius (position of the droplet center at time of detection), time-dependent altitude of the crown rim and the vertical velocity at the time of detection:

$$t_\text{s} \approx t_\text{d} - \frac{h_\text{air} - R_0 - h_\text{crown}(t_\text{s})}{u_z} \tag{S7}$$

We obtain $h_\text{crown}(t)$ from the generated images (figure 2 (b) and equivalent for other raindrop diameters). For each of the images, we determine an interval of confidence for the vertical position of the crown rim. In figure S2 we then fit an expression of the form

$$h_\text{crown}(t) = b - \frac{a}{t + \frac{a}{b}} \tag{S8}$$

to the data points so that, with $k = h_\text{air} - R_0 - b$, we have an implicit equation that we can solve for t_s:

$$0 = t_\text{s}^2 + \left(\frac{k\,b + u_z\,a}{u_z\,b} - t_\text{d}\right) t_\text{s} + a\,\frac{k + b - u_z\,t_\text{d}}{u_z\,b} \tag{S9}$$

$$t_\text{s} = \frac{t_\text{d}}{2} - \frac{k\,b + u_z\,a}{2\,u_z\,b} \overset{+}{(-)} \sqrt{\frac{1}{4}\left(\frac{k\,b + u_z\,a}{u_z\,b} - t_\text{d}\right)^2 - a\,\frac{k + b - u_z\,t_\text{d}}{u_z\,b}} \tag{S10}$$

The only assumption in this approach is that u_z is constant during the 1 to 2 cm flight distance between the moment of separation from the crown and the moment of detection.

We show the maximum altitude of ejected droplets by time of detection in figure S3 (a) and by approximate time of separation from crown in figure S3 (b). The fitted parameters are given in table S1. While in S3 (a) the maximum altitude clearly drops with time – in part caused by fast droplets moving faster through the simulation box and being detected earlier, the decrease of maximum altitude by time is still present in S3 (b), although with more noise. We observe that the droplets separating from the crown have highest velocity in the beginning and lower velocity in the later phase of the splash. The time at which the last droplets (detected at 10 ms) separate from the crown rim is between 2 ms and 5 ms depending on raindrop diameter; however the border is somewhat diffuse.

Based on this, we introduce the cut-off altitude h_{cut}: Above h_{cut}, almost all droplets are counted within the simulated 10 ms time frame whereas below h_{cut}, after the simulated time frame there may be more droplets in this regime. This means that h_{cut} is a threshold showing us where our data is complete or incomplete.

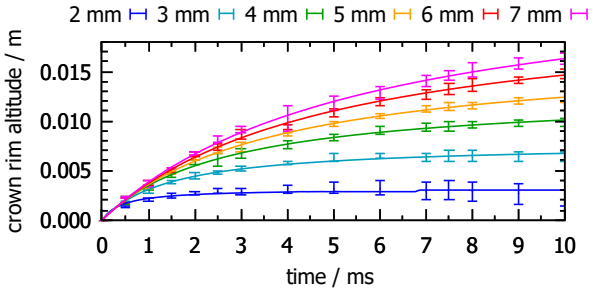


Figure S2: Confidence intervals for the time-dependent altitude of the crown rim $h_\text{crown}(t)$ for different raindrop diameters with equation (S8) fitted.

d/mm	a/(m · ms)	b/m
2	0.00139 ± 0.00055	0.00313 ± 0.00016
3	0.01177 ± 0.00114	0.00776 ± 0.00016
4	0.03451 ± 0.00211	0.01282 ± 0.00021
5	0.06412 ± 0.00353	0.01708 ± 0.00028
6	0.10694 ± 0.00719	0.02183 ± 0.00046
7	0.14563 ± 0.00918	0.02559 ± 0.00053

Table S1: Fitted parameters for eq. (S8).

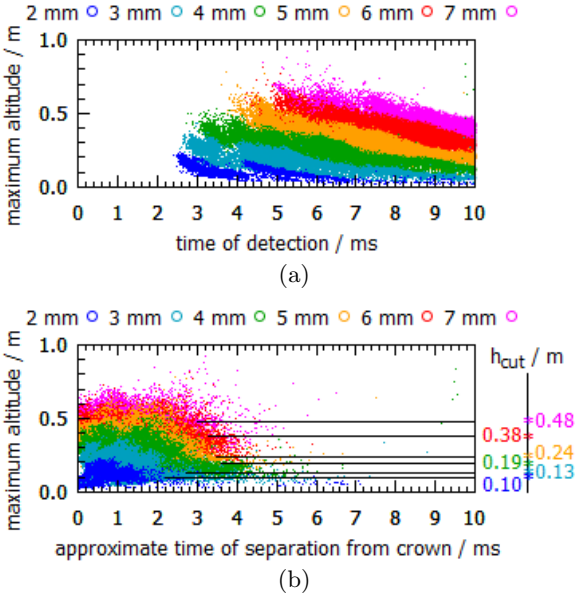


Figure S3: Maximum altitude of ejected droplets by time of detection (a) and by approximate time of separation from crown (b) via calibration through eq. (S10). (b) defines the cut-off altitude h_{cut}, indicated by horizontal lines: Above h_{cut}, almost all droplets are counted within the simulated 10 ms time frame whereas below h_{cut}, after the simulated time frame there may be more droplets in this regime.

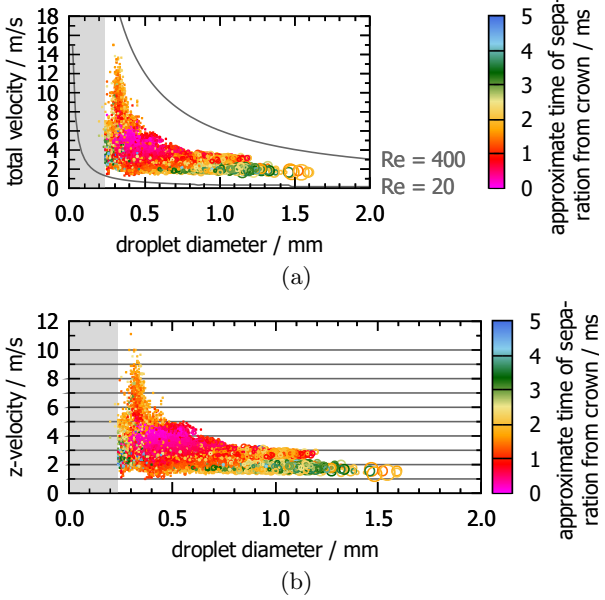


Figure S4: Initial velocity of ejected droplets for the 4 mm diameter raindrop: (a) Total velocity $|\vec{u}_0|$ and (b) z-component of velocity u_z. Total velocity allows estimating the Reynolds number range while the z-component plot with horizontal lines eases understanding of how the point cloud is warped and stretched in the maximum altitude plot in figure 3 in the main text.

S1.3 Drag Force Model for Ejected Droplets

The initial total and z-velocity of the ejected droplets from a 4 mm diameter raindrop are provided in figure S4. From the initial total velocity, we estimate that the Reynolds number initially is in the range 20 – 400 (see figure S4), decreasing as droplets are slowed down by gravity and drag force.

The drag force on a sphere with radius R and velocity $\vec{u}$ moving through a fluid (in our case air) follows

$$\vec{F}_{\text{drag}} = -\frac{1}{2}\,\rho_{\text{a}}\,A\,C_D\,|\vec{u}|\,\vec{u} = -\beta\,C_D\,|\vec{u}|\,\vec{u} \qquad \text{(S11)}$$

with ρ_{a} being the air density and $A = \pi R^2$ being the cross-section of the sphere. $C_D = C_D(Re(u_z))$ is the drag coefficient for a sphere that depends on the Reynolds number $Re = \frac{2R\rho_{\text{a}}u}{\mu_{\text{a}}}$.

Considering not a solid sphere, but a viscous droplet with (dynamic) viscosity contrast λ, Feng [1] provides

$$C_D(Re,\lambda) = \frac{68\,Re^{-\frac{2}{3}}}{\lambda+2} + \frac{\lambda-2}{\lambda+2}\,\frac{24}{Re}\left(1+\frac{Re^{\frac{2}{3}}}{6}\right) \qquad \text{(S12)}$$

for the range $5 < Re \leq 1000$ and

$$\gamma = \frac{3\,\lambda+2}{\lambda+1} \qquad \text{(S13)}$$

$$C_D(Re,\lambda) = \frac{8\,\gamma}{Re}\left(1+\frac{\gamma\,Re}{20}\right) - \frac{\gamma\,Re\,\log(Re)}{100} \qquad \text{(S14)}$$

for the range $0 \leq Re \leq 5$. In our case ($\lambda = \frac{\rho\,\nu}{\mu_{\text{a}}} \approx 59.40$), Feng's model is very close to the solid sphere model by Morrison [2, 3]. We illustrate these models in figure S5.

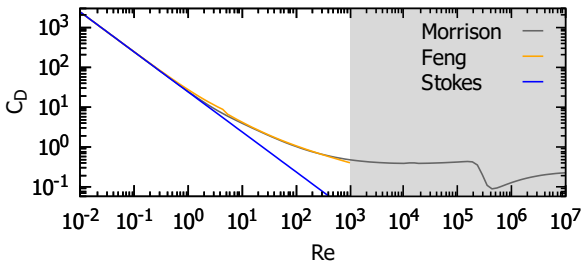


Figure S5: Viscous droplet (Feng) versus solid sphere (Morrison) models for the drag coefficient of a sphere $C_D(Re)$ illustrated. We are only interested in the range $Re < 1000$ (white area), because the ejected droplets do not go any faster. Stokes law ($C_D = \frac{24}{Re}$, blue line) would greatly underestimate the drag force for fast-moving droplets [4].

S2 Model Validation

S2.1 Validation of VoF and Surface Tension Model

A surface tension phenomenon to which the analytic solution is known is the Plateau-Rayleigh instability. It is visible whenever a laminar, thin stream of liquid breaks up into droplets, for example at a faucet. Here we study the isolated phenomenon on a perturbed cylinder of fluid without gravity as a way to validate our VoF and curvature calculation models [5, 6]. We do the simulations with D3Q19 SRT, density $\rho = 1$, initial velocity $\vec{u} = 0$ everywhere, relaxation time $\tau = 1$ and surface tension $\sigma = 0.1$.
According to the analytic solution, we expect that that perturbations with $\lambda < 2\,\pi\,R$ are unstable and decay and that $\lambda_{\max} \approx 9\,R$ is the perturbation wavelength of maximum growth rate [7]. Our simulations (illustrated exemplary in figure S6), that only take a few seconds of compute time each, show good agreement with the analytic solution.
To examine the growth rate quantitatively, we measure the cylinder (or later sphere) radius at the maximum of the initial undulation as the average of the diameters in x- and z-directions. The cylinder of fluid is aligned along the y-axis. We do this measurement after every LBM time step for 3000 time steps for $\lambda \in \{1, 2, 3, ..., 25\}$ [5]. The initial radius of the cylinder is $R = 8$ and the initial perturbation amplitude is $A = 0.1\,R$. The relaxation time is $\tau = 1$ and the surface tension coefficient is $\sigma = 0.1$. The results are plotted in figure S7.

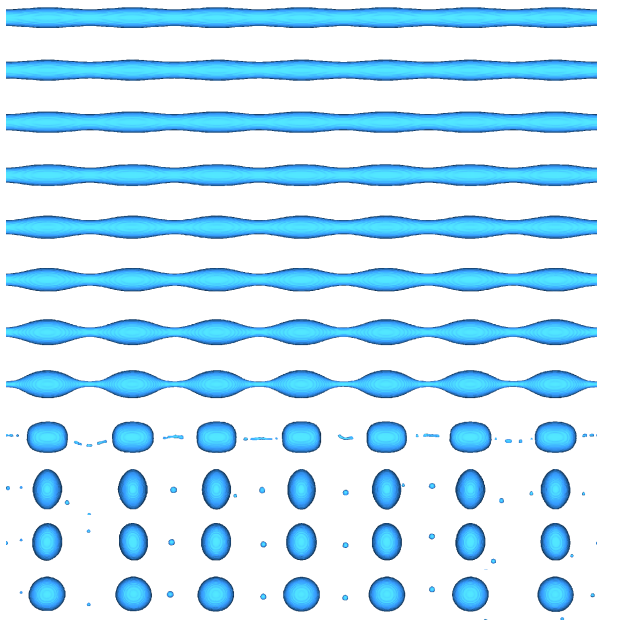


Figure S6: A periodic cylinder of fluid (length $L = 512$, radius $R = \frac{512}{7 \cdot 9}$) is initially perturbed with a sine wave (amplitude $A = 0.1\,R$, wavelength $\lambda = 9\,R$). The perturbation grows until the cylinder separates into droplets. The images (top to bottom) are each separated by 100 LBM time steps.

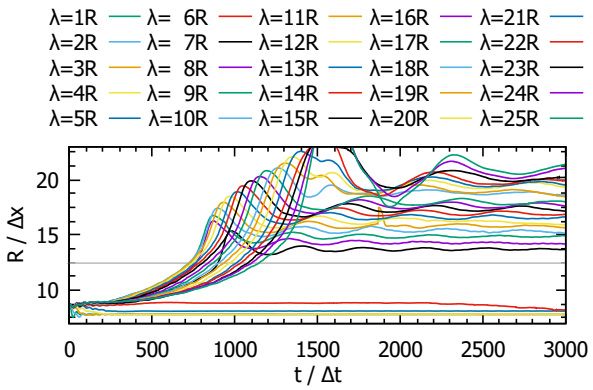


Figure S7: The center radius R of the fluid cylinder (and later sphere) measured over time. For different wavelengths λ of the initial undulation, R will either decay from its initial perturbed state at $R(t = 0\,s) = 8.8$ down to $R = 8.0$ or increase exponentially until the cylinder separates into droplets. The separation radius of approximately $R_{\text{sep}} = 12.5$ is indicated by the horizontal gray line. After separation, the slope of the curve increases even more until the curve reaches a maximum, after which the drop relaxes to a constant radius. For some curves there is a ripple later in the curve caused by satellite droplets fusing with the main drop after separation.

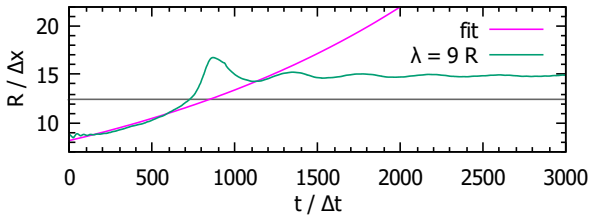


Figure S8: An example of the fitting for $\lambda = 9\,R$ with equation (S15).

For obtaining the growth rate, we fit an exponential of the form

$$R(t) = R_0\, e^{k\,t} \tag{S15}$$

to each curve in the range before separation of the cylinder where $R < 12.5\,m$, as indicated by the horizontal gray line in the plot. An example of one of the fits is shown in figure S8. Plotting the growth rates k for all λ values (figure S9) shows that an initial undulation with $\lambda < 2\,\pi$ is unstable. The maximum growth rate deviates a bit from the theoretical $\lambda_{\max}^{\text{theo}} \approx 9\,R$ at $\lambda_{\max}^{\text{sim}} = 10\,R$.

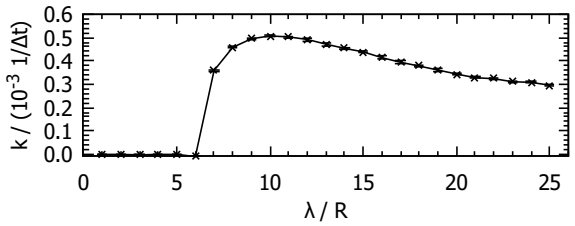


Figure S9: The growth rate k from equation (S15) plotted for different initial undulation wavelengths λ.

S2.2 Oblique Drop Impact

To validate our VoF-LBM solver, we recreate an oblique droplet impact setup [5] and compare our simulation with high-speed images of the experiment [8, 9].
The setup consists of a cubic simulation box filled with fluid in the bottom half with a small droplet initialized just above the surface with a velocity downward at an angle α. The droplet upon impact forms an asymmetric cavity and crown which breaks up into small droplets.
In order to stay as close to the experimental setup as possible, we choose these parameters:

- droplet diameter $d^{\mathrm{SI}} = 0.1\,\mathrm{mm}$
- fluid density $\rho^{\mathrm{SI}} = 1000\,\frac{\mathrm{kg}}{\mathrm{m}^3}$
- dynamic viscosity $\mu^{\mathrm{SI}} = 8.36 \cdot 10^{-4}\,\frac{\mathrm{kg}}{\mathrm{m\,s}}$
- surface tension coefficient $\sigma^{\mathrm{SI}} = 0.072\,\frac{\mathrm{kg}}{\mathrm{s}^2}$
- gravitational acceleration $g^{\mathrm{SI}} = 9.81\,\frac{\mathrm{m}}{\mathrm{s}^2}$
- height of the pool $h^{\mathrm{SI}} = 0.5\,\mathrm{mm}$
- cubic simulation box with side length $L^{\mathrm{SI}} = 1\,\mathrm{mm}$
- Weber number $We = 416.5$
- impact angle $\alpha = 28.5°$

These values result in an impact velocity of $u^{\mathrm{SI}} = 17.32\,\frac{\mathrm{m}}{\mathrm{s}}$ and a Reynolds number of $Re = 2071$. In simulation units, three independent parameters are chosen for conversion to simulation units,

- fluid density $\rho^{\mathrm{sim}} = 1$
- impact velocity $u^{\mathrm{sim}} = 0.05$
- simulation box side length $L^{\mathrm{sim}} = 384$

resulting in the remaining quantities in simulation units to be:

- kinematic shear viscosity $\nu^{\mathrm{sim}} = 9.27 \cdot 10^{-4}$
- surface tension $\sigma^{\mathrm{sim}} = 2.30 \cdot 10^{-4}$
- force per volume $f^{\mathrm{sim}} = 2.13 \cdot 10^{-10}$

The simulations are done with D3Q19 and the SRT operator.
The in figure S10 illustrated time frames are $t = \{0.46, 2.33, 8.22, 12.15, 18.9\} \cdot t_i$, whereby $t_i = \frac{d}{u}$, or in lattice units equivalent to

$$t^{\mathrm{sim}} = \{353, 1789, 6313, 9331, 14515\} \cdot \Delta t.$$

In the last few frames, artifacts in the cavity become visible, caused by several simulation parameters being close to the limit of their stable range.

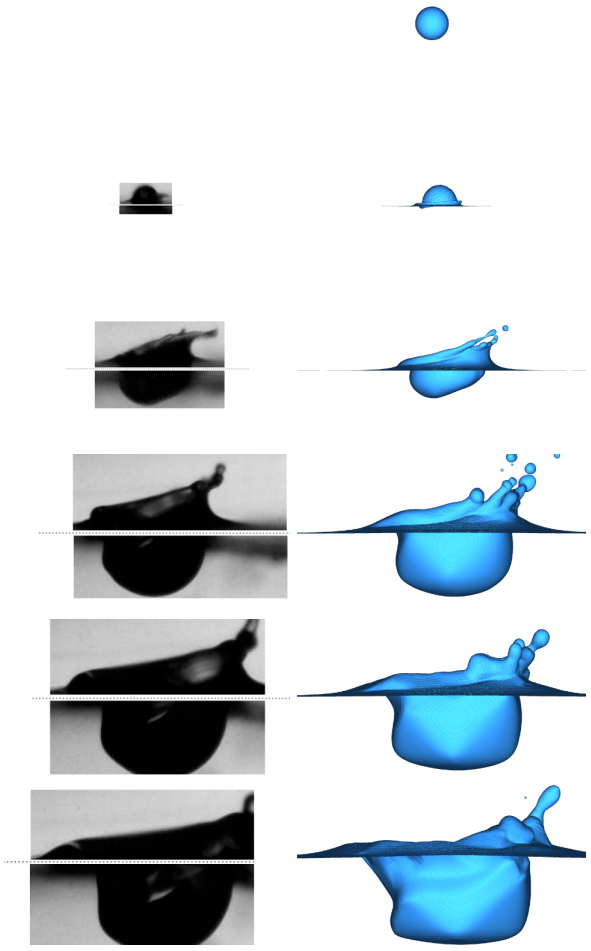


Figure S10: The setup from [9], figure 3 recreated in simulation. The images from the experiment had to be individually rescaled, because the length scale is not kept the same across images and no scale bars are present in the original image series. Here, the length scale is kept constant for all images and the diameter of the drop (top left) is $0.1\,mm$.

S2.3 Crown Formation by Drop Impact on a Shallow Pool

Here we validate our fluid solver on a crown formation setup [10, 11] that consists of a small drop impacting a shallow pool of fluid at high speed [5]. The simulations are done with D3Q19 and the SRT collision operator. For the drop radius only the range $D^{\mathrm{SI}} \in \{2.0, 4.2\}\, mm$ is given in [11], so in the simulation the arithmetic mean is chosen. The fluid in the experiment is not pure water but 70 % glycerol in water by weight. The full list of parameters is:

- drop diameter $d^{\mathrm{SI}} = 3.1\, mm$
- height of the pool $h^{\mathrm{SI}} = 0.5\, d^{\mathrm{SI}}$
- fluid density $\rho^{\mathrm{SI}} = 1177.9\, \frac{kg}{m^3}$, assuming $T^{\mathrm{SI}} \approx 25°C$ [12]
- surface tension coefficient $\sigma^{\mathrm{SI}} = 0.0661\, \frac{kg}{s^2}$, assuming $T^{\mathrm{SI}} \approx 25°C$ [13]
- gravitational acceleration $g^{\mathrm{SI}} = 9.81\, \frac{m}{s^2}$
- box dimensions $L^{\mathrm{SI}}_{x,y} = 30\, mm$, $L^{\mathrm{SI}}_{z} = 15\, mm$
- Reynolds number $Re = 1168$
- Weber number $We = 2010$

These values result in an impact velocity of $u^{\mathrm{SI}} = 6.03\, \frac{m}{s}$ and a kinematic shear viscosity of $\nu^{\mathrm{SI}} = 1.60 \cdot 10^{-5}\, \frac{m^2}{s}$. In simulation units, three independent parameters are chosen for the dimensionalization procedure,

- fluid density $\rho^{\mathrm{sim}} = 1$
- impact velocity $u^{\mathrm{sim}} = 0.05$
- simulation box size in x-direction $L^{\mathrm{sim}}_{x} = 496$

resulting in the remaining quantities in simulation units to be:

- kinematic shear viscosity $\nu^{\mathrm{sim}} = 2.19 \cdot 10^{-3}$
- surface tension $\sigma^{\mathrm{sim}} = 6.37 \cdot 10^{-5}$
- force per volume $f^{\mathrm{sim}} = 4.08 \cdot 10^{-8}$

The in figure S11 simulated time frames are $t^{\mathrm{SI}} = \{0.3, 1.0, 3.0, 7.5, 10.0\}\, ms$, whereby the starting point of the simulation is offset by $t^{\mathrm{SI}}_0 = 0.13\, ms$ to synchronize the first frame with the experiment. In lattice units these times are:

$$t^{\mathrm{sim}} = \{339, 1735, 5724, 14700, 19687\} \cdot \Delta t$$

The simulation shows good agreement with the experiment. In the simulation, the crown is more symmetric, appears a bit wider than in the experiment and later shows slight octagonal artifacts in its shape. The height of the crown is in good agreement with the experiment.
Possible discrepancies may be caused by deviations in the density and surface tension of the fluid as well as the sphere radius.

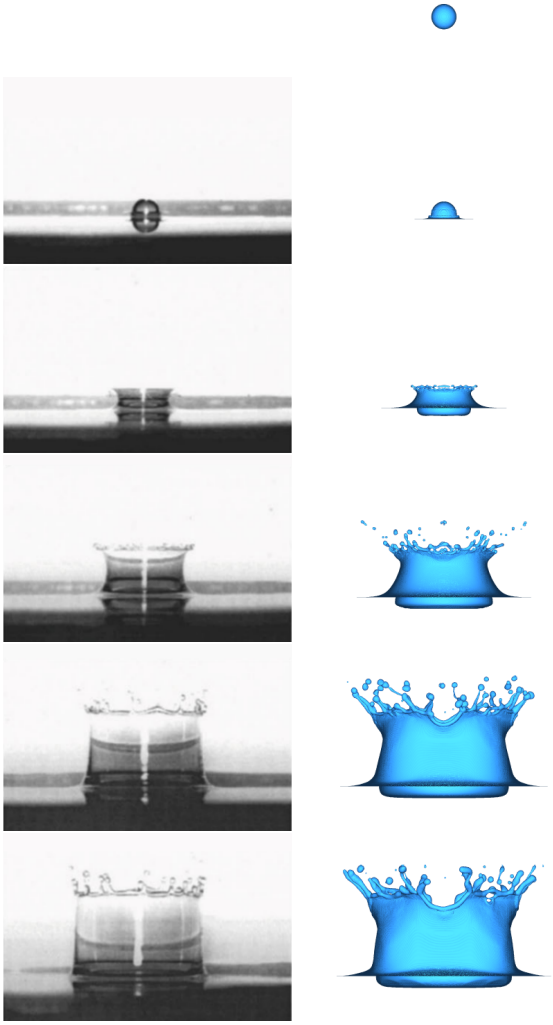
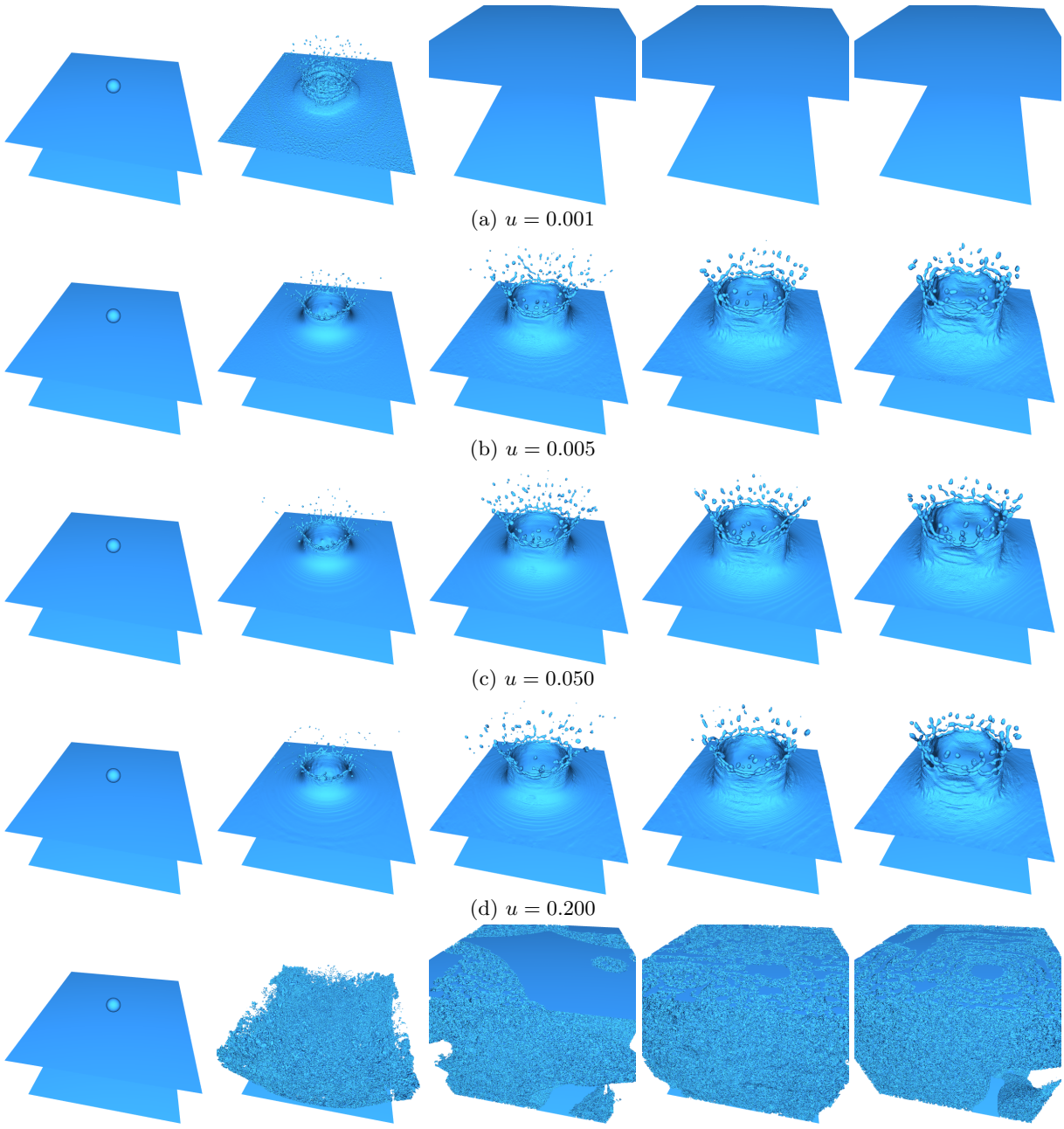


Figure S11: The simulation results are compared to the experiment from [10].

S3 Model Robustness

S3.1 Self-Test: Varying Impact Velocity u in Lattice Units



(a) $u = 0.001$

(b) $u = 0.005$

(c) $u = 0.050$

(d) $u = 0.200$

(e) $u = 0.300$

Figure S12: Visual comparison of the 4 mm diameter raindrop impact simulation without IBM particles for lattice resolution $L_x = 464$ for various impact velocities u in lattice units. Time stamps (left to right) are $t \in \{0.0, 2.5, 5.0, 7.5, 10.0\}$ ms.

S3.2 Influence of LBM Lattice Resolution on Droplet Distribution

The lattice resolution L_x can be varied within the limits of video memory capacity. Generally, physical effects should not depend on the L_x, given it is large enough to resolve the tiny droplets. Figures S19 and S20 give an overview on how different resolutions affect the simulation visually. Our

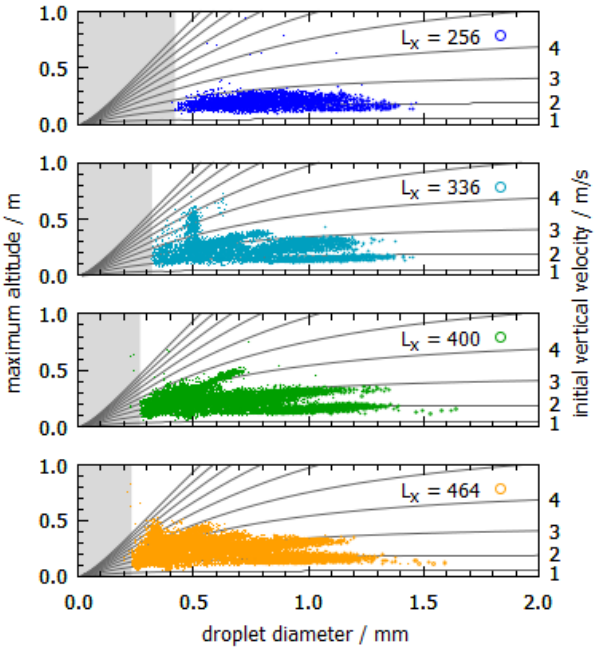


Figure S13: Diameter and maximum altitude of droplets after impact of a 4 mm diameter terminal velocity raindrop for various lattice resolutions. The diameter of the circles indicates the number of microplastic particles enclosed in the droplet.

IBM implementation restricts us to single-GPU simulations and our biggest GPUs (Radeon VII) have 16 GB video memory, limiting the resolution to $464 \times 464 \times 394$. However without particles, we can simulate on multiple GPUs (4x Radeon VII) by domain decomposition [14], enabling resolutions of up to $748 \times 748 \times 636$. The four simulation domains are indicated by thin lines in the illustrations.

We now examine the influence of lattice resolution on the droplet and particle distributions. We run the simulation for $d = 4\,\text{mm}$ and $\rho_p = 1.05\,\frac{\text{g}}{\text{cm}^3}$ 100 times for every lattice resolution, with the microplastic particles each time being initialized at different random positions. The maximum altitude depending on droplet diameter is illustrated in figure S13.
We see that $L_x = 256$ is insufficient for meaningful results. Lattice resolution limits the minimum size of resolved droplets; higher resolutions are able to resolve smaller droplets. The region of large droplets is covered by higher resolutions as well.
We create histograms, revealing the size distribution of droplets (figure S14), fluid volume (figure S15), how many particles are present in droplets depending on size (figure S16) as well as the particle concentration depending on droplet size (figure S17).

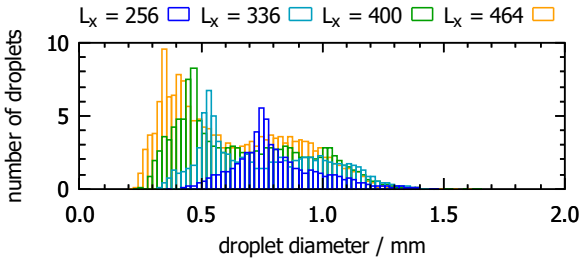


Figure S14: The size distribution of droplets depending on lattice resolution L_x.

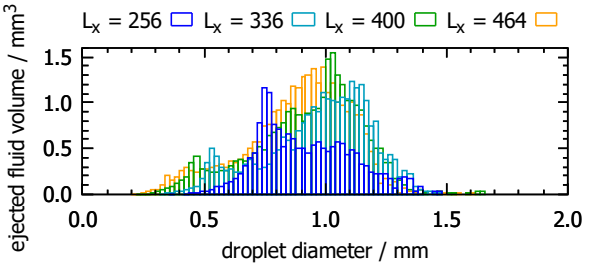


Figure S15: The distribution of ejected fluid volume by droplet diameter and lattice resolution L_x. The (too) coarse lattice resolution $L_x = 256$ cannot resolve small droplets, but the fluid volume must go somewhere, so it accumulates at the smallest possible resolvable droplets, leading to an artificial peak around $d \approx 0.75\,\text{mm}$.

For lattice resolution as low as $L_x = 256$, small droplets cannot be resolved, creating an artificial peak at $d = 0.75\,\text{mm}$ in figure S15 as a numerical artifact.

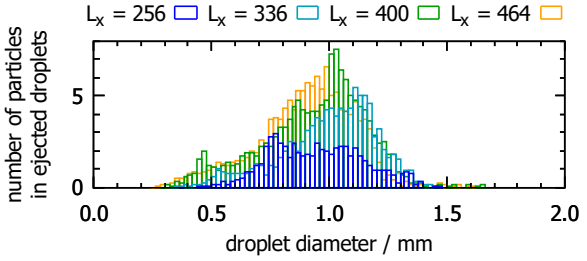


Figure S16: The distribution of microplastic particles in droplets depending on droplet diameter looks suspiciously similar to the distribution of ejected fluid volume depending on droplet diameter in figure S15.

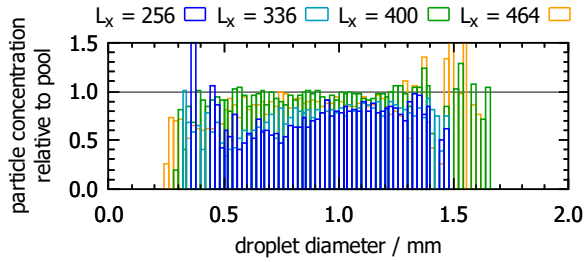
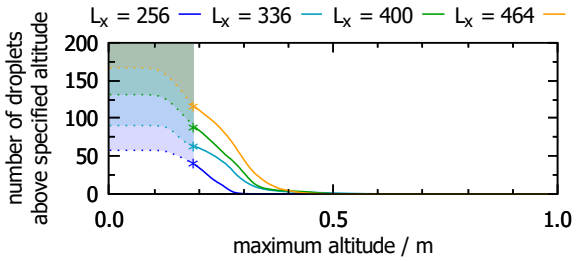
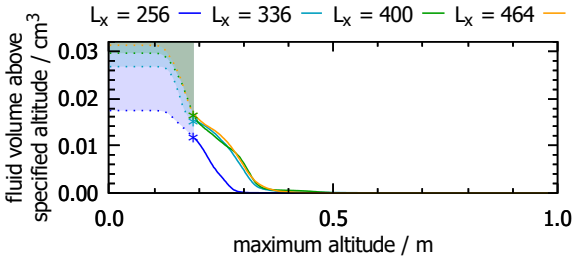


Figure S17: The particle concentration in ejected droplets depending on droplet size.

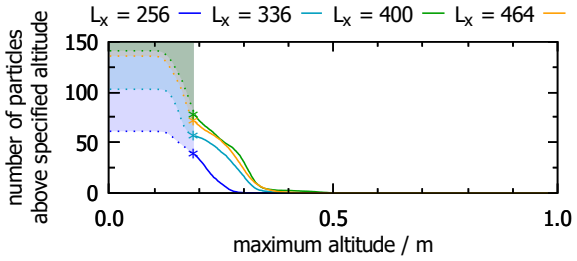
We then determine the amount of droplets (figure S18 (a)), fluid (S18 (b)) and particles in droplets (S18 (c)) ejected above a certain altitude. For higher resolution, the number of droplets is generally larger as more small droplets are resolved. The ejected fluid volume and number of particles in ejected droplets however approach a plateau for increasing resolution. This indicates that the resolution of $L_x = 464$ is sufficient to model particle transport.



(a)

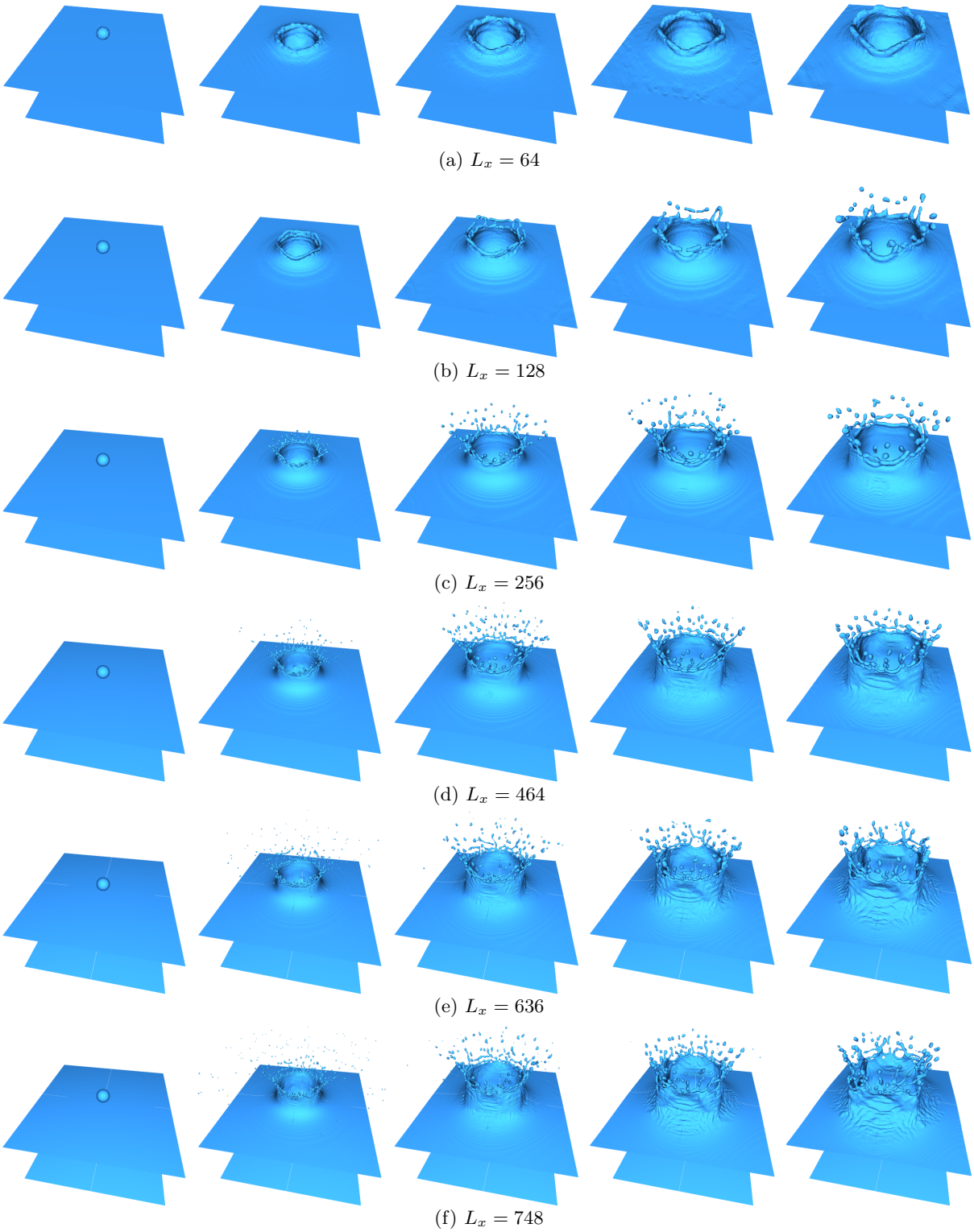


(b)



(c)

Figure S18: (a) Number of droplets, (b) fluid volume and (c) number of particles in droplets ejected above a specified altitude depending on lattice resolution. All other parameters (that determine the physics of the system) are kept the same across simulations. For altitudes lower than h_{cut} (asterisks), the distributions are considered the lower bound (dashed lines) and the upper bound is undefined as indicated by the shaded areas (see section 2.3.2).



(a) $L_x = 64$

(b) $L_x = 128$

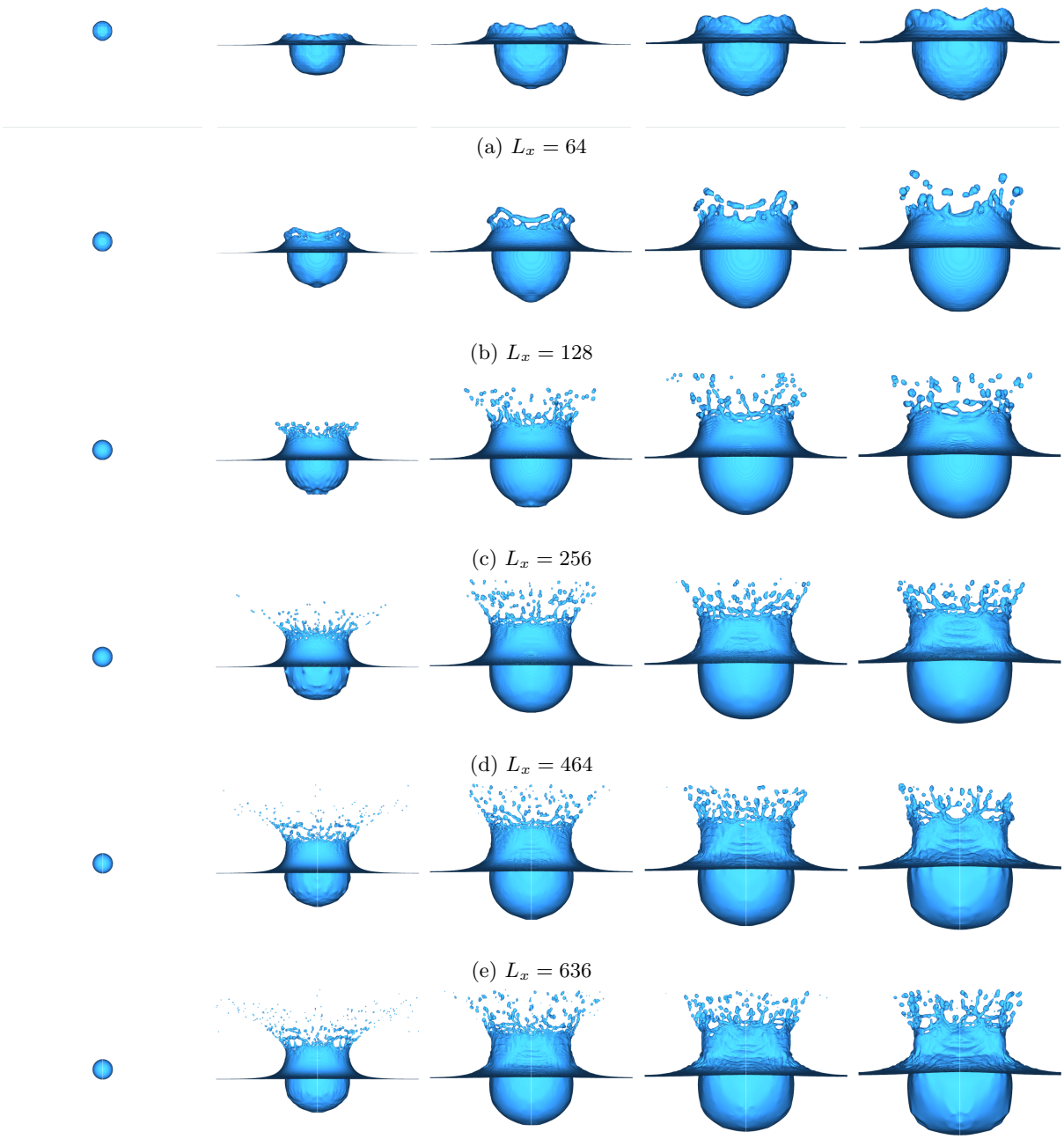
(c) $L_x = 256$

(d) $L_x = 464$

(e) $L_x = 636$

(f) $L_x = 748$

Figure S19: Visual comparison of the 4 mm diameter raindrop impact simulation without IBM particles for various lattice resolutions L_x in perspective view. Time stamps (left to right) are $t \in \{0.0, 2.5, 5.0, 7.5, 10.0\}$ ms. We provide subfigure (f) animated as an additional video file (figure-S19f.mp4) in the supplementary files.



(a) $L_x = 64$

(b) $L_x = 128$

(c) $L_x = 256$

(d) $L_x = 464$

(e) $L_x = 636$

(f) $L_x = 748$

Figure S20: Visual comparison of the 4 mm diameter raindrop impact simulation without IBM particles for various lattice resolutions L_x in orthogonal view. Time stamps (left to right) are $t \in \{0.0, 2.5, 5.0, 7.5, 10.0\}$ ms.

S4 Additional Data

S4.1 Simulation Unit Parameters for Section 3.1

	SI-units	LBM units
drop diameter d	4.1 mm	74.8
impact velocity u	7.2 $\frac{\mathrm{m}}{s}$	0.05
box dimensions $L_{x,y,z}$	10 d, 10 d, 8.5 d	748, 748, 636
pool height h	4 d	299.2
kin. shear viscosity ν	$1.0 \cdot 10^{-6}$ $\frac{\mathrm{m}^2}{\mathrm{s}}$	$1.267 \cdot 10^{-4}$
water density ρ	1018.3 $\frac{\mathrm{kg}}{\mathrm{m}^3}$	1
surface tension σ	0.073 $\frac{kg}{s^2}$	$6.307 \cdot 10^{-5}$
grav. acceleration g	9.81 $\frac{\mathrm{m}}{\mathrm{s}^2}$	$2.593 \cdot 10^{-8}$

Table S2: The simulation parameters for the simulation in section 3.1 before and after unit conversion.

S4.2 Figure 7 in Separate Figures

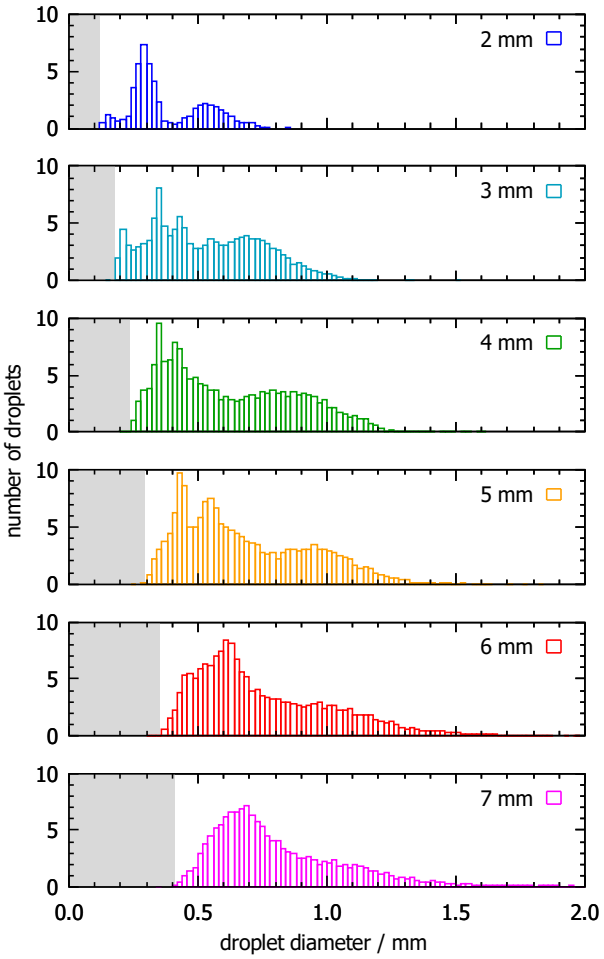


Figure S21: Figure 7 (a) extended as separate figures for each raindrop diameter with the non-resolvable diameter interval marked in gray.

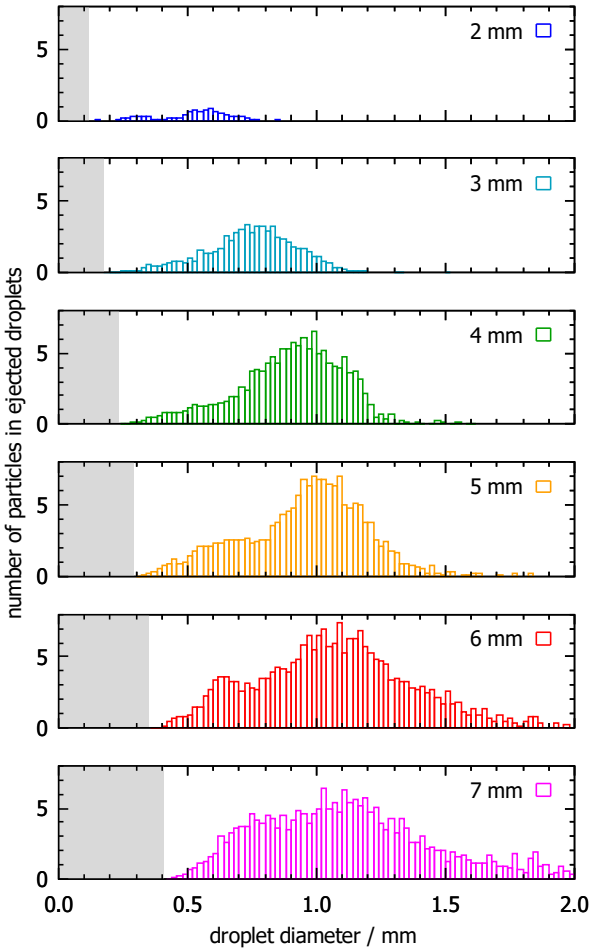


Figure S22: Figure 7 (b) extended as separate figures for each raindrop diameter with the non-resolvable diameter interval marked in gray.

S4.3 Velocity Inclination of Ejected Droplets for 4 mm Diameter Raindrop Impacts

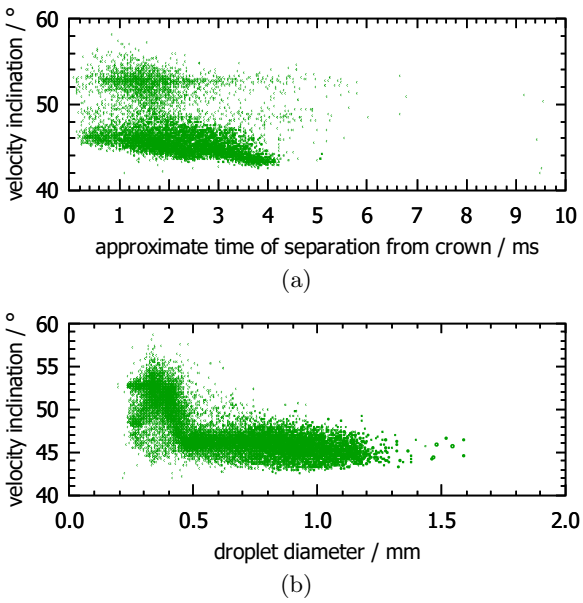


Figure S23: The velocity inclination from the vertical axis of ejected droplets (a) by the approximate time the droplets separate from the crown rim and (b) by droplet diameter for 100 4 mm diameter raindrop impacts. The initially ejected, smaller droplets are inclined more compared to later ejected, larger droplets.

S4.4 Figure 14 (b) for Different Vertical Updraft Velocity

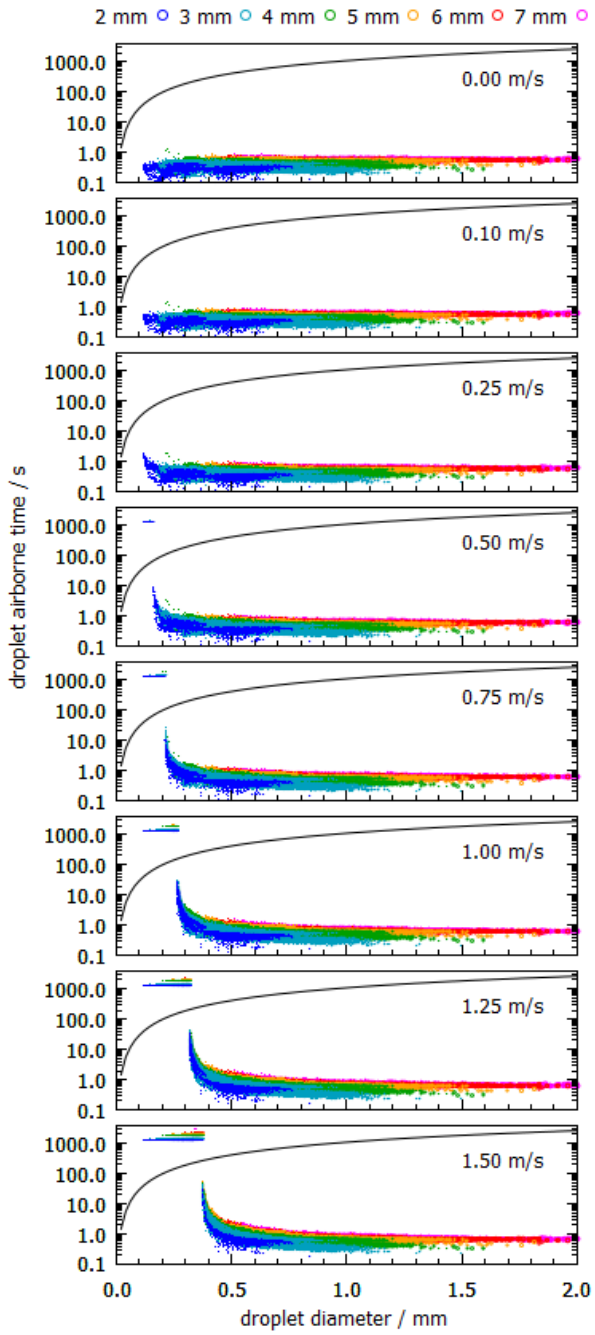


Figure S24: Figure 14 (b) for different vertical updraft velocity. The black curve represents the lifetime of airborne droplets due to evaporation (eq. (11)). Droplets with diverging airborne time have the values capped to be visible in the diagrams.

S4.5 Data table for figure 14 (c)

$u_{\text{updraft}}/\frac{\text{m}}{\text{s}}$	2 mm	3 mm	4 mm	5 mm	6 mm	7 mm
0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.10	0.00	0.00	0.00	0.00	0.00	0.00
0.25	0.00	0.00	0.00	0.00	0.00	0.00
0.50	0.01	0.00	0.00	0.00	0.00	0.00
0.75	0.01	0.04	0.00	0.00	0.00	0.00
1.00	0.27	0.17	0.08	0.00	0.00	0.00
1.25	0.98	0.50	0.55	0.04	0.00	0.00
1.50	1.43	1.78	1.75	0.68	0.04	0.00
2.00	2.30	5.56	6.37	5.56	2.97	0.67

Table S3: Data for figure 14 (c): The number of microplastic particles per raindrop of 2 mm to 7 mm diameter in spray droplets with airborne time longer than the evaporation lifetime. These particles are considered picked up by the atmosphere. Values are given for a concentration of 5000 microplastic particles per cm^3 in sea water.

S4.6 Figure 15 (b) with $V_{\text{pre,total}}$

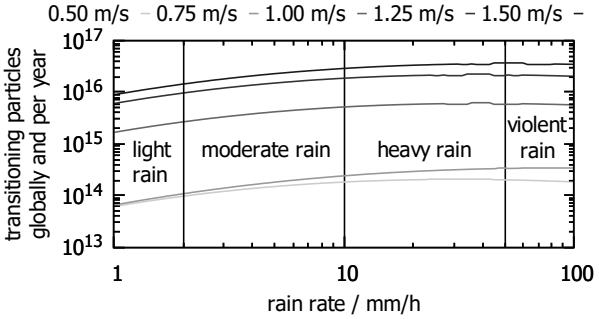


Figure S25: Figure 15 (b) with $V_{\text{pre,total}} = 3.82 \cdot 10^{14}\,\text{m}^3$ set as a fixed parameter. The curves in different shades of gray represent different vertical updraft velocity. If the rain rate was one fixed value everywhere on the globe at all time, then the curves in this figure would be the global estimate for transitioning particles. In nature however, rain rate follows an exponential distribution as modeled by Rice-Holmberg. We take this rain rate distribution into account in the global estimate in the manuscript.

S4.7 Influence of Microplastic Density

Here we examine the influence of particles with different material density on impact dynamics. The density variation for plastics ranges from 920 $\frac{\text{kg}}{\text{m}^3}$ (polypropylene) to 2170 $\frac{\text{kg}}{\text{m}^3}$ (polytetrafluoroethylene). To estimate the effect of gravity and particle material density, we calculate how far a microplastic particle would sediment during our 10 ms simulated time frame using eq. (S3). With the plastic type of highest density (2170 $\frac{\text{kg}}{\text{m}^3}$), we get a maximum sedimentation velocity of

$$u_z = \left(\frac{\rho_p}{\rho} - 1\right) \frac{d^2\, g}{18\, \nu} \approx 0.0067\, \frac{\text{m}}{\text{s}} \qquad \text{(S16)}$$

so during 10 ms the particle travels a distance of 0.067 mm or 0.62 lattice points. Whilst this effect is small, it could have some impact on the results, so we simulate four different material densities just to make sure.
In the histograms revealing the size distribution of droplets and how many particles are present in droplets depending on their diameter (figure S26), we see no significant differences in the data sets.

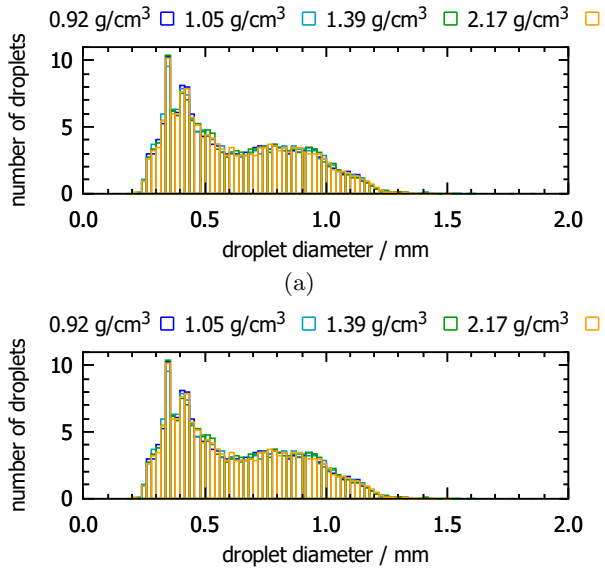


Figure S26: (a) The size distribution of droplets and (b) the distribution of microplastic particles in droplets depending on particle material density ρ_p.

The height distribution of ejected droplets and particles are plotted in figure S27.
During the short duration of the initial phase of the impact, we do not observe any significant differences in both impact dynamics and particle advection for different particle material densities. The particles are entirely passively advected with the fluid. This means

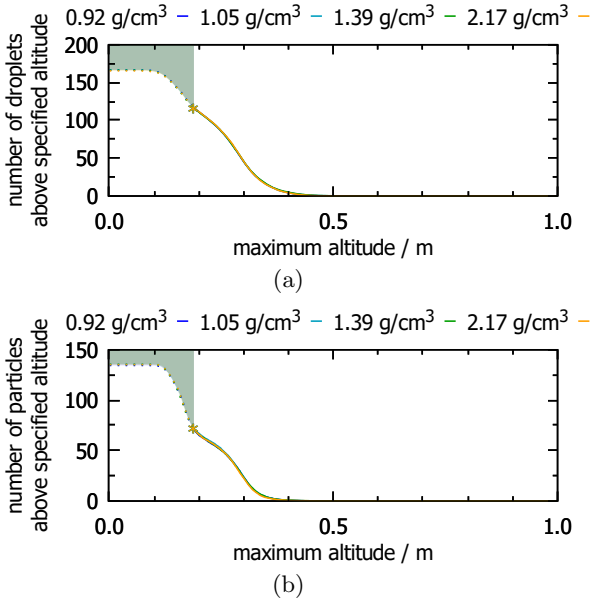
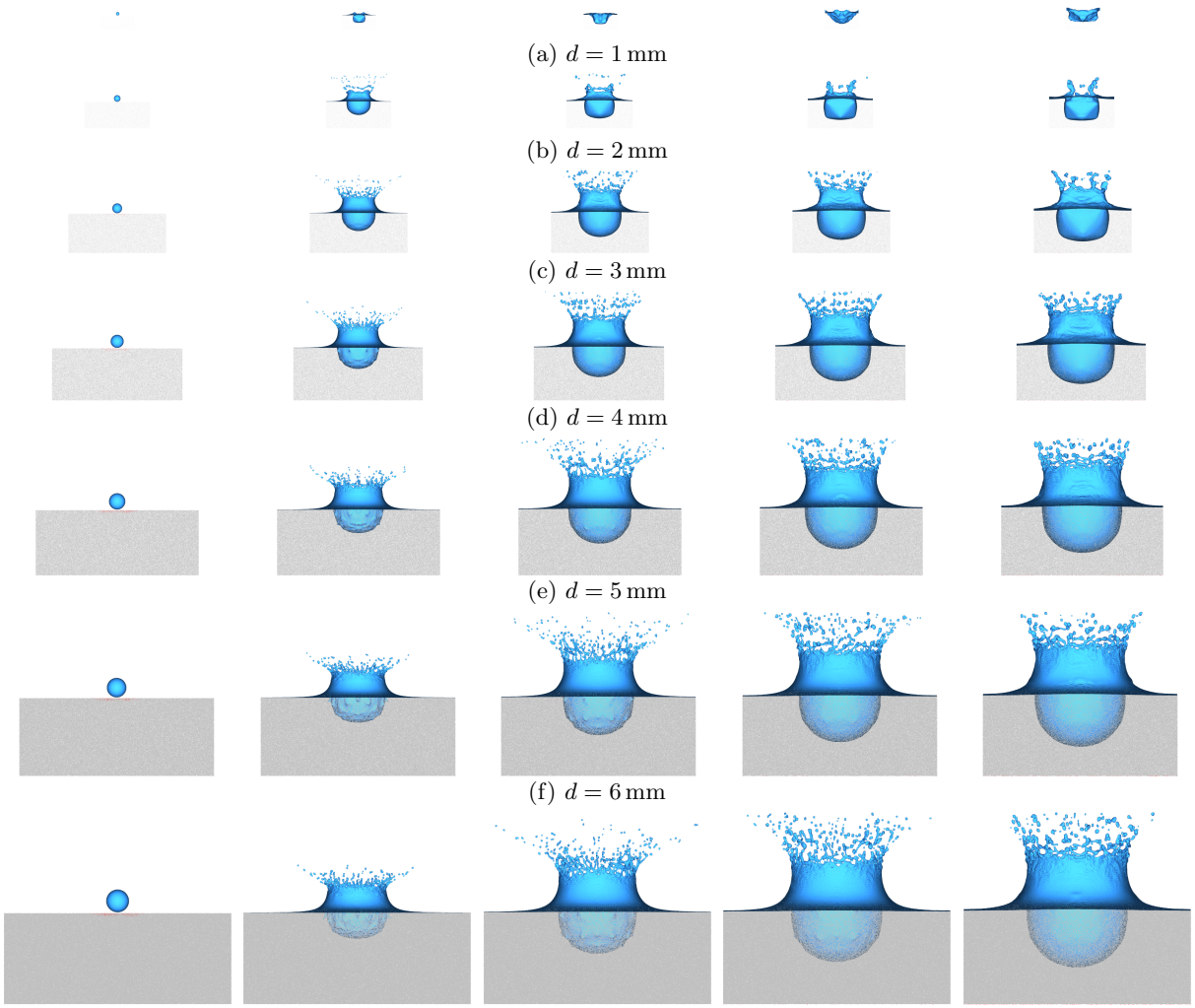


Figure S27: (a) Number of droplets and (b) particles in droplets ejected above a specified altitude for various microplastic particle densities. For altitudes lower than h_{cut} (asterisks), the distributions are considered the lower bound (dashed lines) and the upper bound is undefined as indicated by the shaded areas (see section 2.3.2).

that as long as the particles are small enough, their properties have no influence on the transport behavior during raindrop impacts.
However in the long term, particles less dense than water tend to accumulate at the water surface, making them more likely to cross the interface than particles more dense than water that sediment to greater depths. Only particles directly under the water surface can be ejected during rainfall (see the region where ejected particles originate in section 3.3.4).

S4.8 Visualization of Simulations

S4.8.1 Simulations of Raindrops with Various Diameters



(a) $d = 1\,\text{mm}$

(b) $d = 2\,\text{mm}$

(c) $d = 3\,\text{mm}$

(d) $d = 4\,\text{mm}$

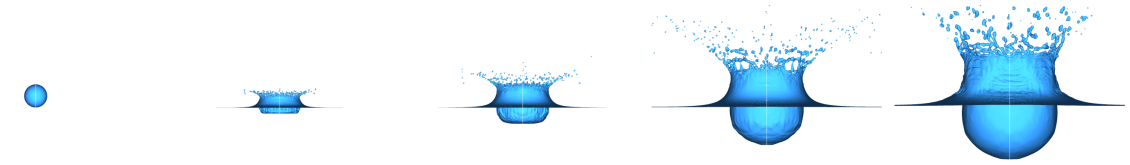
(e) $d = 5\,\text{mm}$

(f) $d = 6\,\text{mm}$

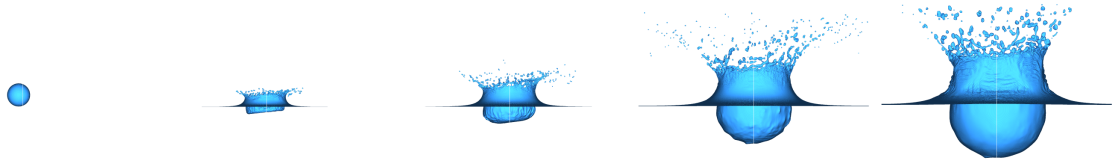
(g) $d = 7\,\text{mm}$

Figure S28: Visual comparison of the impacts of differently sized raindrops. The simulation box size is $L_x = 464$ and illustrated time stamps (left to right) are $t \in \{0.0, 2.5, 5.0, 7.5, 10.0\}$ ms. For smaller raindrops, the simulation box is scaled down such that the box width is always 10 times the raindrop diameter. Ejected particles are marked in red.

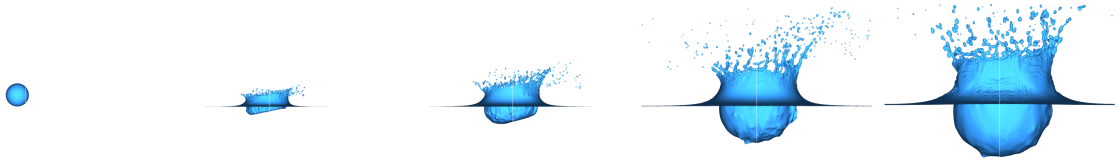
S4.8.2 Oblique Impacts



(a) $\alpha = 0°$



(b) $\alpha = 10°$

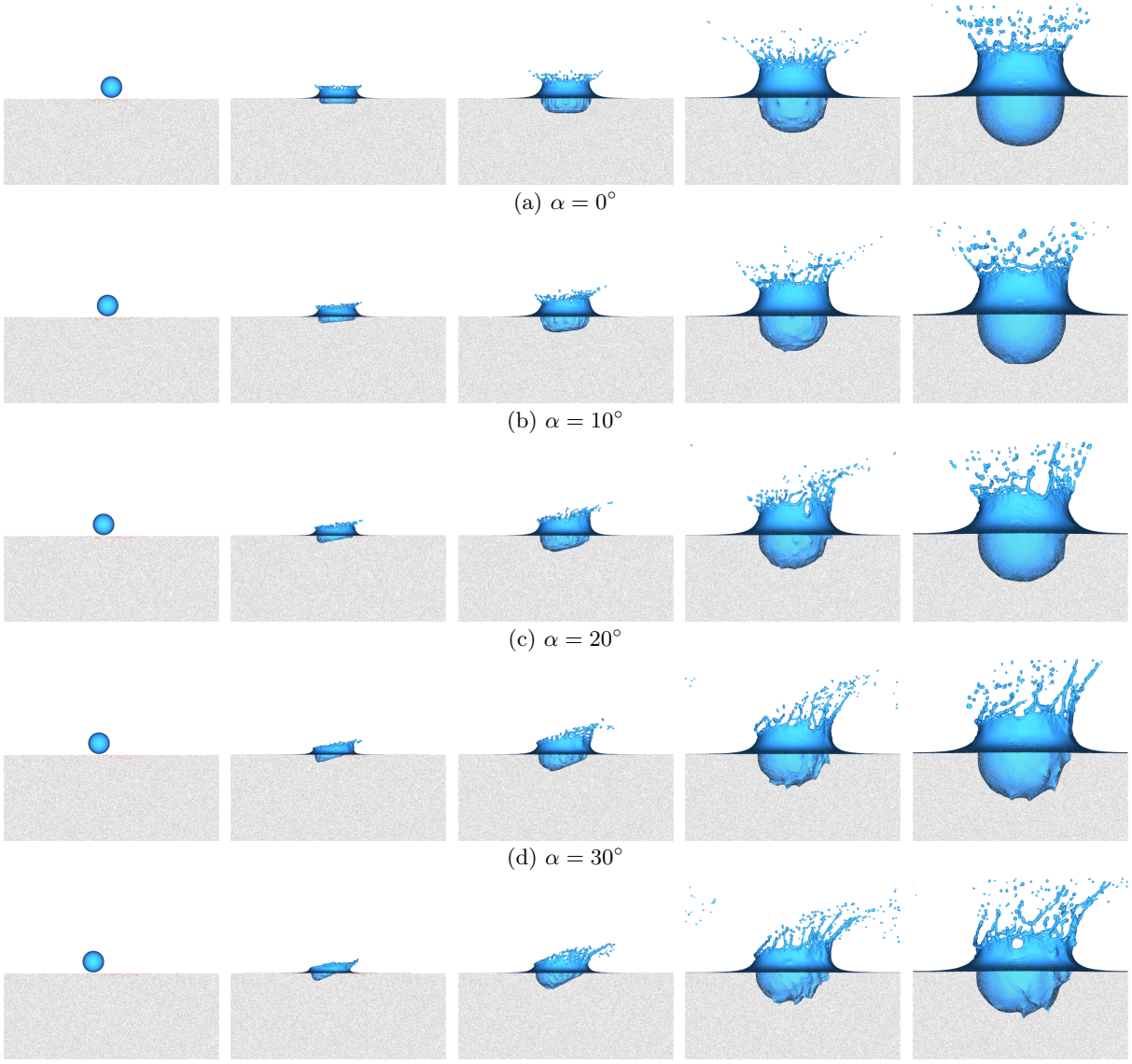


(c) $\alpha = 20°$

(d) $\alpha = 30°$

(e) $\alpha = 40°$

Figure S29: Visual comparison of the 4 mm diameter raindrop impact simulation without microplastic particles for various impact angles α. Time stamps (left to right) are $t \in \{0.0, 0.5, 1.0, 2.5, 5.0\}$ ms and lattice resolution is $L_x = 748$.



(a) $\alpha = 0°$

(b) $\alpha = 10°$

(c) $\alpha = 20°$

(d) $\alpha = 30°$

(e) $\alpha = 40°$

Figure S30: Visual comparison of the 4 mm diameter raindrop impact simulation with microplastic particles for various impact angles α. Time stamps (left to right) are $t \in \{0.0, 0.5, 1.0, 2.5, 5.0\}$ ms and lattice resolution is $L_x = 464$. Ejected particles are marked in red.

S5 References

[1] Zhi-Gang Feng and Efstathios E Michaelides. "Drag coefficients of viscous spheres at intermediate and high Reynolds numbers". In: *J. Fluids Eng.* 123.4 (2001), pp. 841–849.

[2] Faith A Morrison. "Data correlation for drag coefficient for sphere". In: *Department of Chemical Engineering, Michigan Technological University, Houghton, MI* 49931 (2013).

[3] Faith A Morrison. *An introduction to fluid mechanics*. New York, USA: Cambridge University Press, 2013.

[4] HJ Holterman. *Kinetics and evaporation of water drops in air*. Vol. 2012. Wageningen, Netherlands: Citeseer, 2003.

[5] Moritz Lehmann. *High Performance Free Surface LBM on GPUs*. 2019. URL: https://epub.uni-bayreuth.de/5400/.

[6] Moritz Lehmann and Stephan Gekle. "Analytic Solution to the Piecewise Linear Interface Construction Problem and its Application in Curvature Calculation for Volume-of-Fluid Simulation Codes". In: *arXiv preprint arXiv:2006.12838* (2020).

[7] Oren Breslouer. "Rayleigh-Plateau Instability: Falling Jet". In: *Project Report* (2010).

[8] Marise V Gielen et al. "Oblique drop impact onto a deep liquid pool". In: *Physical review fluids* 2.8 (2017), p. 083602.

[9] Sten A Reijers et al. "Oblique droplet impact onto a deep liquid pool". In: *arXiv preprint arXiv:1903.08978* (2019).

[10] An-Bang Wang and Chi-Chang Chen. "Splashing impact of a single drop onto very thin liquid films". In: *Physics of fluids* 12.9 (2000), pp. 2155–2158.

[11] Gangtao Liang et al. "Crown behavior and bubble entrainment during a drop impact on a liquid film". In: *Theoretical and Computational Fluid Dynamics* 28.2 (2014), pp. 159–170.

[12] Kähler Nian-Sheng Cheng Volk. *Calculate density and viscosity of glycerol/water mixtures*. 2018. URL: http://www.met.reading.ac.uk/~sws04cdw/viscosity_calc.html (visited on 11/23/2019).

[13] Glycerine Producers' Association et al. *Physical properties of glycerine and its solutions*. Washington, D.C., USA: Glycerine Producers' Association, 1963.

[14] Fabian Häusl. *MPI-based multi-GPU extension of the Lattice Boltzmann Method*. 2019. URL: https://epub.uni-bayreuth.de/5689/.

8.2 Publication 2

Modeling of vertical microplastic transport by rising bubbles

by

Moritz Lehmann, Fabian Häusl, and Stephan Gekle

Microplastics and Nanoplastics 3.4 (2023), pp. 1–6
https://doi.org/10.1186/s43591-023-00053-7

Modeling of vertical microplastic transport by rising bubbles

Moritz Lehmann[1*], Fabian Häusl[1] and Stephan Gekle[1]

February 11, 2023

*Correspondence: moritz.lehmann@uni-bayreuth.de
[1]Biofluid Simulation and Modeling – Theoretische Physik VI, University of Bayreuth

Abstract

Microplastic particle concentration at the sea surface is critical for quantifying microplastic transport across the water-air interface. Previous studies suggest that the concentration at the sea surface is enhanced compared to bulk concentration, yet little is known about the detailed mechanisms behind this enhancement. In this work, we model one particular process in simulation that may contribute to this enhanced surface concentration: bubble scavenging. Using lattice-Boltzmann Volume-of-Fluid simulations, we find that rising bubbles indeed generate a net flow of particles toward the surface. The efficiency of the process, however, highly depends on the microplastic particle surface properties. Clean, hydrophobic particles adhere much better to the bubble surface and are therefore transported significantly better than weathered, hydrophilic particles that are only entrained in the flow around a bubble.

Keywords: microplastics; bubbles; sea surface; lattice Boltzmann method; Volume-of-Fluid; GPU; OpenCL

1 Introduction

Waves on the ocean surface create myriads of air bubbles [1] that rise to the surface and burst. During rise, bubbles can interact with particles suspended in the water [2–7] and enrich particle concentration prior to bubble burst by bubble scavenging [8–12]. At burst, bubbles eject fine water droplets into the air, either in the form of film droplets or jet droplets of various size depending on bubble diameter [13–21]. This process is associated with aerosol production [10, 22–25], including bacteria [26, 27] and organic compounds [28]. Besides these, bursting can also lead to the ejection of microplastic particles into the air [29–33] similar to microplastic ejection by impacting raindrops [34].

Knowing the concentration of microplastic particles in the sea surface microlayer (SML) is key for estimating environmental relevance of this water-air transport. Experimental studies show large local variations in marine microplastic particle concentration [35–38], and find that concentration at the SML is largely enhanced compared to bulk concentration [39–41], yet don't identify the mechanisms leading to this difference in concentration between SML and bulk.

In this work we investigate vertical microplastic transport in the water column with the bubble scavenging mechanism by using computer simulations. Specifically we aim to understand the impact of particle wetting properties on transport efficiency.

2 Methods

2.1 Volume-of-Fluid Lattice Boltzmann Method

In this work we use the Volume-of-Fluid (VoF) lattice Boltzmann method (LBM) implementation *FluidX3D* [34, 42–47] that has been extended to simulate rising bubbles with Hoshen-Kopelman [48] volume tracking and the ideal gas law

$$p\,V = n\,R\,T = \text{const.} \tag{1}$$

The method is thoroughly validated in [34, 44–47, 49]. VoF provides three classes of Cartesian grid points – fluid, interface and gas. The fluid phase is simulated with regular LBM, the interphase is kept sharp at a thickness of one lattice cell and handles surface tension, and the gas phase is not simulated and treated as vacuum. To accommodate for bubbles, all separate gas domains are tracked with a Hoshen-Kopelman approach, computing their volume and pressure. Since our simulations are isothermal, the product $p\,V$ must remain constant, which is ensured by modifying density in reconstructed gas equilibrium populations in the VoF-LBM model. Special consideration is given to events when a bubble splits in two or more smaller bubbles or when two or more bubbles merge, for which trigger events are detected [47].

2.2 Immersed-boundary method

Microplastic particles are modeled by the immersed-boundary method (IBM) as in [34, 42]. A single IBM point-particle (with an effective hydrodynamic diameter of the lattice constant) corresponds to one microplastic particle. These IBM particles are neutrally buoyant and do not interact with each other, reflecting the natural situation where the microplastic concentration is expected to be rather low (between 1 to 7 particles per liter [35, 50]). Neutral buoyancy simplifies the IBM to one-way-coupling, meaning particles are only passively advected by the velocity field and interface forces and do not exert forces back on the fluid. In addition, the non-interaction allows us to simulate high particle concentrations for statistically meaningful results and then linearly scale down concentrations to environmental estimates. Unlike in an experiment, where such a large concentration would significantly increase Einstein viscosity, there is no change in viscosity in the simulation as the particles are modeled as point-particles rather than spheres.

For this study, the interaction of the particles with the water surface is critical. We consider two scenarios:

1. Non-sticky particles: Particles are only prevented from leaving the water phase with a repelling hard potential as in [34].

2. Sticky particles: Particles are prevented from leaving the water phase with a repelling hard potential as in [34], but additionally, once entering the direct vicinity of the water surface (distance of one lattice cell or less), a second attracting hard potential locks them onto the surface.

A technical difficulty in both situations is that the exact surface position in VoF-LBM is unknown. Hence the approach is to apply a repelling force if during trilinear velocity interpolation for a particle on the grid, at least one of the eight grid points is gas. The force is applied by replacing the unknown velocity of the gas point with the lattice speed of sound ($\frac{1}{\sqrt{3}}$ grid cells per time step, the fastest velocity possible in LBM units) in direction opposed to the local surface normal approximation. In case of a sticky surface, an attractive force is applied if at least one of the eight points is interface. This is done by adding the lattice speed of sound in the direction of the local surface normal approximation to the fluid velocity at the interface point.

In nature, other electrostatic interactions between particles bubble vortex also play a role [51], which are neglected by our simplified model. Further, the shear forces in the bubble vortex may separate particle aggregates and enhance particle fragmentation, but these effects are also not taken into account, as we only study non-interacting, single particles.

2.3 Simulation parameters

All simulations are carried out with these parameters for water: kinematic shear viscosity $\nu = 1.0 \cdot 10^{-6}\,\frac{\mathrm{m}^2}{\mathrm{s}}$, density $\rho = 1000\,\frac{\mathrm{kg}}{\mathrm{m}^3}$, surface tension $\sigma = 0.072\,\frac{\mathrm{kg}}{\mathrm{s}^2}$, gravitational acceleration $g = 9.81\,\frac{\mathrm{m}}{\mathrm{s}^2}$. To eliminate one possible complication in the model, the microplastic particles have neutral buoyancy with a density of $\rho_p = 1000\,\frac{\mathrm{kg}}{\mathrm{m}^3}$.

With a resulting Bond number of $Bo = 2.18$ and Morton number $Mo = 2.63 \cdot 10^{-11}$, the expected bubble behavior is between "spherical" and "wobbling" [52]. This behavior is matched by our simulations (figures 1 and 3).

3 Results and discussion

Rising bubbles in a water column can pick up particles in a process known as bubble scavenging [8–12]. Other works have already found that the microplastic concentration at the water surface is enriched [39–41], yet the mechanism for this enrichment is not identified. This suggests that bubble scavenging may apply to microplastic particles as well. We quantify this on a model system with computer simulations.

It is expected that weathered particles in nature stick less to the water surface due to their increased hydropilicity [7, 53–55]. Thus we separately investigate the transport of weathered, non-sticky particles, and clean, sticky particles.

The simulation box geometry is $1.6\,\mathrm{cm} \times 1.6\,\mathrm{cm} \times 12.8\,\mathrm{cm}$ (figures 1 and 3) and the boundaries in horizontal directions are periodic. One bubble with diameter $d_b = 4\,\mathrm{mm}$ is initially placed at $z_{\mathrm{initial}} = d_b$ and the simulation is terminated once the bubble reaches the position $z_{\mathrm{final}} = 32\,d_b - d_b$, so the bubble travels a total distance of $h_b = 12\,\mathrm{cm}$. The microplastic particle concentration is set to $C = 10^5$ particles per cm^3. The total particle count is about 3269298 with slight variation depending on lattice resolution in the simulation. A concentration this high allows to obtain accurate particle counts with just a single simulation. In simulation units, the bubble diameter is set to $d_b^{\mathrm{sim}} = 74$. This corresponds to the maximum box size allowed by the 40GB GPU memory on the Nvidia A100 which is used in the present simulations.

The simulated bubble travels the distance of $h_b = 0.12\,\mathrm{m}$ in $t = 0.88\,\mathrm{s}$, resulting in an average velocity of $136\,\frac{\mathrm{mm}}{\mathrm{s}}$ (equivalent to Reynolds number $Re = 545$), less than the experimental value of approximately $200\,\frac{\mathrm{mm}}{\mathrm{s}}$ [56]. This is expected, because in the simulation, the

bubble starts with zero velocity and the flow needs to accelerate first. The behavior of a mostly spherical, wobbling bubble matches experimental findings [52].

3.1 Transport of non-sticky particles

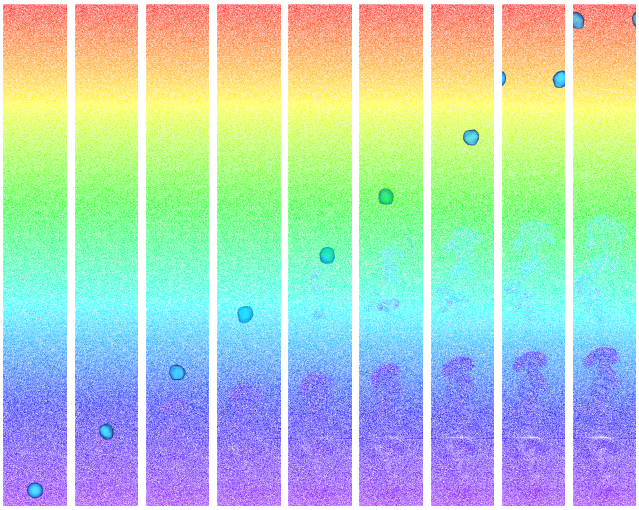


Figure 1: Illustration of the simulation for the rising bubble (particles do not stick to the bubble). The bubble ascends in a spiral as a result of non-laminar flow. Particles are colored by initial z-position. Images are not in uniform time intervals, but in uniform intervals of traveled distance. This figure is provided as a video in the supplementary files.

Figure 1 shows the simulation of the rising bubble where particles do not stick to the water surface. At first the bubble rises straight, but after the initial acceleration phase it pursues a spiraling trajectory, visible when it goes behind the slice of visualized particles and then comes back to front. This rotational behavior is consistent with experimental observations [51, 57]. At the end of the image series, the bubble passes half-way through the lateral periodic boundaries. Particles are colored by their initial z-position to be able to see where particles move in the vertical direction. A plume of particles is clearly visible in the lower half of the column, where the bubble still has been traveling in the plane of visualized particles.

In perfectly laminar flow and in the absence of additional effects beyond hydrodynamics, net particle transport through entrainment – particles dragged up by the flow around the bubble – would be impossible due to symmetry of the flow. Only when leaving the laminar regime, this transport mode is possible. Observations indicate that hydrophobic particles then may drop into the sub-bubble vortex [57]. The 4 mm diameter bubble is well outside the laminar flow regime at $Re = 545$, reflected in the clearly asymmetric distribution of traveled vertical particle distance in figure 2.

When taking the average of the traveled vertical distance for all particles and dividing by the traveled vertical distance of the bubble, $h_{avg,rel} = \frac{1}{h_b}\frac{1}{N}\sum_{i=1}^{N} h_i = -8.56 \cdot 10^{-4}$, with $N = 3269298$, the value is negative. This is expected, because the bubble volume, devoid of particles, starts at the bottom and moves to the top, so the fluid containing the particles has a net downward movement. To compute the net particle movement, the number of particles expected in the bubble volume $N_b = C \cdot \frac{\pi}{6} d_b^3 = 3351$, times the distance traveled by the bubble $h_b = 0.12\,\mathrm{m}$ is added:

$$h_{avg,rel,net} = \frac{1}{h_b}\frac{1}{N+N_b}\left(N_b\,h_b + \sum_{i=1}^{N} h_i\right) = \quad (2)$$

$$= \frac{1}{h_b}\frac{1}{N+N_b}\left(N_b\,h_b + h_{avg,rel}\,N\,h_b\right) = \quad (3)$$

$$= \frac{1}{N+N_b}\left(N_b + h_{avg,rel}\,N\right) = \quad (4)$$

$$= +1.69 \cdot 10^{-4} \quad (5)$$

So overall the net movement almost cancels out.

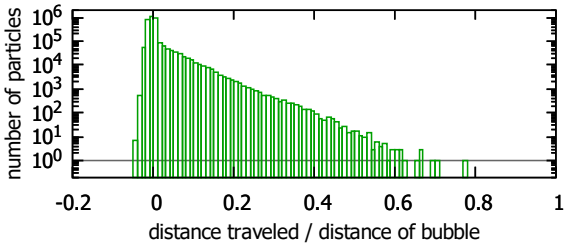
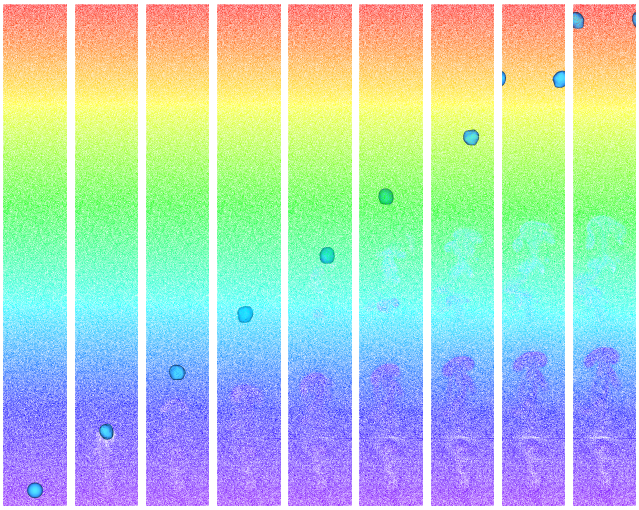


Figure 2: The distribution of the vertical travel distance of microplastic particles relative to the travel distance of the bubble. In this simulation, particles do not stick to the bubble. The vast majority of values are around 0 (logarithmic scale), but the distribution clearly is asymmetric towards positive distances, indicating particle entrainment as a consequence of non-laminar flow.

3.2 Transport of sticky particles

A possibly very efficient mechanism for particle transport is sticking of particles to the water-air interface of the bubble thus dragging the particles along with the rising bubble as shown in figure 3. Indeed, in simulations with sticking particles, the distribution of traveled vertical distance in figure 4 shows that the bubble picks up



0ms 116ms 229ms 334ms 443ms 548ms 670ms 774ms 880ms

Figure 3: Illustration of the simulation for the rising bubble (particles stick to the bubble). Particles are colored by initial z-position. Images are not in uniform time intervals, but in uniform intervals of traveled distance. The rising bubble initially plunges a void in particle concentration that quickly disappears again due to mixing. This figure is provided as a video in the supplementary files.

more and more particles along its ascent and transports them the remaining way up.

The average relative particle distance $h_{avg,rel} = \frac{1}{h_b}\frac{1}{N}\sum_{i=1}^{N} h_i = +2.84 \cdot 10^{-3}$ now is clearly positive. The net average relative particle distance

$$h_{avg,rel,net} = +3.86 \cdot 10^{-3} \tag{6}$$

then also is positive, meaning direct capture does enrich the particle concentration at the water surface.

Figure 4 shows the effects of both particle entrainment and direct capture. A plateau is visible from about $0.4\,h_b$ to $1.0\,h_b$ (red line). From this plateau we calculate an effective cross-section area in which any particle gets stuck to the ascending bubble: While the bubble travels between $0.4\,h_b$ to $1.0\,h_b$, traversing a distance of $h = 0.072\,\mathrm{m}$, it picks up $N = 20369$ particles. With the known initial concentration of $C = 10^5$ particles per cm^3, this number of particles corresponds to a fluid volume of $V = \frac{N}{C} = 2.0 \cdot 10^{-7}\,\mathrm{m}^3$ and cylindrical cross-section area $A = \frac{V}{h} = 2.83 \cdot 10^{-6}\,\mathrm{m}^2$. This is $\frac{A}{(d_b/2)^2\,\pi} = 23\%$ of the cross-section area of the bubble, along which the bubble picks up any particle that it encounters and transports it to the surface. In other words, particles in the inner part of the cylindrical column of water above the bubble,

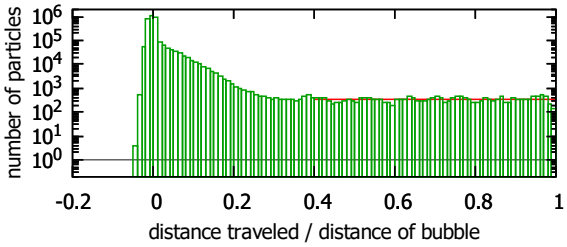


Figure 4: The distribution of the vertical distance of microplastic particles relative to the travel distance of the bubble. Here particles stick to the bubble. The bubble picks up new particles along the entire distance of travel and transports them to the top, visible as a plateauing distribution highlighted by the red line.

when following the streamlines in the flow field created by the bubble, get close enough to the water surface on the upper half of the bubble to stick to it.

Finally, we verify the influence of numerical grid resolution on our results. For lower simulation resolution, we find cross-section areas of 29% ($d_b^{\mathrm{sim}} = 48$), 31% ($d_b^{\mathrm{sim}} = 32$) and 38% ($d_b^{\mathrm{sim}} = 24$), all of which are similar to 23% in figure 4. The larger values for lower resolution are a result of the hard-potential around the bubble surface extending by one lattice point, so for a smaller bubble in simulation units, the relative thickness of the hard potential is larger, increasing the bubble radius of influence where particles adhere to the interface.

4 Conclusions

On the simulation model of a 4 mm diameter air bubble, we investigated the interactions between microplastic particles and air bubbles in water during bubble scavenging, when particle diameters are significantly smaller than the bubble diameter. We considered two possible mechanisms: entrainment – particles being dragged up in the non-laminar flow caused by the bubble – and direct capture – particles sticking to the bubble. The sticking mechanism is expected to be particularly relevant for hydrophobic microplastic particles. Pristine particles are indeed rather hydrophobic and thus tend to stick to bubbles, but become increasingly hydrophilic when left weathering in the environment, sticking less to bubbles. Our simulations indicate that the direct capture mechanism significantly increases vertical upward transport in the water column when bubbles are present. We therefore conclude that particle weathering may decrease upward transport in the water column during bubble scavenging. However, future laboratory experiments are needed to confirm our results.

Acknowledgements

We acknowledge the NVIDIA Corporation for donating a Titan Xp GPU and an A100 40GB GPU for our research.

Funding

Open Access funding enabled and organized by Projekt DEAL. This study was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1357 - 391977956, and Number 491183248. Funded by the Open Access Publishing Fund of the University of Bayreuth.

Abbreviations

LBM – lattice Boltzmann method
VoF – Volume-of-Fluid
IBM – immersed-boundary method
SML – sea surface microlayer

Availability of data and materials

The data sets used and/or analyzed during the current study are publicly available on Zenodo (https://doi.org/10.5281/zenodo.7655104) and the simulation source code is available from the corresponding author upon reasonable request.

Ethics approval and consent to participate

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Authors' contributions

ML and FH implemented and validated the simulation software. ML conducted the simulations and evaluated the data. ML and SG contributed to study design. ML wrote the manuscript. The authors read and approved the final manuscript

5 References

[1] Momoki Koga. "Bubble entrainment in breaking wind waves". In: *Tellus* 34.5 (1982), pp. 481–489.

[2] Martin E Weber, Duncan C Blanchard, and Lawrence D Syzdek. "The mechanism of scavenging of waterborne bacteria by a rising bubble". In: *Limnology and Oceanography* 28.1 (1983), pp. 101–105.

[3] E Mileva. "Solid particle in the boundary layer of a rising bubble". In: *Colloid and Polymer Science* 268.4 (1990), pp. 375–383.

[4] Anh V Nguyen, John Ralston, and Hans J Schulze. "On modelling of bubble–particle attachment probability in flotation". In: *International Journal of Mineral Processing* 53.4 (1998), pp. 225–249.

[5] Chi M Phan et al. "Investigations of bubble–particle interactions". In: *International Journal of Mineral Processing* 72.1-4 (2003), pp. 239–254.

[6] Yaowen Xing et al. "Recent experimental advances for understanding bubble-particle attachment in flotation". In: *Advances in colloid and interface science* 246 (2017), pp. 105–132.

[7] P Ahmadi et al. "Systematic Evaluation of Physical Parameters Affecting the Terminal Settling Velocity of Microplastic Particles in Lakes Using CFD. Front". In: *Environ. Sci* 10 (2022), p. 875220.

[8] AF Carlucci and PM Williams. "Concentration of bacteria from sea water by bubble scavenging". In: *ICES Journal of Marine Science* 30.1 (1965), pp. 28–33.

[9] Duncan C Blanchard, Lawrence D Syzdek, and Martin E Weber. "Bubble scavenging of bacteria in freshwater quickly produces bacterial enrichment in airborne jet drops". In: *Limnology and oceanography* 26.5 (1981), pp. 961–964.

[10] Josephine Y Aller et al. "The sea surface microlayer as a source of viral and bacterial enrichment in marine aerosols". In: *Journal of aerosol science* 36.5-6 (2005), pp. 801–812.

[11] Peter LL Walls and James C Bird. "Enriching particles on a bubble through drainage: Measuring and modeling the concentration of microbial particles in a bubble film at rupture". In: *Elementa: Science of the Anthropocene* 5 (2017).

[12] Roman Marks et al. “Rising bubbles as mechanism for scavenging and aerosolization of diatoms”. In: *Journal of Aerosol Science* 128 (2019), pp. 79–88.

[13] Donald E Spiel. “The sizes of the jet drops produced by air bubbles bursting on sea-and freshwater surfaces”. In: *Tellus B: Chemical and physical meteorology* 46.4 (1994), pp. 325–338.

[14] Jin Wu. “Jet drops produced by bubbles bursting at the surface of seawater”. In: *Journal of physical oceanography* 32.11 (2002), pp. 3286–3290.

[15] Peter LL Walls, Louis Henaux, and James C Bird. “Jet drops from bursting bubbles: How gravity and viscosity couple to inhibit droplet production”. In: *Physical Review E* 92.2 (2015), p. 021002.

[16] Elisabeth Ghabache and Thomas Séon. “Size of the top jet drop produced by bubble bursting”. In: *Physical Review Fluids* 1.5 (2016), p. 051901.

[17] Peter LL Walls et al. “Quantifying the potential for bursting bubbles to damage suspended cells”. In: *Scientific reports* 7.1 (2017), pp. 1–9.

[18] Luc Deike et al. “Dynamics of jets produced by bursting bubbles”. In: *Physical Review Fluids* 3.1 (2018), p. 013603.

[19] Alexis Berny et al. “Role of all jet drops in mass transfer from bursting bubbles”. In: *Physical Review Fluids* 5.3 (2020), p. 033605.

[20] Alexis Berny et al. “Statistics of jet drop production”. In: *Geophysical Research Letters* 48.10 (2021), e2021GL092919.

[21] Francisco J Blanco-Rodríguez and JM Gordillo. “On the jets produced by drops impacting a deep liquid pool and by bursting bubbles”. In: *Journal of Fluid Mechanics* 916 (2021).

[22] AH Woodcock. “Bursting bubbles and air pollution”. In: *Sewage and Industrial Wastes* (1955), pp. 1189–1192.

[23] GM Afeti and FJ Resch. “Distribution of the liquid aerosol produced from bursting bubbles in sea and distilled water”. In: *Tellus B* 42.4 (1990), pp. 378–384.

[24] Henri Lhuissier and Emmanuel Villermaux. “Bursting bubble aerosols”. In: *Journal of Fluid Mechanics* 696 (2012), pp. 5–44.

[25] Bingqiang Ji, Zhengyu Yang, and Jie Feng. “Compound jetting from bubble bursting at an air-oil-water interface”. In: *Nature communications* 12.1 (2021), pp. 1–10.

[26] Duncan C Blanchard and Lawrence D Syzdek. “Water-to-air transfer and enrichment of bacteria in drops from bursting bubbles”. In: *Applied and environmental microbiology* 43.5 (1982), pp. 1001–1005.

[27] Duncan C Blanchard. “The ejection of drops from the sea and their enrichment with bacteria and other materials: a review”. In: *Estuaries* 12.3 (1989), pp. 127–137.

[28] Ruo-Shan Tseng et al. “Sea-to-air transfer of surface-active organic compounds by bursting bubbles”. In: *Journal of Geophysical Research: Oceans* 97.C4 (1992), pp. 5201–5206.

[29] Maria Masry et al. “Experimental evidence of plastic particles transfer at the water-air interface through bubble bursting”. In: *Environmental Pollution* 280 (2021), p. 116949.

[30] Ruei-Feng Shiu et al. “New insights into the role of marine plastic-gels in microplastic transfer from water to the atmosphere via bubble bursting”. In: *Water Research* 222 (2022), p. 118856.

[31] Bingqiang Ji, Amrit Singh, and Jie Feng. “Water-to-Air Transfer of Nano/Microsized Particulates: Enrichment Effect in Bubble Bursting Jet Drops”. In: *Nano Letters* (2022).

[32] Shanye Yang et al. “Constraining Microplastic Particle Emission Flux from the Ocean”. In: *Environmental Science & Technology Letters* (2022).

[33] Lena Dubitsky, Oliver McRae, and James C Bird. “Enrichment of Scavenged Particles in Jet Drops Determined by Bubble Size and Particle Position”. In: *Physical Review Letters* 130.5 (2023), p. 054001.

[34] Moritz Lehmann et al. “Ejection of marine microplastics by raindrops: a computational and experimental study”. In: *Microplastics and Nanoplastics* 1.18 (2021), pp. 1–19.

[35] Katsiaryna Pabortsava and Richard S Lampitt. “High concentrations of plastic hidden beneath the surface of the Atlantic Ocean”. In: *Nature communications* 11.1 (2020), pp. 1–11.

[36] Ana L d F Lacerda et al. “Plastics in sea surface waters around the Antarctic Peninsula”. In: *Scientific reports* 9.1 (2019), pp. 1–12.

[37] Trishan Naidoo and David Glassom. “Sea-surface microplastic concentrations along the coastal shelf of KwaZulu–Natal, South Africa”. In: *Marine pollution bulletin* 149 (2019), p. 110514.

[38] Tamara Gajšt et al. “Sea surface microplastics in Slovenian part of the Northern Adriatic”. In: *Marine pollution bulletin* 113.1-2 (2016), pp. 392–399.

[39] Young Kyoung Song et al. “Large accumulation of micro-sized synthetic polymer particles in the sea surface microlayer”. In: *Environmental science & technology* 48.16 (2014), pp. 9014–9021.

[40] Doo-Hyeon Chae et al. “Abundance and distribution characteristics of microplastics in surface seawaters of the Incheon/Kyeonggi coastal region”. In: *Archives of environmental contamination and toxicology* 69.3 (2015), pp. 269–278.

[41] Zachary T Anderson et al. “A rapid method for assessing the accumulation of microplastics in the sea surface microlayer (SML) of estuarine systems”. In: *Scientific Reports* 8.1 (2018), pp. 1–11.

[42] Hannes Laermanns et al. “Tracing the horizontal transport of microplastics on rough surfaces”. In: *Microplastics and Nanoplastics* 1.11 (2021), pp. 1–12.

[43] Fabian Häusl. *MPI-based multi-GPU extension of the Lattice Boltzmann Method*. 2019. URL: https://epub.uni-bayreuth.de/5689/.

[44] Moritz Lehmann. *High Performance Free Surface LBM on GPUs*. 2019. URL: https://epub.uni-bayreuth.de/5400/.

[45] Moritz Lehmann et al. “Accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit, and customized 16-bit number formats”. In: *Phys. Rev. E* 106 (1 2022), p. 015308. DOI: 10.1103/PhysRevE.106.015308. URL: https://link.aps.org/doi/10.1103/PhysRevE.106.015308.

[46] Moritz Lehmann and Stephan Gekle. “Analytic Solution to the Piecewise Linear Interface Construction Problem and Its Application in Curvature Calculation for Volume-of-Fluid Simulation Codes”. In: *Computation* 10.2 (2022), p. 21.

[47] Fabian Häusl. *Soft Objects in Newtonian and Non-Newtonian Fluids : a Computational Study of Bubbles and Capsules in Flow*. 2022. URL: https://epub.uni-bayreuth.de/5960/.

[48] Stefan Frijters, Timm Krüger, and Jens Harting. “Parallelised Hoshen–Kopelman algorithm for lattice-Boltzmann simulations”. In: *Computer Physics Communications* 189 (2015), pp. 92–98.

[49] Christoph Schwarzmeier et al. “Comparison of free surface and conservative Allen-Cahn phase field lattice Boltzmann method”. In: *arXiv preprint arXiv:2206.11637* (2022).

[50] C Anela Choy et al. “The vertical distribution and biological transport of marine microplastics across the epipelagic and mesopelagic water column”. In: *Scientific reports* 9.1 (2019), pp. 1–9.

[51] Roman Marks. “Bubble mediated polymerization of RNA and DNA”. In: *AIMS Biophysics* 9.2 (2022), pp. 96–107.

[52] Roland Clift, John R Grace, and Martin E Weber. “Bubbles, drops, and particles”. In: (2005).

[53] Ahmed Al Harraq and Bhuvnesh Bharti. “Microplastics through the Lens of Colloid Science”. In: *ACS Environmental Au* 2.1 (2021), pp. 3–10.

[54] Yang Changfu et al. “Interface behavior changes of weathered polystyrene with ciprofloxacin in seawater environment”. In: *Environmental Research* 212 (2022), p. 113132.

[55] Guangzhou Liu et al. “Sorption behavior and mechanism of hydrophilic organic chemicals to virgin and aged microplastics in freshwater and seawater”. In: *Environmental Pollution* 246 (2019), pp. 26–33.

[56] N Kugou, K Ishida, and A Yoshida. “Experimental study on motion of air bubbles in seawater (terminal velocity and drug coefficient of air bubble rising in seawater)”. In: *WIT Transactions on The Built Environment* 68 (2003).

[57] Roman Marks. “Bubble rotational features-preliminary investigations”. In: *Oceanography: Open Access* (2014), pp. 1–6.

8.3 Publication 3

Accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit, and customized 16-bit number formats

by

Moritz Lehmann, Mathias J Krause, Giorgio Amati, Marcello Sega, Jens Harting, and Stephan Gekle

Physical Review E 106, 015308 (2022)
http://dx.doi.org/10.1103/PhysRevE.106.015308

Accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit, and customized 16-bit number formats

Moritz Lehmann [1,*] Mathias J. Krause [2] Giorgio Amati [3] Marcello Sega [4] Jens Harting [4,5] and Stephan Gekle [1]

[1]*Biofluid Simulation and Modeling–Theoretische Physik VI, University of Bayreuth, Bayreuth, Germany*
[2]*Institute of Mechanical Process Engineering and Mechanics, Karlsruhe Institute of Technology, Karlsruhe, Germany*
[3]*CINECA, SCAI–SuperComputing Applications and Innovation Department, Rome Branch, Italy*
[4]*Helmholtz Institute Erlangen-Nürnberg for Renewable Energy, Erlangen, Germany*
[5]*Department of Chemical and Biological Engineering and Department of Physics, Friedrich-Alexander-Universität, Erlangen, Germany*

(Received 12 January 2022; accepted 7 June 2022; published 26 July 2022)

Fluid dynamics simulations with the lattice Boltzmann method (LBM) are very memory intensive. Alongside reduction in memory footprint, significant performance benefits can be achieved by using FP32 (single) precision compared to FP64 (double) precision, especially on GPUs. Here we evaluate the possibility to use even FP16 and posit16 (half) precision for storing fluid populations, while still carrying arithmetic operations in FP32. For this, we first show that the commonly occurring number range in the LBM is a lot smaller than the FP16 number range. Based on this observation, we develop customized 16-bit formats—based on a modified IEEE-754 and on a modified posit standard—that are specifically tailored to the needs of the LBM. We then carry out an in-depth characterization of LBM accuracy for six different test systems with increasing complexity: Poiseuille flow, Taylor-Green vortices, Karman vortex streets, lid-driven cavity, a microcapsule in shear flow (utilizing the immersed-boundary method), and, finally, the impact of a raindrop (based on a volume-of-fluid approach). We find that the difference in accuracy between FP64 and FP32 is negligible in almost all cases, and that for a large number of cases even 16-bit is sufficient. Finally, we provide a detailed performance analysis of all precision levels on a large number of hardware microarchitectures and show that significant speedup is achieved with mixed FP32/16-bit.

DOI: 10.1103/PhysRevE.106.015308

I. INTRODUCTION

The lattice Boltzmann method (LBM) [1–4] is a powerful tool to simulate fluid flow. The parallel nature of the underlying algorithm has led to (multi-)GPU implementations [5–62], becoming a popular choice as speedup can be up to two orders of magnitude compared to CPUs at similar power consumption. However, most GPUs have only poor FP64 (double precision) arithmetic capabilities[1] and thus the vast majority of GPU codes have been implemented in FP32 (single precision), while most CPU codes are written in FP64. This difference, and, in particular, whether FP32 is sufficient for LBM simulations compared to FP64, has been a point of persistent discussion within the LBM community [15–20,31–36,52–58,60,63–65]. Nevertheless, only a few papers [19,35,36,52,60,66] provide some comparison on how floating-point formats affect the accuracy of the LBM and mostly find only insignificant differences between FP64 and FP32 except at very low velocity and where floating-point round-off leads to spontaneous symmetry breaking. Besides the question of accuracy, a quantitative performance comparison across different hardware microarchitectures is missing, as the vast majority of LBM software is either written only for CPUs [67–79] or only for Nvidia GPUs [30–56] or CPUs and Nvidia GPUs [18–29].

A second point of concern has been the amount of video memory on GPUs, which is in general smaller than standard memory on CPU systems and can thus lead to restrictions in domain size. LBM solely works on density distribution functions (DDFs) f_i (also called fluid populations)—floating-point numbers [80–83]—that need to be loaded from and written to video memory in every time step. These DDFs take up the majority of the consumed memory. If wanting to reduce the memory footprint of LBM with reduced floating-point precision, it comes to mind to store the DDFs in a lower precision number format (streaming step) while doing arithmetic in higher (floating-point) precision (collision step). This is equivalent to decoupling arithmetic precision and memory precision [84,85]. As a desirable side effect, since the limiting factor regarding compute time is memory bandwidth [12–21,30–45,52–55,59,60,63,64,67,86–88], lower precision DDFs also vastly increase performance. Such a mixed precision variant, where arithmetic is done in FP64 and DDF storage in FP32, has already been used by Refs. [35,57]. Using FP32 arithmetic and FP16 DDF storage would be even better, but has not yet been attempted due to concerns about possibly

*Corresponding author: moritz.lehmann@uni-bayreuth.de

[1]As of June 2022, the only GPUs with >2 TFLOPs/s in FP64 are H100, MI250(X), MI210, A100, CMP 170HX, MI100, A30, V100(S), Titan V, GV100, MI60, MI50, Radeon Pro VII, GP100, P100, Radeon VII, W9100, and W8100. All other data-center, gaming, and pro GPUs have limited FP64 capabilities.

insufficient accuracy. Lower 16-bit precision has already been successfully applied to other fluid solvers [89–91] and to a lot of other high-performance computing software [92,93].

The purpose of this paper is thus twofold: First, to render mixed FP32/16-bit precisions feasible for LBM, we introduce customized 16-bit number formats that turn out to be superior to standard IEEE-754 FP16 in LBM applications and in many cases perform as accurately as FP32. Therein, we leverage that some of the FP32 bits do not contain physical information or are entirely unused, similar to Ref. [89]. This approach requires minimal code interventions and can be easily combined with any velocity set, collision operator, swap algorithm, or LBM extension. In addition to using custom-built floating-point formats, we show that shifting the DDFs by subtracting the lattice weights and computing the equilibrium DDFs in a specific order of operations as originally proposed by Skordos [66] and further investigated by He and Luo [94] and Gray and Boek [60]—an optimization beneficial across all floating-point formats and already widely used [6–12,24–26,31,35,53,60,66,68–76,88,94]—turns out absolutely crucial for the 16-bit compression.

Second, we present an extensive comparison of FP64, FP32, FP16, shifted posit16 as well as our customized formats. Regarding LBM accuracy, we study Poiseuille flow through a cylinder [95], Taylor-Green vortex energy dissipation [66,96], Karman vortices [97] from flow around a cylinder, lid-driven cavity [30,33,37,39,49,52,98–103], deformation of a microcapsule in shear flow [104–106] with the immersed-boundary method (IBM) extension, and microplastic particle transport during a raindrop impact [10] with the volume-of-fluid and IBM extensions. Regarding performance, we exploit the capability of our FluidX3D LBM implementation written in OpenCL [6–12] to provide benchmarks for all floating-point variants on a large variety of hardware, from the world's fastest datacenter GPU over various consumer GPUs and CPUs from different vendors to even a mobile phone ARM system-on-a-chip (SoC), and show roofline analysis [64,87,107] for one hardware example.

II. LATTICE BOLTZMANN ALGORITHM

A. LBM—overview

The LBM is a Navier-Stokes flow solver that discretizes space into a Cartesian lattice and time into discrete time steps [1–4]. For each point on the lattice, density ρ and velocity $\vec{u}$ of the flow are computed from so-called density distribution functions (DDFs) f_i (also called fluid populations). The DDFs are floating-point numbers and represent how many fluid molecules move between neighboring lattice points in each time step. Because of the lattice, only certain directions are possible for this exchange and there are various levels of this directional discretization, in 3D typically 19 (including the center point), where space-diagonal directions are left out. After exchange of DDFs from and to neighboring lattice points (streaming), the DDFs are redistributed locally on each lattice point (collision). For the collision, there are various approaches, the most common being the single-relaxation-time (SRT), two-relaxation-time (TRT), and multirelaxation-time (MRT) collision operators [1,12].

The computation of the streaming part is done by copying the DDFs in memory to their new location. The algorithm is provided in Appendix A 2 a with notation as in Appendix A 3.

B. DDF-shifting

To achieve maximum accuracy, it is essential not to work with the DDFs f_i directly, but with shifted $f_i^{\text{shifted}} := f_i - w_i$ instead [53,60,66,88,94]. $w_i = f_i^{\text{eq}}(\rho = 1, \vec{u} = 0)$ are the lattice weights and ρ and $\vec{u}$ are the local fluid density and velocity. This requires a small change in the equilibrium DDF computation,

$$f_i^{\text{eq-shifted}}(\rho, \vec{u}) := f_i^{\text{eq}}(\rho, \vec{u}) - w_i \tag{1}$$

$$= w_i \rho \cdot \left(\frac{(\vec{u} \circ \vec{c}_i)^2}{2\,c^4} + \frac{\vec{u} \circ \vec{c}_i}{c^2} + 1 - \frac{\vec{u} \circ \vec{u}}{2\,c^2} \right) - w_i \tag{2}$$

$$= w_i \rho \cdot \left(\frac{(\vec{u} \circ \vec{c}_i)^2}{2\,c^4} - \frac{\vec{u} \circ \vec{u}}{2\,c^2} + \frac{\vec{u} \circ \vec{c}_i}{c^2} \right) + w_i (\rho - 1), \tag{3}$$

and density summation:

$$\rho = \sum_i \left(f_i^{\text{shifted}} + w_i \right) = \left(\sum_i f_i^{\text{shifted}} \right) + 1. \tag{4}$$

We emphasize that it is key to choose Eq. (3) exactly as presented without changing the order of operations,[2] otherwise the accuracy may not be enhanced at all [60,66,94]. With this exact order, the round-off error due to different sums is minimized. This offers a large benefit, most prominently on FP16 accuracy, by substantially reducing numerical loss of significance at no additional computational cost. Since it is also beneficial for regular FP32 accuracy, it is already widely used in LBM codes such as our FluidX3D [6–12], OpenLB [68–71], ESPResSo [24–26], Palabos [72–76], and some versions of waLBerla [53]. In Appendix A 2, we provide the entire algorithm without and with DDF-shifting for comparison and in Appendix A 3 we clarify our notation.

We also recommend doing the summation of the DDFs in alternating + and − order during computation of the velocity $\vec{u}$ to further reduce numerical loss of significance, for example, $u_x = (f_1 - f_2 + f_7 - f_8 + f_9 - f_{10} + f_{13} - f_{14} + f_{15} - f_{16})/\rho$ for the x component in D3Q19.

Gray and Boek [60] also proposed computing $(\rho - 1) = \sum_i f_i^{\text{shifted}}$ as a separate variable and directly inserting it into Eq. (3); while we do not advise against this, we found its benefit to be insignificant at any floating-point precision while increasing complexity of the code and thus omit it in our implementation.

Although without DDF-shifting, the equation for the equilibria dictates the number distribution of the DDFs, with DDF-shifting applied, the DDFs are always centered around zero. Higher-order equilibria definitions such as Refs. [108–110], an alternative to Eq. (3), are likely to work as well with 16-bit compression if DDF-shifting is applied, but further validation is required.

[2]To minimize the overall number of floating-point operations, terms should be precomputed such that $f_i^{\text{eq-shifted}} = A \cdot (\frac{1}{2}(B^2 + C) \pm B) + D$ requires only three fused-multiply-add (FMA) operations.

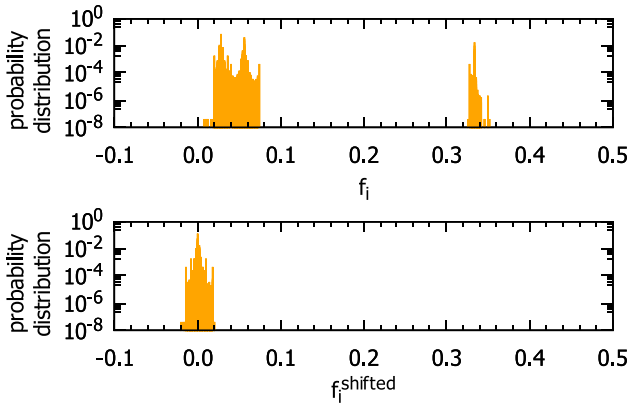


FIG. 1. Histogram of the DDFs for the lid-driven cavity simulation from Sec. IV D (Re = 1000, Ma = 0.17, grid resolution 128^3) after 100 000 LBM time steps. The simulation is performed without the DDF-shifting (top) and with DDF-shifting (bottom), both times in FP32/FP32.

C. Which range of numbers does the LBM use?

In Fig. 1, we present the distribution of f_i and f_i^{shifted} for the example system of the lid-driven cavity from Sec. IV D. A more detailed look at the DDF distributions of this system are provided in Figs. 19 and 20 in the Appendix. Similar data for the remaining setups are given in Appendix Fig. 21. It is quite remarkable how the number range in all cases is very limited. The f_i accumulate around the LBM lattice weights (for D3Q19 $w_i \in \{\frac{1}{36}, \frac{1}{18}, \frac{1}{3}\}$) and the f_i^{shifted} accumulate around 0, where floating-point accuracy is best. So for FP32 not only are the trailing bits of the mantissa expected to be nonphysical numerical noise [89], but also some bits of the exponent are entirely unused, meaning one can waive these bits without losing accuracy.

To find the theoretical maximum number range of f_i and f_i^{shifted}, we insert $\vec{u}_j = c\,\frac{\vec{c}_j}{|\vec{c}_j|}$ in Eqs. (A4) and (3) and find that $\frac{1}{\rho}\,|f_i^{\text{eq}}| \leqslant \delta$ or $\frac{1}{\rho}\,|f_i^{\text{eq-shifted}}| \leqslant \delta^{\text{shifted}}$, respectively, with the values of δ and δ^{shifted} depending on the velocity set in use (Table I).

With $\tau > 0.5$, through Eq. (A5), we get in the worst case

$$|f_i| \lessapprox |2\,f_i^{\text{eq}}| \lessapprox 2\,\rho\,\delta, \tag{5}$$

$$|f_i^{\text{shifted}}| \lessapprox |2\,f_i^{\text{eq-shifted}}| \lessapprox 2\,\rho\,\delta^{\text{shifted}}, \tag{6}$$

respectively, because the DDFs in stable simulations are expected to follow the equilibrium DDFs. The density ρ typically deviates only little from $\rho \approx 1$. Assuming $\rho < 2$ leads to $|f_i| \lessapprox 2$ being the worst-case maximum number range (D3Q13, no DDF-shifting). With the more typical D3Q19 and DDF-shifting, the same number range $|f_i^{\text{shifted}}| \lessapprox 2$ restricts the density to a less strict $\rho < 6$. Keeping the sign is required because f_i^{shifted} (and also f_i) can be negative.

$|f_i^{\text{shifted}}| \lessapprox 2$ and the resulting $\rho < 6$ is even sufficient for covering a fairly large class of compressible flows. The shock simulations in Ref. [108], for example, range in density from 0.6 to 2.2, so these simulations could possibly work as well. However, careful considerations need to be made for the individual setup to not exceed this limit. If a higher value for density is required, the floating-point formats with limited range could be shifted toward higher numbers; however, careful validation is required as this comes at the cost of worse accuracy at small numbers. The later proposed posit formats $P16_0S$ and $P16_1S$ do not put a limit on density, so they would be a better fit for simulations with large density variation.

TABLE I. The numerical value of δ and δ^{shifted} depending on the used velocity set.

	D2Q9	D3Q7	D3Q13	D3Q15	D3Q19	D3Q27
δ	0.45	0.47	0.50	0.42	0.34	0.30
δ^{shifted}	0.31	0.35	0.25	0.31	0.17	0.21

III. NUMBER REPRESENTATION MODELS

A 16-bit number can represent only 65 536 different values. The task is to spread these along the number axis in a way that the most commonly used numbers are represented with the best possible accuracy. There is a variety of number representations that come to mind as a 16-bit storage format: fixed-point, floating-point as well as the recently developed posit format [111], and each of them can be adjusted specifically for the LBM. Figure 2 illustrates the number formats investigated in this paper and Fig. 3 shows their accuracy characteristics.

A. Floating-point

1. Overview

In the normalized number range, a floating-point number [80–83] is represented as

$$x = \underbrace{(-1)^s}_{\text{sign}} \cdot \underbrace{2^{e-b}}_{\text{exponent}} \cdot \underbrace{(1 + 2^{-n_m}\,m)}_{\text{mantissa}}, \tag{7}$$

with s being the sign bit, e being an integer representing the exponent, and m being an integer representing the mantissa. b is a constant called exponent bias and n_m is the number of bits in the mantissa (values in Table II). The precision is $\log_{10}(2^{n_m+1})$ decimal digits[3] and the truncation error is $\epsilon = 2^{-n_m}$.

When the exponent e is zero, the mantissa is shifted to the right as a way to represent even smaller numbers close to zero, although at less precision. This is called the denormalized number range and making use of it during the conversions that will be described below is not straightforward, but essential alongside correct rounding to keep decent accuracy with the 16-bit formats.

2. Customized FP16 formats for the LBM

In our lattice Boltzmann simulations, we implement and test three different 16-bit floating-point formats:

(1) FP16: Standard IEEE-754 FP16, with FP32 ↔ FP16 conversion supported on all CPUs and GPUs from within the last 12 years.

[3]The +1 refers to the implicit leading bit of the mantissa.

TABLE II. Comparing the properties of the number formats used here to store the LBM DDFs f_i.

	Bits	n_m	b	Digits	ϵ	Range	Smallest normalized number	Smallest denormalized number
IEEE FP64	64	52	1023	16.0	2.2×10^{-16}	$\pm1.797693\times10^{308}$	2.225074×10^{-308}	4.940656×10^{-324}
IEEE FP32	32	23	127	7.2	1.2×10^{-7}	$\pm3.402823\times10^{38}$	1.175494×10^{-38}	1.401298×10^{-45}
IEEE FP16	16	10	15	3.3	9.8×10^{-4}	$\pm6.550400\times10^{4}$	6.103516×10^{-5}	5.960464×10^{-8}
FP16S	16	10	30	3.3	9.8×10^{-4}	$\pm1.999023\times10^{0}$	1.864464×10^{-9}	1.818989×10^{-12}
FP16C	16	11	15	3.6	4.9×10^{-4}	$\pm1.999512\times10^{0}$	6.103516×10^{-5}	2.980232×10^{-8}
Posit $P16_0S$	16	0–13	–	$\leqslant4.2$	$\geqslant1.2\times10^{-4}$	$\pm1.280000\times10^{2}$	–	4.768372×10^{-7}
Posit $P16_1S$	16	0–12	0	$\leqslant3.9$	$\geqslant2.4\times10^{-4}$	$\pm2.097152\times10^{6}$	–	2.910383×10^{-11}
Posit $P16_2C$	16	0–12	0	$\leqslant3.9$	$\geqslant2.4\times10^{-4}$	$\pm1.999756\times10^{0}$	–	1.734724×10^{-18}

(2) FP16S: We downscale the number range of IEEE-754 FP16 by $\times2^{-15}$ to ±2 and use the convenience that all modern CPUs and GPUs can do IEEE-754 floating-point conversion in hardware.

(3) FP16C: We allocate one bit less for the exponent (to decrease number range towards small numbers) and one bit more for the mantissa (to gain accuracy). The number range is also limited to ±2. This custom format requires manual conversion from and to FP32.

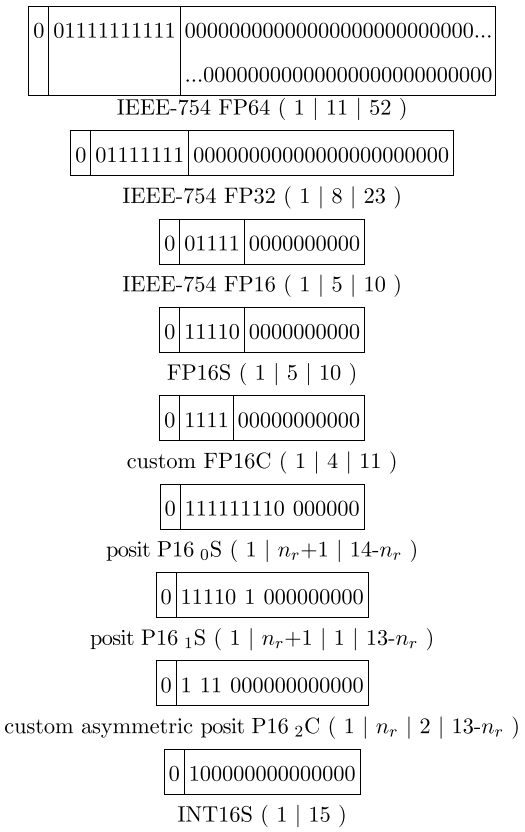


FIG. 2. The number 1.0 represented by the different formats we investigate here. The leftmost single bit is the sign s and the rightmost segment is the mantissa m. For floating-point (FP), the center segment is the exponent e. FP16S and FP16C are new formats specifically designed to store the DDFs. Fixed-point (INT16S) does not have an exponent. Posits have dynamic partitioning of the segments, with an extra regime segment and an optional exponent segment next to the mantissa.

When looking at Table II and Fig. 3, FP16S and FP16C differ in extended range toward small numbers versus halved truncation error ϵ. The question arises which of these two traits is more important for LBM. FP16 is inferior to both FP16S and FP16C as it combines lower mantissa accuracy and less range toward small numbers. Since FP16S comes at no additional computational cost and complexity compared to FP16, FP16S should always be preferred over FP16 for storing the DDFs.

3. *Floating-point conversion:* FP32 ↔ FP16S

The IEEE-754 FP32 ↔ FP16/FP16S conversion is supported in hardware and therefore only briefly described below.

FP32 → FP16S: For the FP32 → FP16 conversion, OpenCL provides the function `vstore_half_rte` that is executed in hardware. To convert to the FP16S format instead, we shift the number range up by 2^{15} via regular FP32 multiplication right before conversion. This is equivalent to increasing the exponent bias b by 15.

FP16S → FP32: For the FP16 → FP32 conversion, OpenCL provides the function `vload_half` that is executed in hardware. To convert from the FP16S format instead, we

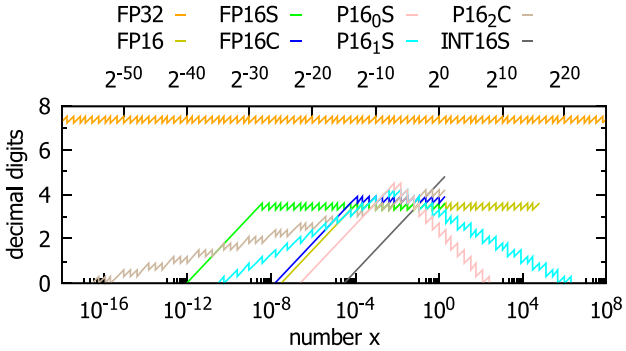


FIG. 3. Accuracy characteristics of the number formats investigated in this paper. This plot shows only the local minima (measured graphs see Fig. 18). FP16C reduces number range but increases accuracy in the normalized regime (horizontal part). For floating-point formats, the downward slope indicates the denormalized part, where accuracy behaves like fixed point. We also show 16-bit fixed-point scaled by $\times2^{-14}$ (INT16S). Posits (P16) have slopes left and right, with highest accuracy in the middle, which here is shifted from 1 to $\frac{1}{128}$, hence the "S". $P16_0S/P16_1S$ have 0/1-bit exponents, making the slopes more (less) steep and decreasing (increasing) number range. $P16_2C$ is a custom format with 2-bit exponent but asymmetric slope.

shift the number range down by 2^{-15} via regular FP32 multiplication right after conversion.

4. Floating-point conversion: FP32 ↔ FP16C

For FP32 ↔ FP16C conversion, we developed a set of fast conversion algorithms that work in any programming language and on any hardware which we describe in some more detail further below. An OpenCL C version is presented in Listing 1.

We ditch `NaN` and `Inf` definitions for an extended number range by a factor of 2 and less complicated and faster conversion. In PTX assembly [112], the FP32 → FP16C conversion takes 25 instructions and FP16C → FP32 takes 26 instructions.

FP32 → FP16C: The first step is to interpret the bits of the FP32 input number as `uint`, for which there is the `as_uint(float x)` function provided by OpenCL. The sign bit remains identical as the leftmost bit via bit-masking and bit-shifting. To assure correct rounding, we add a `1` to the 12th bit from left (`0x00000800`), because mantissa bits at positions 12 to 0 later are truncated. Next, we extract the exponent e by bit-masking and bit-shift by 23 places to the left.

For normalized numbers, the exponent is decreased by the difference in bias $127 - 15 = 112$ and bit-shifted to the right by 11 places. A final bit-mask ensures that there is no overflow into the sign bit. The mantissa is bit-shifted in place and or-ed to sign and exponent.

For denormalized numbers, we first add a `1` to place 24 (`0x00800000`) of the mantissa (to later figure out how many places the mantissa was shifted) and then bit-shift it to the right by as many places as the new exponent is below zero. Correct rounding, however, makes this a bit more difficult: We need to add `1` for rounding to the leftmost place of the mantissa that is cut off. To undo the initial rounding we did earlier, instead of `0x00800000`, we add `0x00800000-0x00000800=0x007FF800`, then shift by one place less than the new exponent is below zero, add `1` to the rightmost bit and finally shift right the one remaining place.

The exponent itself is the switch deciding whether the normalized or denormalized conversion is used. As an optional safety measure, we add saturation: If the number is larger than the maximum value, we override all exponent and mantissa bits to `1` (bitwise or with `0x7FFF`).

FP16C → FP32: To convert back to FP32, we first extract the exponent e and the mantissa m by bit-masking and bit-shifting. Additionally, since we intend to avoid branching, we already count the number of leading zeros[4] v in the mantissa for decoding the denormalized format: We cast m to `float`,[5] reinterpret the result as `uint`, bit-shift the exponent right by 23 bits and subtract the exponent bias, giving us the base-2 logarithm of m, equivalent to 31 minus the number of leading zeros.

The sign bit is bit-masked and bit-shifted in place. The exponent e again decides for normalized or denormalized numbers: For normalized numbers ($e \neq 0$), the exponent is increased by the difference in bias $127 - 15 = 112$ and or-ed together with the bit-shifted mantissa. For denormalized numbers ($e = 0$ and $m \neq 0$), the mantissa is bit-shifted to the right by the number of leading zeros and the shift-indicator `1` is removed by bit-masking. The mantissa is or-ed with the exponent which is set by the number of leading zeros and bit-shifted in place.

Finally, the `uint` result is reinterpret as `float` via the OpenCL function `as_float(uint x)`.

B. Posit

1. Overview

The novel posit format (type-III Unum) [90,111,113,114] is different from floating-point in that the bit segment for the mantissa (and also exponent) is variable in size and there is another bit segment, the regime, with variable size as well. The posit number representation is

$$x = \underbrace{(-1)^s}_{\text{sign}} \cdot \underbrace{(2^{n_e+1})^r}_{\text{regime}} \cdot \underbrace{2^e}_{\text{exponent}} \cdot \underbrace{(1 + 2^{-n_m} m)}_{\text{mantissa}}, \tag{8}$$

with sign s, regime r, exponent e, and mantissa m. $n = 1 + (n_r + 1) + n_e + n_m$ is the total number of bits, whereby n_r, n_e, and n_m are the variable numbers of bits in regime, exponent and mantissa, respectively.

For very small numbers, the regime bit pattern looks like `000..01` (negative r), then gets shorter toward `01` ($r = -1$), flips to `10` ($r = 0$) and then gets longer again, looking like `111..10` (positive r). The last bit is the regime terminator bit that unambiguously tells the length of the regime. This bit is not included in the regime size n_r, so the size of the regime bit pattern is $n_r + 1$. n_r determines the value of the regime: $r = -n_r$ if the regime terminator bit is `1` or $r = n_r - 1$ if the regime terminator bit is `0`.

For increasing regime size, the remaining bits for exponent and mantissa are shifted to the right, so the mantissa (and if no mantissa bits are left also the exponent) become shorter and precision is reduced.

Posits can be designed with different (fixed) exponent sizes or no exponent at all. Just like for floating-point, larger exponent increases the range but decreases accuracy. This way, the posit format is designed to deliver variable accuracy based on where the number is in the regime: best accuracy is around ± 1.0 where the regime is shortest (superior to floating-point) but for both tiny and large numbers, much precision is lost [114].

2. Customized posit formats for the LBM

As a storage format for LBM DDFs, where numbers close to 0 need to be resolved best and numbers outside the ± 2 range are not required at all, the standard 16-bit posit formats seems an unfavorable choice. However, by multiplying a constant before and after conversion, similar to FP16S, we shift the most accurate part down to smaller numbers. We take a closer look at three different posit formats:

(1) $\text{P16}_0\text{S}$: 16-bit posit without exponent, shifted down by $\times 2^7$. In the interval $[2^{-11}, 2^{-3}]$, accuracy is equal to or better

[4]The OpenCL function `clz(m)` also counts the number of leading zeros. While translated into a single `clz.b32` PTX instruction (instead of `cvt.rn.f32.u32 mov.b32 shr.u32`), `clz.b32` executes much slower, leading to noticeably worse performance.

[5]Casting an `int` to `float` implicitly does a `log2` operation to determine the exponent.

than FP16S and in the interval $[2^{-10}, 2^{-4}]$, accuracy is equal to or better than FP16C. The range toward small numbers is very poor and for numbers $>2^{-3}$, accuracy is vastly degraded.

(2) $P16_1S$: 16-bit posit with one-bit exponent, shifted down by $\times 2^7$. In the interval $[2^{-13}, 2^{-1}]$, accuracy is equal to or better than FP16S and in the interval $[2^{-11}, 2^{-3}]$, accuracy is equal to or better than FP16C. For numbers $<2^{-13}$ or $>2^{-1}$, accuracy is reduced. The range toward small numbers is between FP16S and FP16C. This format poses no limitations on the density ρ because its number range is $\pm 2^{21}$.

(3) $P16_2C$: Custom asymmetric 16-bit posit with two-bit exponent, not shifted. By only covering the lower flank, we can get rid of the bit reserved for the regime sign, thus making the regime shorter by one bit and increasing the mantissa size by one bit in turn. The conversion algorithms are vastly simplified with the asymmetric regime. Accuracy is better or equal to FP16C in the interval $[2^{-7}, 2]$ and equal or better than FP16S in the interval $[2^{-11}, 2]$. For smaller numbers, accuracy is slowly reduced, but the range toward small numbers is excellent.

Both $P16_0S$ and $P16_1S$ provide numbers >2 that are unused in the LBM. Shifting the number range further down would degrade accuracy for larger numbers too much. Since the LBM with DDF-shifting uses numbers around 0 and it is not entirely clear in which order of magnitude accuracy is most important, it is also unclear if the increased accuracy in the center interval will benefit more than the decreased accuracy further away from the center will adversely affect.

3. FP32 ↔ *posit conversion*

Conversion between FP32 and posit is not supported in hardware (yet). Since the reference conversion algorithm in the SoftPosit library [115] is not written for speed primarily, we provide self-written, ultrafast conversion algorithms in Listing 1 in OpenCL C. These work on any hardware. A detailed description of how the algorithms work is omitted here but can be inferred by studying the provided listings. Note that the posit specification [111] does two's complement for negative numbers to have no duplicate zero and an infinity definition instead. To simplify the conversion algorithms and since infinity is not required in our applications, we just use the sign bit to reduce operations, so there is positive and negative zero.

C. Fixed-point

16-bit fixed-point format with a range scaling of ± 2 has discrete additive steps of $2^{-14} \approx 6.110^{-5}$, so this is also the smallest possible value. Compared to floating-point, precision is worse for small numbers and better for large numbers. For the LBM, this is insufficient and does not work.

D. Required code interventions

At all places where the DDFs are used as kernel parameters, their data type is made switchable with a macro (`fpXX`). At any location where the DDFs are loaded or stored in memory, the load (store) operation is replaced with another macro as provided in Listing 1 for FP32, FP16S, FP16C, $P16_0S$, $P16_1S$, and $P16_2C$. In the Appendix in Listing 2, we provide the core of our LBM implementation, exemplary for D3Q19 SRT.

IV. ACCURACY COMPARISON

A. 3D Poiseuille flow

A standard setup for LBM validation is a Poiseuille flow through a cylindrical channel [95]. For the channel walls, we use standard nonmoving midgrid bounce-back boundaries [1,12] and we drive the flow with a body force as proposed by Guo *et al.* [116]. Simulations are done with the D3Q19 velocity set and a single-relaxation-time (SRT) collision operator. We compare the simulated flow profile $u_{\mathrm{sim}}(r)$ with the analytic solution [117]

$$u_{\mathrm{theo}}(r) = \frac{f}{4\,\rho\,\nu}\,(R^2 - r^2) \tag{9}$$

to compute the error. Here, $\rho = 1$ is the average fluid density,

$$r = \sqrt{\left(y - \frac{L_y}{2}\right)^2 + \left(z - \frac{L_z}{2}\right)^2} \tag{10}$$

is the radial distance from the channel center, R is the channel radius,

$$\nu = \frac{2\,R\,u_{\max}}{Re} = \frac{\tau}{3} - \frac{1}{6} \tag{11}$$

is the kinematic shear viscosity, and τ is the relaxation time. The dimensions of the simulation box are

$$L_x = 1, \quad L_y = L_z := 2\,(R+1). \tag{12}$$

The flow is driven by a force per volume f that is calculated by rearranging Eq. (9) with $r = 0$:

$$f = \frac{4\,\rho\,\nu\,u_{\max}}{R^2}. \tag{13}$$

In accordance with Ref. [12], we define the error as the L_2 norm [1, p. 138]:

$$E = \sqrt{\frac{\sum_{r=0}^{R} |u_{\mathrm{sim}}(r) - u_{\mathrm{theo}}(r)|^2}{\sum_{r=0}^{R} |u_{\mathrm{theo}}(r)|^2}}. \tag{14}$$

In Fig. 4, we keep the Reynolds number and center velocity constant at $\mathrm{Re} = 10$ and $u_{\max} = 0.1$, and vary channel radius

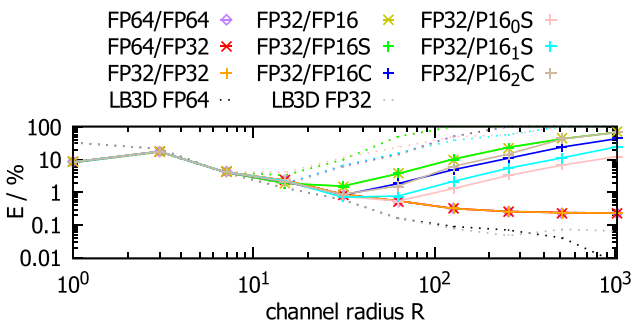
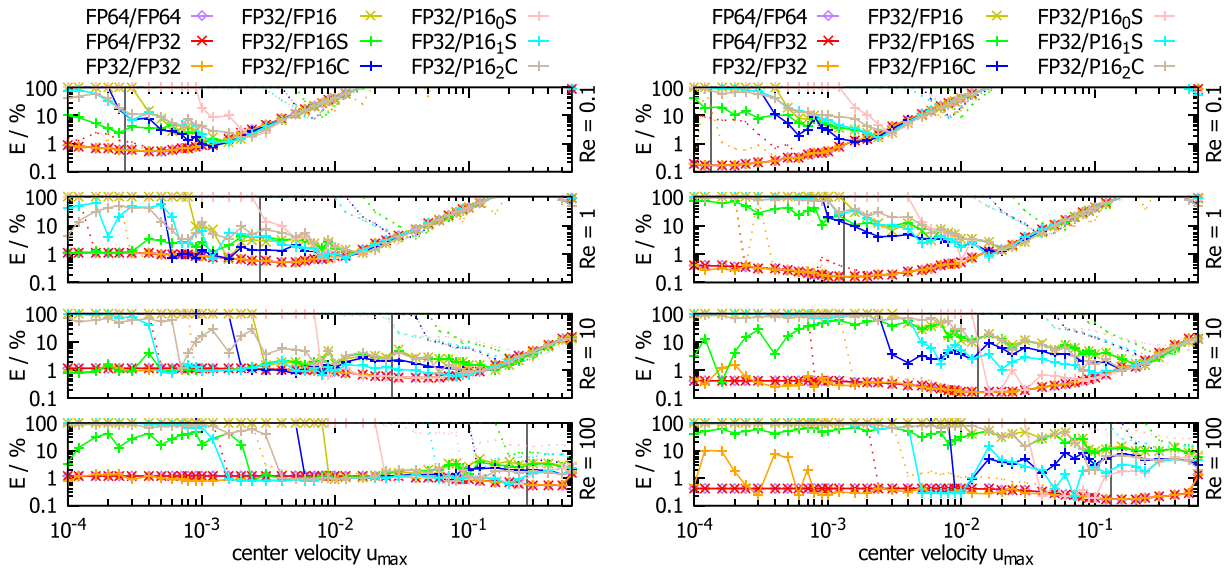


FIG. 4. Error of D3Q19 SRT Poiseuille flow for varying channel radius R (lattice resolution) at constant Reynolds number $\mathrm{Re} = 10$ and constant center flow velocity $u_{\max} = 0.1$. The dashed lines represent corresponding simulations without DDF-shifting.



(a) $R = 31$ (b) $R = 63$

FIG. 5. Error of D3Q19 SRT Poiseuille flow for varying center velocity $u_{\max}$ at constant Reynolds number $\mathrm{Re} \in \{0.1, 1, 10, 100\}$ and constant channel radius (a) $R = 31$ and (b) $R = 63$. The dashed curves represent corresponding simulations without DDF-shifting. The vertical lines represent the LBM relaxation time $\tau = 1$.

R and kinematic shear viscosity ν accordingly. For $R \leqslant 15$, we see almost no difference between any of the floating-point variants. Here the staircase effect of the channel walls dominates the error. Moving toward larger radii, the error increases at first for FP32/FP16 and FP32/FP16S and later for FP32/FP16C as well, while FP64/xx and FP32/FP32 show no difference in this regime either. 16-bit posit formats hold up even better here with their increased peak accuracy. P16$_2$C for small R behaves like FP16C and then migrates over to FP16S as R becomes larger and the DDFs become smaller. We also simulate the same system without DDF-shifting (dashed lines) to quantify the difference. Already here we see that the 16-bit formats become unfeasible without DDF-shifting.

To confirm that the observed agreement between FP32/FP32 and FP64/FP64 is not a coincidence of our implementation, in Fig. 4 we include data from a simulation of the very same system with the LB3D code [79] that is further described in the Appendix.

We now investigate the error in more detail for a constant channel radius $R \in \{31, 63\}$ in Fig. 5. We simulate the flow in the channel for different Reynolds numbers $\mathrm{Re} \in \{0.1, 1, 10, 100\}$ and vary the center velocity $u_{\max}$ and kinematic shear viscosity ν accordingly.

We find that the higher the Reynolds number, the further the minimal error is shifted toward larger $u_{\max}$, always staying close to where $\tau = 1$ (vertical lines). The better small numbers can be resolved, the lower $u_{\max}$ can be chosen before the error suddenly becomes large. The better the accuracy of the mantissa, the lower the overall error, up to a certain point where discretization errors dominate at large $u_{\max}$.

It is important to consider that compute time is proportional to $u_{\max}$ and that $u_{\max} < u_{\max,\tau=1} = \frac{\mathrm{Re}}{12R}$ smaller than at the error minimum is thus less practically relevant. In the domain $u_{\max} \geqslant u_{\max,\tau=1}$ (in Fig. 5, right of the vertical lines), FP16C is almost always superior to FP16S, especially at higher Re. Posits show their superior precision most of the time, if the DDFs are just in the right interval.

We find that without DDF-shifting, the 16-bit formats become very inaccurate. For FP32/FP32, there is some benefit at higher Re and especially low velocities $u_{\max}$. For FP64, the DDF-shifting does not make any noticeable difference in this setup as discretization errors dominate.

To better understand where the error comes from in the Poiseuille channel radially, we exemplary plot the error contribution as a function of the radial coordinate r for the parameters $R = 63$, $u_{\max} = 0.1$, $\mathrm{Re} = 10$ in Fig. 6. We find that for FP64 to FP32, the largest error contribution is near the channel wall (staircase effect along the no-slip bounce-back boundaries). For FP32/16-bit, there is equal error contribution near the wall, but the majority of the error comes from close to the channel center. The wall poses a boundary condition not only for velocity $[u(R) = 0]$ but also for the

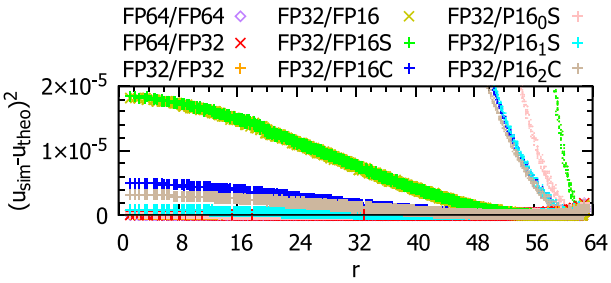


FIG. 6. Radial error profile of D3Q19 SRT Poiseuille flow for a channel radius $R = 63$ and center flow velocity $u_{\max} = 0.1$ at constant Reynolds number $\mathrm{Re} = 10$. The small dots on the right represent corresponding simulations without DDF-shifting. Note that the error contribution is on a linear scale here.

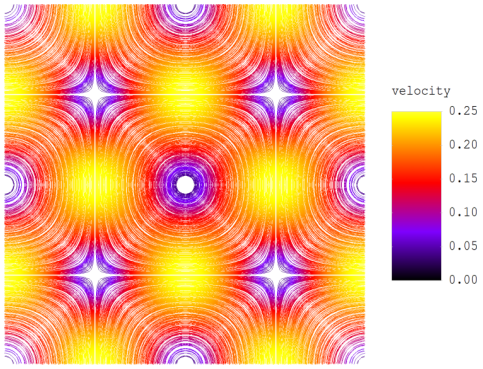


FIG. 7. Illustration of the velocity field at $t = 0$ with colored streamlines.

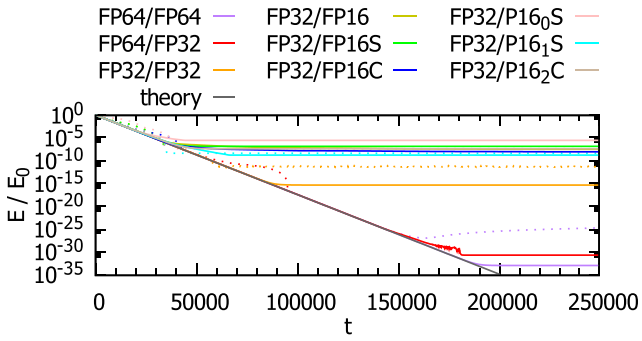


FIG. 8. Relative kinetic energy $E(t)/E_0$ of a D2Q9 SRT simulation of Taylor-Green vortices compared to the analytic solution in Eq. (18). Dashed lines represent corresponding simulations without DDF-shifting.

velocity error. Going radially inward from the channel wall, at first the staircase effect smooths out, lowering the error, but then each concentric ring of lattice points accumulates systematic floating-point errors, so at the channel center the error is largest. For FP32/FP32, this error behavior is barely noticeable but visible upon close inspection. For FP64, the floating-point errors are so tiny that the staircase smoothing continues all the way through the radial profile, making the error smallest in the center. Without DDF-shifting, there is no noticeable difference for FP64 and FP32 compared to when DDF-shifting is done, but the 16-bit formats become unfeasible.

B. Taylor-Green vortices

An especially well suited setup for testing the behavior at low velocities is Taylor-Green vortices. A periodic grid of vortices is initialized with velocity magnitude u_0 (illustrated in Fig. 7) and then over time viscous friction slows down the vortices while they remain in place on the grid. In 2D, the analytic solution [66,96] reads

$$u_x(t) = +u_0 \cos(k\,x) \sin(k\,y)\, e^{-2\,\nu\,k^2\,t}, \tag{15}$$

$$u_y(t) = -u_0 \sin(k\,x) \cos(k\,y)\, e^{-2\,\nu\,k^2\,t}, \tag{16}$$

$$\rho(t) = 1 - \frac{3\,u_0^2}{4}\left(\cos(2\,k\,x) + \cos(2\,k\,y)\right) e^{-4\,\nu\,k^2\,t}, \tag{17}$$

and at $t = 0$ is used to initialize the simulation with $u_0 = 0.25$. Here $\nu = \frac{\tau}{3} - \frac{1}{6} = \frac{1}{6}$ is the kinematic shear viscosity at $\tau = 1$ and $k = \frac{2\,\pi\,N}{L}$. $L = 256$ is the side length of the square lattice and $N = 1$ is the number of periodic tiles in one direction. The kinetic energy

$$E(t) = \int_0^L \int_0^L \frac{\rho}{2}\left(u_x^2 + u_y^2\right) dx\,dy$$
$$= u_0^2\,\pi^2\, e^{-4\,\nu\,k^2\,t} \tag{18}$$

drops exponentially with time t. $E_0 = E(t = 0)$ denotes the initial kinetic energy. We compute the kinetic energy from the simulation as the discrete sum across all lattice points and compare it to the analytic solution in Fig. 8. The simulated kinetic energy drops exponentially as well, but at some point it does not drop further and remains constant as a result of floating-point errors. The relative energies of the plateaus are no coincidence: The plateaus are located at approximately the truncation error ϵ squared (Table II) for the respective number format in use. Particularly interesting is that for FP64/FP32 the plateau is much lower than for FP32 ϵ^2, being closer to FP64 ϵ^2. With P16$_0$S, the DDFs are outside of the most accurate interval, so accuracy is poor overall.

Finally, we note that the plateaus only reach down to ϵ^2 if DDF-shifting is properly implemented as presented in Eq. (3). Without DDF-shifting, there is significant loss in accuracy across all number formats.

C. Karman vortex street

Our next setup is a Karman vortex street in two dimensions [97]: a cylinder with radius $R = 32$ is placed into a simulation box with dimensions 512×1024. At the box perimeter, a velocity of $\vec{u} = (0,\ 0.15)$ is enforced using nonreflecting equilibrium boundaries [12,118]. The Reynolds number is set to $\mathrm{Re} = \frac{2R|u|}{\nu} = 250$, defining the kinematic shear viscosity $\nu = \frac{\tau}{3} - \frac{1}{6}$ and relaxation time τ.

If starting the simulation with perfectly symmetric initial conditions, only floating-point errors can eventually trigger the Karman vortex instability. We notice that in some cases, the instability would not start at all even after several hundred thousand time steps. To avoid this nonphysical behavior, we initialize the velocity $\vec{u} = (0,\ 0.15)$ not only at the simulation box perimeter but also on the left half $x < 256$. This immediately triggers the Karman vortex instability regardless of floating-point setting.

We probe the velocity at the simulation box center (256, 512) over time in Fig. 9. This demonstrates that, when DDF-shifting is done, the 16-bit formats are almost indistinguishable from FP64 ground truth both qualitatively and quantitatively, with only minimal phase-shift for FP16, FP16S, and P16$_0$S.

To assess in detail where eventual differences may be present beyond a single velocity point probe, we look at the vorticity throughout the simulation box. In Fig. 10, we show the vorticity in the very much zoomed-in range of ±0.001. For the 16-bit formats, in low vorticity areas there is noise present. Comparing FP16 and FP16S, the extended number range

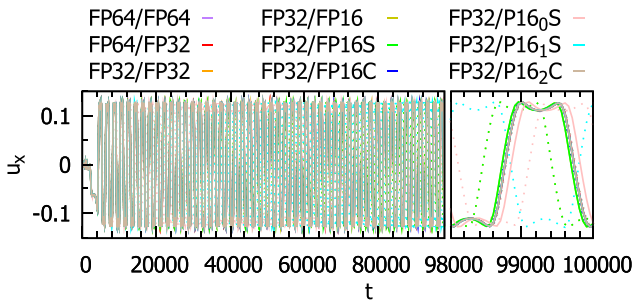


FIG. 9. Velocity x component of the Karman vortex street at the simulation box center (256, 512) over time for various floating-point precision. Dashed lines represent corresponding simulations without DDF-shifting. Even after 100 000 LBM time steps (50 vortex periods), the 16-bit graphs still cover the FP64 ground truth as amplitude, frequency, and even phase appear indistinguishable. Only zooming in at the last oscillation period reveals minuscule differences in phase for FP16, FP16S, and P16$_0$S. The phase shift in the 16-bit graphs is large without the DDF-shifting optimization. FP32/FP16C, FP32/FP32, and FP64/FP32 are indistinguishable from FP64 ground truth even when zooming in.

toward small numbers has no benefit here. FP16C with DDF-shifting mostly mitigates this noise, showing that the noise purely originates in smaller mantissa accuracy and numeric loss off significance. Our custom posit P16$_2$C has similarly low noise. P16$_0$S shows artifacts.

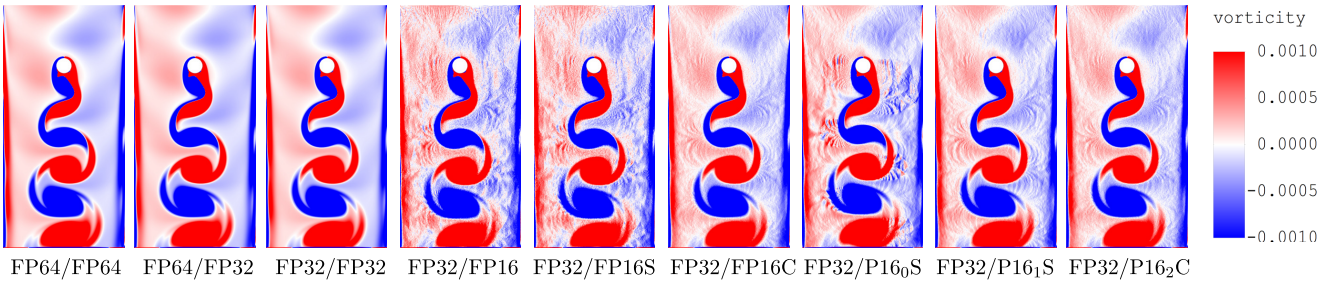
D. Lid-driven cavity

The lid-driven cavity is a common test setup for the LBM [30,33,37,39,49,52,98–100] and other Navier-Stokes solvers [101–103]. We here implement it in a cubic box at Reynolds number Re = 1000. On the lid, velocity parallel to the y-axis is enforced through moving bounce-back boundaries [1,12]. The box edge length is $L = 128$, the velocity at the top lid is $u_0 = 0.1$ in lattice units, and the kinematic shear viscosity is set by the Reynolds number $\mathrm{Re} = \frac{L u_0}{\nu} = 1000$. We simulate 100 000 LBM time steps with the D3Q19 SRT scheme.

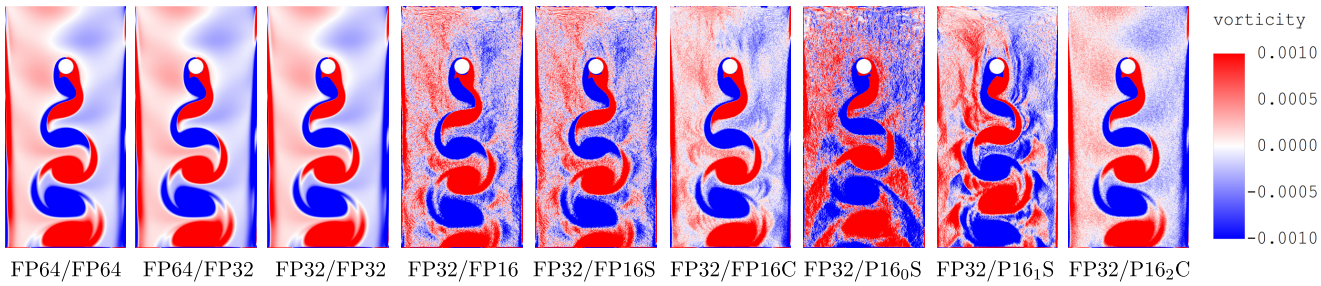
Figure 11 shows the y (z) velocity along horizontal (vertical) probe lines through the simulation box center. All number formats except P16$_0$S look indistinguishable, even without DDF-shifting. Only when zooming in (not shown), for the simulations without DDF-shifting, deviations in relative velocity in the second digit become visible. With DDF-shifting, deviations are present only in the fourth digit, being smallest for FP16C, P16$_1$S, and P16$_2$C.

E. Capsule in shear flow

Here we test the number formats on a microcapsule in shear flow, one of the standard tests for microfluidics simulations in medical applications [105,106]. The D3Q19 multi-relaxation-time (MRT) [1,12] LBM is extended with the IBM [119] to simulate the deformable microcapsule in flow. For the IBM, we use the same level of precision as for the LBM arithmetic,



(a) simulations with DDF-shifting



(b) simulations without DDF-shifting

FIG. 10. Vorticity in the vastly overexposed range ±0.001 for simulations (a) with and (b) without DDF-shifting, after 100 000 LBM time steps. All simulations very accurately predict the vortex street, with frequency and amplitude of the vortices being identical and only insignificant differences in phase-shift even after 50 vortex periods. FP32 is indistinguishable from FP64 ground truth. For 16-bit, in the low vorticity range there is noise present, equally for FP16 and FP16S, but vastly reduced for FP16C and P16$_2$C. Omitting DDF-shifting vastly increases this noise and also adds significant phase shift as can be seen by comparing the position of the last red vortex at the bottom.

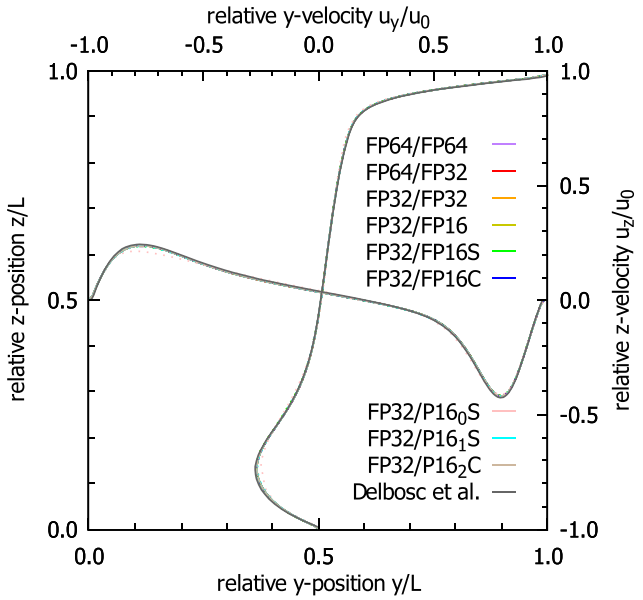


FIG. 11. The y-velocity along a vertical probe line through the simulation box center as well as the z-velocity along a horizontal line through the simulation box center. At the top lid, the velocity is fixed. As the flow goes one rotation clockwise, the width of the high-velocity peak increases and the height decreases. Dashed lines represent corresponding simulations without DDF-shifting. As a reference, we show the data from Delbosc *et al.* [33].

so either FP64 or FP32. As illustrated in Fig. 12, we place an initially spherical capsule of radius $R = 13.5$ in the center of a simulation box with the dimensions $128\times64\times192$, and we compute 385 000 time steps. At the top and bottom of the simulation box, a shear flow is enforced via moving bounce-back boundaries [120]. The membrane of the capsule is discretized into 5120 triangles and membrane forces, consisting of shear forces (neo-Hookean) [104,105,121] as well as volume forces (volume has to be conserved), are computed as in Ref. [105].

The Reynolds number is $\mathrm{Re} = 0.05$, the kinematic shear viscosity is $\nu = \frac{1}{3}$, and we simulate various capillary numbers $\mathrm{Ca} = \frac{\dot\gamma\,\mu\,R}{k_1} \in \{0.010,\ 0.025,\ 0.05,\ 0.1,\ 0.2\}$ by varying the membrane shear modulus k_1. The shear rate is $\dot\gamma = 1.310^{-5}$ in simulation units. To cross validate our results, we perform the same simulations with ESPResSo (FP32 for LBM, FP64 for IBM) [24], which has been cross validated with boundary-

FIG. 12. Illustration of the capsule in shear flow (FP32/FP32, $\mathrm{Ca} = 0.1$) simulation at dimensionless times $\dot\gamma\,t \in \{1,\ 2,\ 3,\ 4,\ 5\}$. Each image shows the simulation box from the side, with the top and bottom moving bounce-back boundaries marked in green. The capsule initially deforms to an elongated shape and then performs tank-treading, i.e., rotating the membrane while keeping its deformed shape.

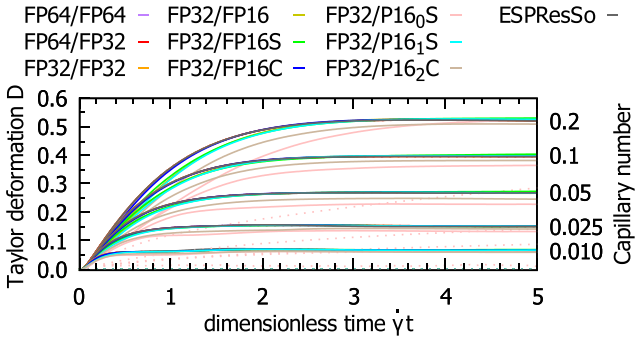


FIG. 13. Taylor deformation of the capsule first increases and then plateaus as the capsule starts tank-treading. This plateau depends on the Capillary number. Dashed lines represent corresponding simulations without DDF-shifting.

integral simulations and many others in Ref. [105]. In Fig. 13, we plot the Taylor deformation $D = \frac{a-c}{a+c}$ over time, with the largest and smallest semiaxes a and c of the deformed capsule [105]. We see that even in this complex scenario, the FP16C simulations produce physically accurate results with only insignificant deviations from FP64. The other 16-bit formats, especially posits, perform noticeably worse here. Without DDF-shifting, while FP32 still appears identical to ground truth, all 16-bit simulations do not produce the correct outcome (deformation remains close to zero). This emphasizes that DDF-shifting is essential for the lower precision formats.

F. Raindrop impact

Finally, we examine how number formats affect a volume-of-fluid LBM simulation of a 4 mm diameter raindrop impacting a deep pool at 8.8 $\frac{\mathrm{m}}{\mathrm{s}}$ terminal velocity. This system is described and extensively validated in Ref. [10] to study microplastic particle transport from the ocean into the atmosphere. The particles are simulated with the IBM. There, simulations are performed in FP32/FP32 with the maximum lattice size that fits into memory, so FP64 is not used here as it does not fit into the memory of a single GPU. The dimensionless numbers for this setup are Reynolds number $\mathrm{Re} = \frac{d\,u}{\nu} = 33498$, Weber number $\mathrm{We} = \frac{d\,u^2\,\rho}{\sigma} = 4301$, Froude number $\mathrm{Fr} = \frac{u}{\sqrt{d\,g}} = 44.4$, Capillary number $\mathrm{Ca} = \frac{u\,\rho\,\nu}{\sigma} = 0.1284$ and Bond number $\mathrm{Bo} = \frac{d^2\,\rho\,g}{\sigma} = 2.179$. The simulated domain is $464\times464\times394$ lattice points and runs on a single AMD Radeon VII GPU. The impact is simulated for 10 ms time, equivalent to 20 416 time steps in LBM units.

The raindrop impact is illustrated in Fig. 14. Note that the fully parallelized GPU implementation of the IBM with floating-point `atomic_add_f` makes the simulation nondeterministic [10,12] and that the exact breakup of the crown into droplets is expected to be randomly different every time. We see minor artifacts at the bottom of the cavity for FP32/P16$_1$S, but otherwise no qualitative differences in random crown breakup.

To be able to obtain statistics of ejected droplets and particles, we run the simulation 100 times each with FP32/FP32, FP32/FP16S, FP32/FP16C, and FP32/P16$_1$S. The microplastic particles each time are initialized at different

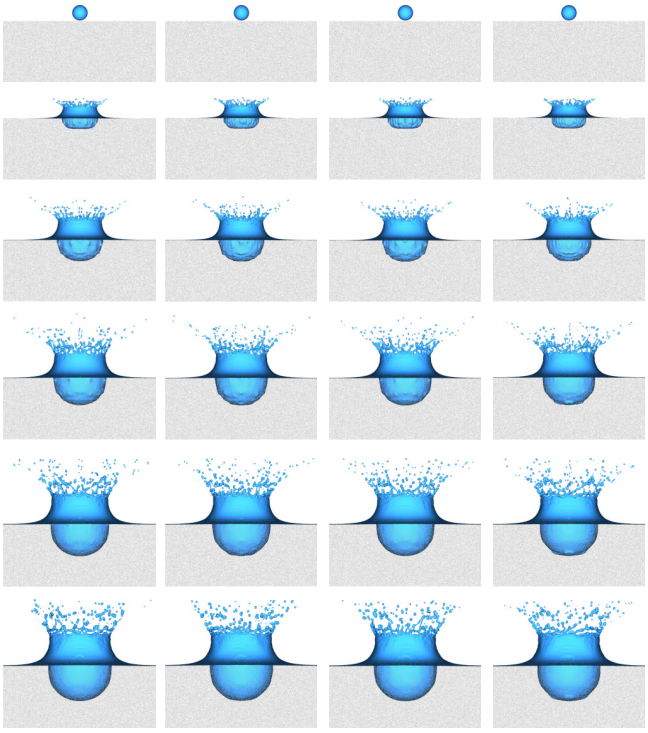


FIG. 14. A 4 mm diameter raindrop impacting a deep pool at 8.8 $\frac{\text{m}}{\text{s}}$ terminal velocity, illustrated at times $t \in \{0, 1, 2, 3, 4, 5\}$ ms after impact as used in Ref. [10].

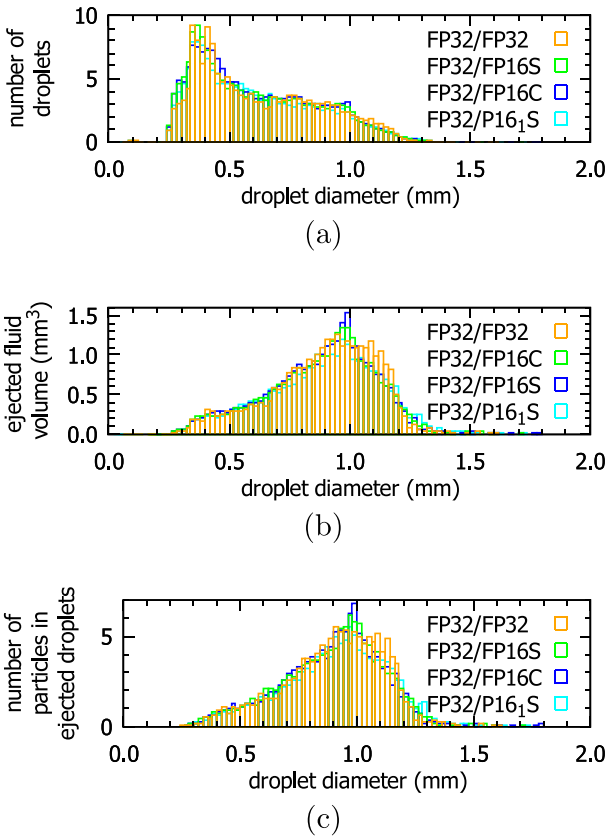


FIG. 15. (a) The size distribution of droplets, (b) the distribution of ejected fluid volume by droplet diameter, and (c) the distribution of microplastic particles in droplets for 100 simulations each conducted with FP32/FP32, FP32/FP16S, FP32/FP16C and FP32/P16$_1$S.

random positions, resulting in slightly different random crown breakup. Ejected droplets that touch the top of the simulation box are measured and then deleted as detailed in Ref. [10]. In histograms of the size, volume, and particle count depending on droplet diameter (Fig. 15), we see no significant differences across the data sets.

To conclude this section, we find that all FP32/FP16S, FP32/FP16C, and FP32/P16$_1$S are able to recreate the results of FP32/FP32 in raindrop impact simulations without negative impact on the accuracy of the results, while significantly reducing the memory footprint of these simulations. This in turn enables simulations higher lattice resolution, potentially increasing accuracy by resolving smaller droplets.

V. MEMORY AND PERFORMANCE COMPARISON

For GPUs, the most efficient streaming step implementation [63] is the One-Step-Pull scheme (AB-Pattern) with two copies of the DDFs in memory, because the noncoalesced memory read penalty is lower than the noncoalesced write penalty on GPUs [12,15,30,33,35,37,38,51,53,54], see Fig. 22 in the Appendix. One-Step-Pull further greatly facilitates implementing LBM extensions like Volume-of-Fluid, so it is a popular choice. Our FluidX3D base implementation (no-slip bounce-back boundaries, no extensions, as in Listing 2) with DdQq velocity set has memory requirements per lattice point as shown in Table III. For D3Q19, going from FP32/FP32 to FP32-16x reduces the memory footprint by ≈45%, to 93 bytes per node. If 16-bit compression was combined with in-place streaming schemes like AA-Pattern [34], Esoteric-Twist [62], Shift-and-Swap-Streaming [59], or the simple Esoteric-Pull [9], the memory footprint can even be reduced by ≈67%, to only 55 bytes per node.

Although our main goal with FP16 is to reduce memory footprint and allow for larger simulation domains, as a side effect, performance is vastly increased as a result of less memory transfer in every LBM time step. For our base implementation with the DdQq velocity set, the amount of memory transfers per lattice point per time step is shown in Table IV. Writing velocity and density to memory in each time step is not required for LBM without extensions. Theoretical speedup from FP32/FP32 to FP32/16-bit is 80% for all velocity sets and swap algorithms.

While most LBM implementations are limited to one particular hardware platform—either CPUs [67–79], Nvidia

TABLE III. Memory requirements in byte per lattice point of LBM floating-point variants for the One-Step-Pull swap algorithm with two copies of the DDFs for the DdQq velocity set.

	$\vec{u}$	ρ	flags	f_i	$\sum$
FP64/FP64	$8\,d$	8	1	$16\,q$	$8\,d+9+16\,q$
FP64/FP32	$8\,d$	8	1	$8\,q$	$8\,d+9+\ 8\,q$
FP32/FP32	$4\,d$	4	1	$8\,q$	$4\,d+5+\ 8\,q$
FP32/16-bit	$4\,d$	4	1	$4\,q$	$4\,d+5+\ 4\,q$

TABLE IV. Memory transfer in byte per lattice point per time step of LBM floating-point variants for the DdQq velocity set.

	flags	f_i	$\sum$
FP64/FP64	q	$16\,q$	$17\,q$
FP64/FP32	q	$8\,q$	$9\,q$
FP32/FP32	q	$8\,q$	$9\,q$
FP32/16-bit	q	$4\,q$	$5\,q$

GPUs [30–56], CPUs and Nvidia GPUs [18–29] or mobile SoCs [122,123]—only few use OpenCL [5–17]. With FluidX3D also being implemented in OpenCL, we are able to benchmark our code across a large variety of hardware, from the world's fastest data-center GPUs over gaming GPUs and CPUs to even the GPUs of mobile phone ARM SoCs. This enables us to determine LBM performance characteristics on various hardware microarchitectures. In Fig. 16, we show performance and efficiency on various hardware for D3Q19 SRT without extensions (only no-slip bounce-back boundaries are enabled in the code). The benchmark setup consists of a cubic box without any boundary nodes and with periodic boundary conditions in all directions. The standard domain size for the benchmark is 256^3, except where device memory is not enough; there we use the largest cubic box that fits into memory.

We group the tested devices into four classes with different performance characteristics:

(1) FP64-capable dedicated GPUs (high FP64:FP32 compute ratio) provide excellent efficiency for FP64/xx, FP32/FP32, and FP32/FP16S. They have such fast memory bandwidth that the FP32 ↔ FP16C software conversion brings FP32/FP16C from the bandwidth limit into the compute limit, reducing its efficiency.

(2) Non-FP64-capable dedicated GPUs (low FP64:FP32 compute ratio) have a particularly high FP32 arithmetic hardware limit, so even with the FP32 ↔ FP16C software conversion the algorithm remains in the memory bandwidth limit. FP32/xx efficiency is excellent except for older Nvidia Kepler. However, due to the poor FP64 arithmetic capabilities, FP64/xx efficiency is low as LBM here runs entirely in the compute limit rather than memory bandwidth limit. Surprisingly, FP64/FP32 runs even slower than FP64/FP64. This is because there is additional overhead for the FP64 ↔ FP32 cast conversion in the compute limit, despite less memory bandwidth being used.

(3) Integrated GPUs (iGPUs) overall show low performance and low efficiency. This is expected due to the slow system memory and cache hierarchy. Some older models do not support FP64 arithmetic at all.

(4) CPUs also show low performance and low efficiency. The low efficiency on CPUs is less of a property of the implementation or a result of OpenCL, and more related to CPU microarchitectures in general [67]. Other native CPU implementations of the LBM have equally low hardware efficiency [67,68,71,73] as a result of multilevel caching, inter-CPU communication, and other hardware properties unfavorable for LBM. To illustrate this further, our implementation runs about as fast on the Mali-G72 MP18 mobile phone GPU as CPU codes on between 2 and 16 cores, depending on the CPU model [19,23,27,67,68,71,73,86].

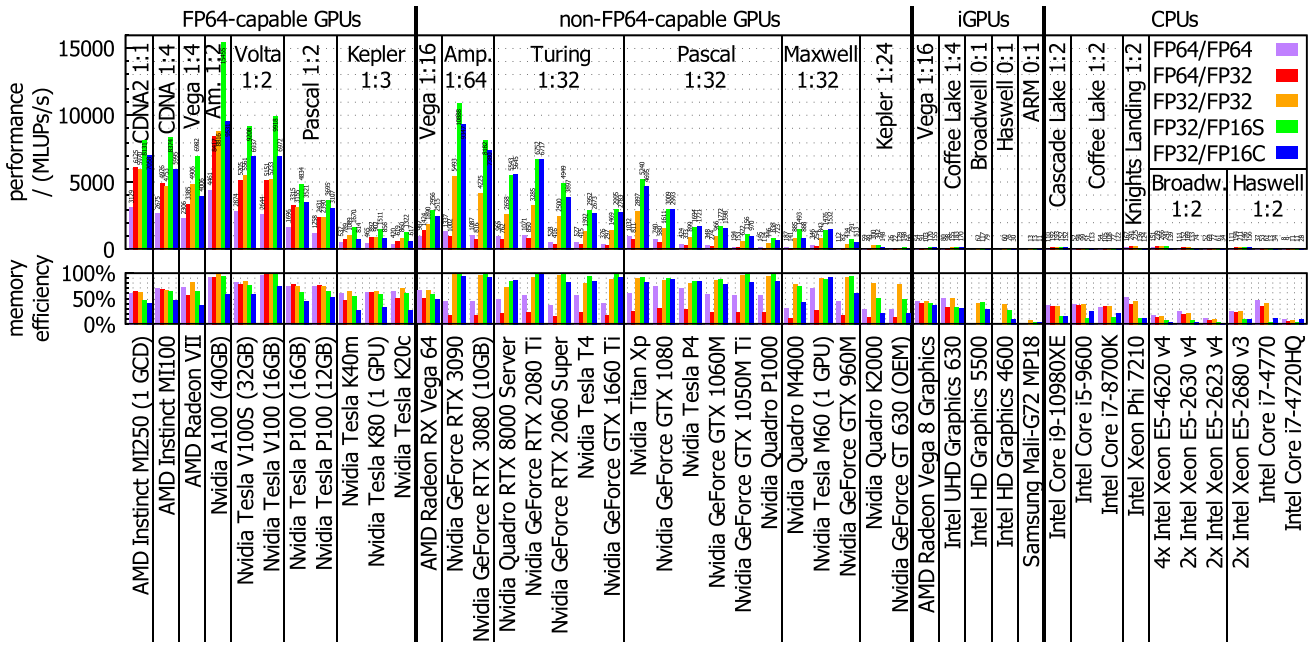


FIG. 16. Performance of FluidX3D with D3Q19 SRT on different hardware (code as in listing 2). The unit MLUPs/s is an acronym for mega lattice updates per second, meaning how many times 10^6 LBM lattice points are computed every second. To obtain the efficiency, we divide the measured MLUPs/s by the data sheet memory bandwidth times the number of bytes transferred per lattice point and time step (Table IV). Performance characteristics differ depending on the FP64 arithmetic capabilities as indicated by the FP64:FP32 compute ratios of the microarchitectures. The two GCDs of the MI250 are separate GPUs with 64 GB unified memory each, similar to dual-GPU cards such as the Tesla K80; driver 3423.0 (HSA1.1,LC) and ROCm 5.1.3 was used. CPU benchmarks are on all cores. Values in Table V.

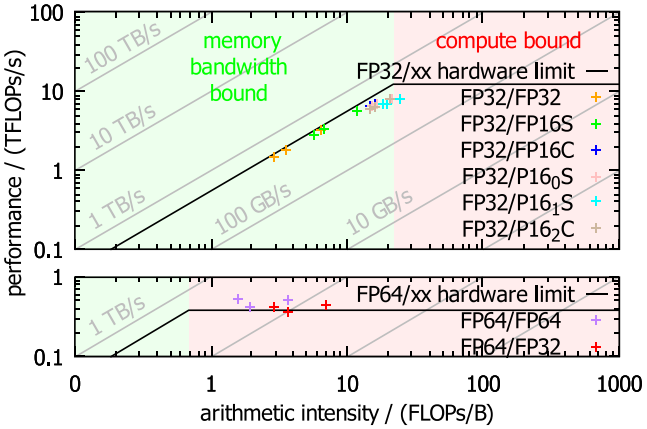


FIG. 17. Roofline model analysis of FluidX3D with the D3Q19 velocity set, running on an Nvidia Titan Xp GPU. For each floating-point type, the three data points (left to right) correspond to the SRT, TRT, and MRT collision operators. The arithmetic hardware limit is different for FP64/xx and FP32/xx, so we use two plots.

It is of note that performance on CPUs with large cache greatly depends on the domain size: If a large fraction of the domain fits into the L3 cache, efficiency (relative to memory bandwidth) is significantly better. Our CPU tests use a domain size of 256^3, so only an insignificant $\approx$1% is covered by L3 cache—a scenario representative of typical applications.

On the vast majority of hardware, we actually reach the theoretical 80% speedup as indicated by the hardware efficiency remaining equal for FP32/FP32 and FP32/FP16S. Some hardware, namely, the Nvidia Turing and Volta microarchitectures, do actually reach 100% efficiency with FP32/FP32 and FP32/FP16S. The Nvidia RTX 2080 Ti is at 100% efficiency even with FP32/FP16C, since the Nvidia Turing microarchitecture can do concurrent floating-point and integer computation and the 2080 Ti has high enough compute power per memory bandwidth to entirely remain in the memory bandwidth limit. Some efficiency values are even above 100% as Nvidia Turing and Ampere A100 are capable of memory compression to increase effective bandwidth beyond the memory specifications [124,125]. Nvidia Pascal GeForce and Titan GPUs (that lack ECC memory) lock into P2 power state with reduced memory clock for compute applications to prevent memory errors [126], lowering maximum bandwidth and making perfect (data sheet) efficiency impossible.

FP32/P16$_0$S and FP32/P16$_2$C performance is very similar to FP32/FP16C (data not shown), since the conversion needs to be emulated in software as well. FP32/P16$_1$S performance is a bit lower because the conversion algorithm is slightly more complex.

To better understand why performance is excellent with FP32/xx but not with FP64/xx on non-FP64-capable GPUs, we perform a roofline analysis [64,107] for the Nvidia Titan Xp in Fig. 17. The number of arithmetic operations and memory transfers is determined by automated counting of the corresponding PTX assembly instructions [112] of the stream-collide kernel. We note that we count the arithmetic intensity as the sum of floating-point and integer operations because the Pascal microarchitecture computes floating-point and integer on the same CUDA cores. For D3Q19 SRT FP32/FP32, for example, we count 255 floating-point operations and 248 integer operations. LBM performance scales proportionally to memory bandwidth, which is indicated by diagonal lines. The factor of proportionality is different for FPxx/64 (323 byte memory transfer per LBM time step), FPxx/32 (171 byte), and FPxx/16 (95 byte) as the amount of memory transfer is different (Table IV). FP16 reduces the number of memory transfers, so the arithmetic intensity (number of arithmetic operations divided by memory transfers) is increased. The manual conversion from and to FP16C significantly increases the number of arithmetic operations, further raising arithmetic intensity. Nevertheless, even with the arithmetic-heavy matrix multiplication of the MRT collision operator, all data points are still within the memory bandwidth limit and thus almost equally efficient compared to FP32. Actual memory clocks during the benchmark are 3.5% lower than the data sheet value (hardware limit) due to the Titan Xp locking into P2 power state [126], inhibiting perfect efficiency for FP32/xx. In contrast, FP64/xx is in the compute limit, greatly reducing performance. The data points in the compute limit can be a bit above the hardware limit if core clocks are boosted beyond official data sheet values.

VI. CONCLUSIONS

In this paper, we studied the consequences of the employed floating-point number format on accuracy and performance of lattice Boltzmann simulations. We used six different test systems ranging from simple, pure fluid cases (Poiseuille flow, Taylor-Green vortices, Karman vortex streets, lid-driven cavity) to more complex situations such as immersed-boundary simulations for a microcapsule in shear flow or a Volume-of-Fluid simulation of an impacting raindrop. For all of these, we thoroughly compared how FP64, FP32, FP16, and posit16 (mixed) precision affect the accuracy of the LBM. In the mixed variants, a higher precision floating-point format is used for arithmetics and a lower precision format is used for storing the DDFs. Based on the observation that a number range of ± 2 is sufficient for storing DDFs, we designed two customized 16-bit number formats specifically tailored to the needs of LBM simulations: a custom 16-bit floating-point format (FP16C) with halved truncation error compared to the standard IEEE-754 FP16 format by taking one bit from the exponent to increase the mantissa size and a specifically designed asymmetric posit variant (P16$_2$C). Conversion to these formats can be implemented highly efficiently and code interventions are only a few lines.

In all setups that we have tested and for the majority of parameters, FP32 turned out to be as accurate as FP64, provided that proper DDF-shifting [66] is used. Our custom FP16C format considerably diminished errors and noise and turned out to be a viable option for FP32/16-bit mixed precision in many cases. 16-bit posits with their variable precision have shown to be very compelling options too. Especially, P16$_1$S in some cases could beat our FP16C. In other cases, however, where the DDFs are outside the most favorable number range, the simulation error is increased significantly for the FP32/posit16 simulations.

Regarding performance, we find that pure FP64 runs very poorly on the vast majority of GPUs, with the exception of

very few data-center GPUs with extended FP64 arithmetic capabilities such as MI250/MI100/A100/V100(S)/P100. FP64/FP32 mixed precision can be almost as fast as pure FP32 on these special data-center GPUs. However, somewhat counterintuitively, on all GPUs with poor FP64 capabilities, FP64/FP32 is even slower than pure FP64 due to the conversion overhead. In general, pure FP32 then is a better choice since it enables excellent computational efficiency across all GPUs, especially considering that it is equally accurate to FP64 in all but edge cases. Computational efficiency is also excellent for FP32/FP16S mixed precision across all GPUs, reaching a maximum performance of 15455 MLUPs (D3Q19) on a single 40 GB Nvidia A100. On almost all GPUs that we have tested, we see the theoretical speedup of 80% that FP32/16-bit mixed precision offers for D3Q19, alongside 45% reduced memory footprint. Our custom format FP32/FP16C requires manual floating-point conversion which is heavy on integer computation. Nevertheless, FP32/FP16C runs efficiently on most GPUs with good FP32 arithmetic capabilities compared to their respective memory bandwidth and the theoretically expected 80% speedup can be achieved.

In conclusion, we show that pure FP32 precision is sufficient for most application scenarios of the LBM and that with our specifically tailored FP16C number format, in many cases even mixed FP32/FP16C precision can be used without significant loss of accuracy.

ACKNOWLEDGMENTS

We acknowledge support through the computational resources provided by BZHPC, LRZ, CINECA, and JSC JURECA-DC Evaluation Platform. We acknowledge the NVIDIA Corporation for donating a Titan Xp GPU and an A100 40GB GPU for our research. We thank M. Lehmann, M. Meinhart, and R. Kellnberger for running the benchmarks on their PCs. This study was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) No. SFB 1357–391977956. We further acknowledge funding from Deutsche Forschungsgemeinschaft in the framework of FOR 2688, Instabilities, Bifurcations and Migration in Pulsating Flow, Projects No. B3 (417989940) and No. B2 (417989464).

M.L. and M.K. contributed the original concept. M.L. conducted the simulations and wrote the paper. M.L., M.K., G.A., and S.G. contributed essential ideas, paper review, and literature research. M.L., G.A., M.S., and J.H. contributed test setups and benchmarks.

APPENDIX

1. The LB3D code

While in the most of this paper, we used the FluidX3D code [6–12], we also confirmed selected results with the LB3D lattice Boltzmann simulation package [79]. For this, we ported the FP64/FP64 routines to FP32/FP32 also in LB3D. LB3D is an MPI-based, general-purpose simulation package that includes various multicomponent and multiphase lattice Boltzmann methods, coupled to point particle molecular dynamics, discrete element methods [127], and immersed boundary [128,129] methods, as well as finite element solvers for advection-diffusion problems, including the Nernst-Planck equation [130]. For the Poiseuille test, we used second-order accurate, midgrid bounce-back boundary conditions.

2. LBM equations in a nutshell

The coloring indicates the level of precision for the equations below:
lower precision storage, conversion, higher precision arithmetic.

a. Without DDF-shifting

(1) Streaming:

$$f_i^{\text{temp}}(\vec{x},t) = f_i^{\text{A}}(\vec{x}-\vec{e}_i,\,t). \tag{A1}$$

(2) Collision (SRT):

$$\rho(\vec{x},t) = \sum_i f_i^{\text{temp}}(\vec{x},t), \tag{A2}$$

$$\vec{u}(\vec{x},t) = \frac{1}{\rho(\vec{x},t)} \sum_i \vec{c}_i\, f_i^{\text{temp}}(\vec{x},t), \tag{A3}$$

$$f_i^{\text{eq}}(\vec{x},t) = f_i^{\text{eq}}(\rho(\vec{x},t),\,\vec{u}(\vec{x},t)) = w_i\,\rho\cdot\left(\frac{(\vec{u}\circ\vec{c}_i)^2}{2\,c^4}+\frac{\vec{u}\circ\vec{c}_i}{c^2}+1-\frac{\vec{u}\circ\vec{u}}{2\,c^2}\right), \tag{A4}$$

$$f_i^{\text{B}}(\vec{x},\,t+\Delta t) = \left(1-\frac{\Delta t}{\tau}\right) f_i^{\text{temp}}(\vec{x},t) + \frac{\Delta t}{\tau}\, f_i^{\text{eq}}(\vec{x},t). \tag{A5}$$

b. With DDF-shifting

(1) Streaming:

$$f_i^{\text{temp}}(\vec{x},t) = f_i^{\text{A}}(\vec{x}-\vec{e}_i,\,t). \tag{A6}$$

(2) Collision (SRT):

$$\rho(\vec{x},t) = \left(\sum_i f_i^{\text{temp}}(\vec{x},t)\right) + 1, \tag{A7}$$

$$\vec{u}(\vec{x},t) = \frac{1}{\rho(\vec{x},t)} \sum_i \vec{c}_i \, f_i^{\text{temp}}(\vec{x},t), \tag{A8}$$

$$f_i^{\text{eq-shifted}}(\vec{x},t) = f_i^{\text{eq-shifted}}(\rho(\vec{x},t),\, \vec{u}(\vec{x},t)) = w_i\,\rho \cdot \left(\frac{(\vec{u}\circ\vec{c}_i)^2}{2\,c^4} - \frac{\vec{u}\circ\vec{u}}{2\,c^2} + \frac{\vec{u}\circ\vec{c}_i}{c^2}\right) + w_i\,(\rho - 1), \tag{A9}$$

$$f_i^{\text{B}}(\vec{x},\, t+\Delta t) = \left(1 - \frac{\Delta t}{\tau}\right) f_i^{\text{temp}}(\vec{x},t) + \frac{\Delta t}{\tau}\, f_i^{\text{eq}}(\vec{x},t). \tag{A10}$$

3. List of physical quantities and nomenclature

Quantity	SI-units	Defining equation(s)	Description
$\vec{x}$	m	$\vec{x} = (x, y, z)$	3D position in Cartesian coordinates
t	s	–	Time
Δx	m	$\Delta x := 1$	Lattice constant (in lattice units)
Δt	s	$\Delta t := 1$	Simulation time step (in lattice units)
c	$\frac{m}{s}$	$c := \frac{1}{\sqrt{3}} \frac{\Delta x}{\Delta t}$	Lattice speed of sound (in lattice units)
ρ	$\frac{kg}{m^3}$	$\rho = \sum_i f_i$	Mass density
$\vec{u}$	$\frac{m}{s}$	$\vec{u} = \sum_i \vec{c}_i f_i$	Velocity
f_i	$\frac{kg}{m^3}$	(A1)	Density distribution functions (DDFs)
f_i^{eq}	$\frac{kg}{m^3}$	(A4)	Equilibrium DDFs
i	1	$0 \leqslant i < q$	LBM streaming direction index
q	1	$q \in \{7, 9, 13, 15, 19, 27\}$	Number of LBM streaming directions
$\vec{c}_i$	$\frac{m}{s}$	[12], Eq. (11)	Streaming velocities
$\vec{e}_i$	m	$\vec{e}_i = \vec{c}_i\,\Delta t$	Streaming directions
w_i	1	[12], [Eq. (10)], $\sum_i w_i = 1$	Velocity set weights
τ	s	$\tau = \frac{\nu}{c^2} + \frac{\Delta t}{2}$	LBM relaxation time
ν	$\frac{m^2}{s}$	$\nu = \frac{\mu}{\rho}$	Kinematic shear viscosity
$\vec{f}$	$\frac{kg}{m^2\,s^2}$	$\vec{f} = \frac{\vec{F}}{V}$	Force per volume
(L_x, L_y, L_z)	(m, m, m)	$L_x\,L_y\,L_z = V$	Simulation box dimensions
g	$\frac{m}{s^2}$	$g := 9.81\,\frac{m}{s^2}$	Gravitational acceleration
σ	$\frac{kg}{s^2}$	–	Surface tension coefficient

4. Measured number format characteristics

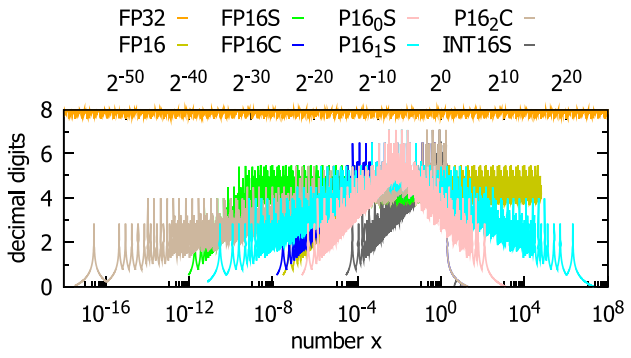
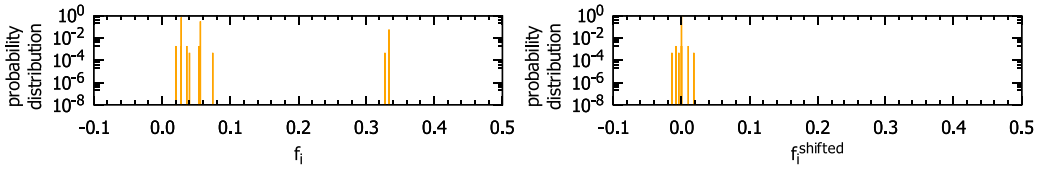
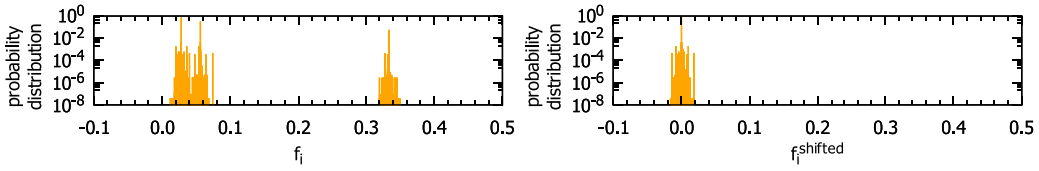


FIG. 18. Measured accuracy characteristics of the number formats investigated in this paper. The number of decimal digits for a given number x is computed via $-\log_{10}(|\log_{10}(\frac{x_{\text{represented}}}{x})|)$ [90,111,113]. Only the local minima are the relevant criterion for the error. Note that this definition for the number of decimal digits is off from the $\log_{10}(2^{n_m+1})$ definition by about 0.4.

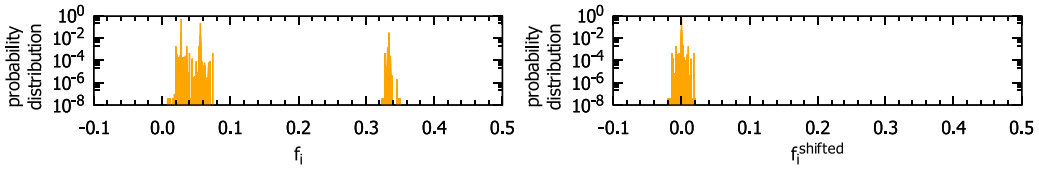
5. Numerical values of f_i and f_i^{shifted} for the lid-driven cavity (FP32/FP32)



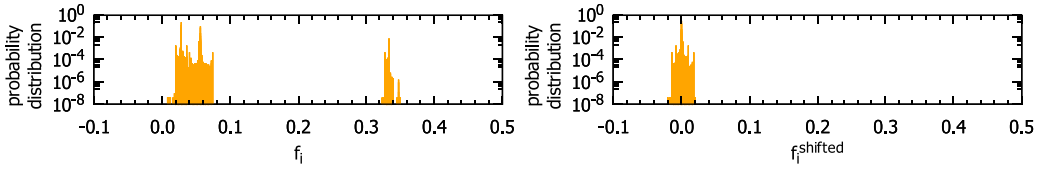
(a) Lid-driven cavity, after $t = 1$ time steps.



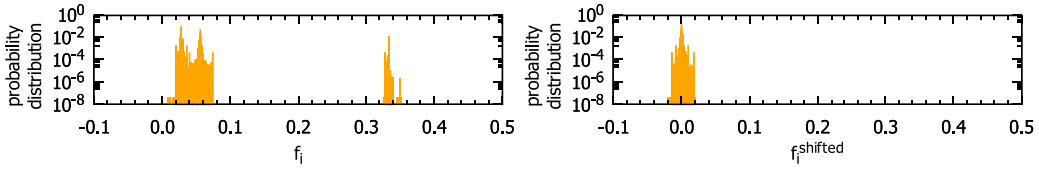
(b) Lid-driven cavity, after $t = 10$ time steps.



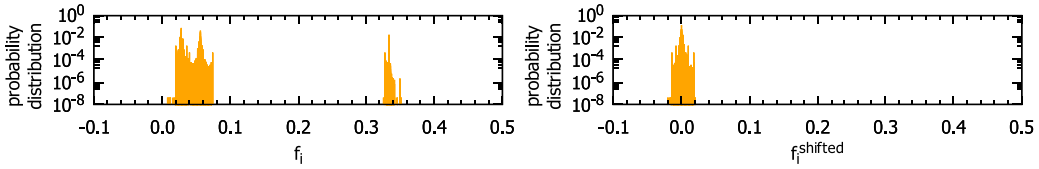
(c) Lid-driven cavity, after $t = 100$ time steps.



(d) Lid-driven cavity, after $t = 1000$ time steps.



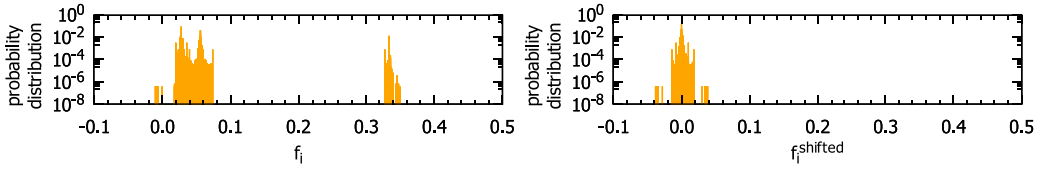
(e) Lid-driven cavity, after $t = 10000$ time steps.



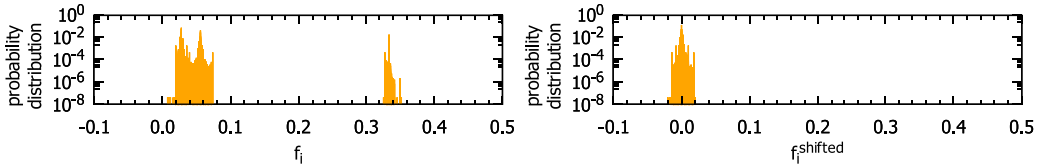
(f) Lid-driven cavity, after $t = 100000$ time steps.

FIG. 19. Numerical values of f_i and f_i^{shifted} for the lid-driven cavity (FP32/FP32, Re = 1000, Ma = 0.17, grid resolution $L = 128$) at various points in time.

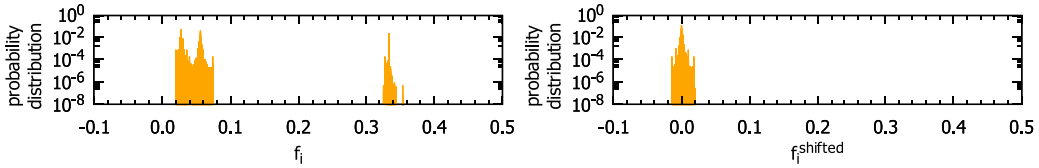
6. Numerical values of f_i and f_i^{shifted} for all setups (FP32/FP32)



(a) Lid-driven cavity at grid resolution $L = 64$.

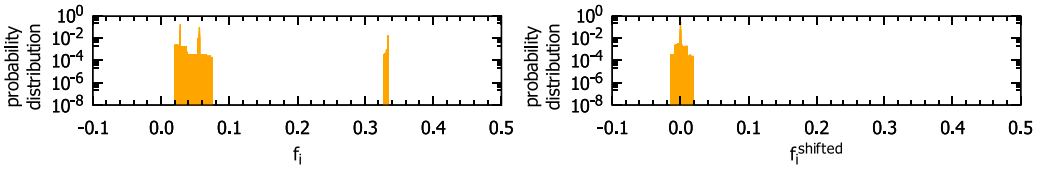


(b) Lid-driven cavity at grid resolution $L = 128$.

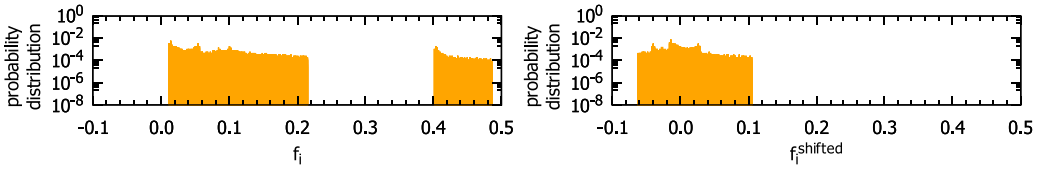


(c) Lid-driven cavity at grid resolution $L = 256$.

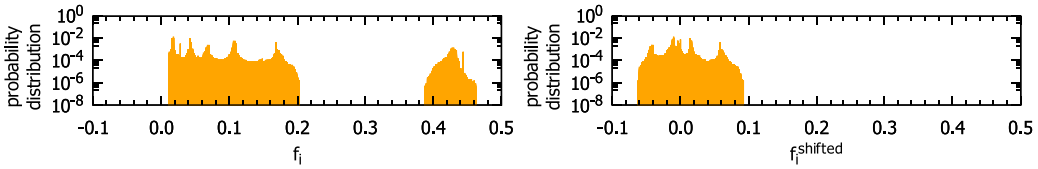
FIG. 20. Numerical values of f_i and f_i^{shifted} for the lid-driven cavity (FP32/FP32, Re $= 1000$, Ma $= 0.17$, after $t = 100\,000$ time steps) at various grid resolutions.



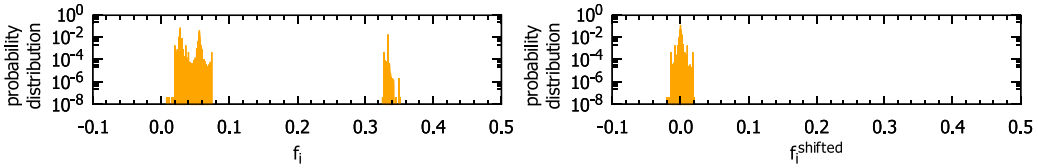
(a) Poiseuille flow, $R = 63$, $u_{max} = 0.1$, $Re = 10$, at the time of convergence, like in figure 6.



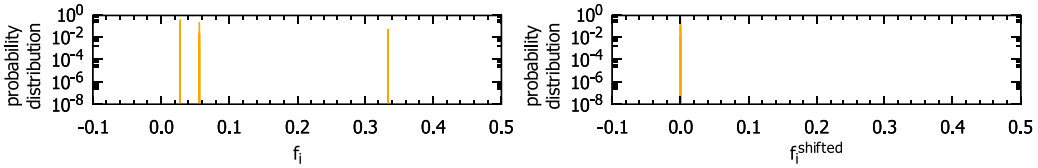
(b) Taylor-green vortices, at time of initialization $t = 0$.



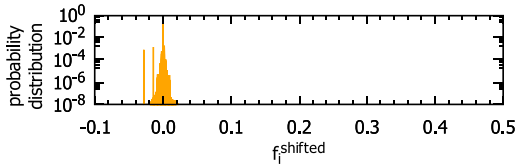
(c) Karman vortex street, after $t = 100000$ time steps.



(d) Lid-driven cavity, after $t = 100000$ time steps, like in figures 1 and 11.



(e) Microcapsule in shear flow at $Ca = 0.1$, after dimensionless time $\dot{\gamma}\, t = 5$.



(f) Raindrop impact at resolution $464 \times 464 \times 394$, with IBM particles, after $t = 10\,\mathrm{ms}$ time.

FIG. 21. Numerical values of f_i and f_i^{shifted} for all setups (FP32/FP32).

7. Properties of benchmarked hardware

TABLE V. Properties of benchmarked hardware.

Device	Data sheet				Measured memory bandwidth/ GB/s				LBM performance (D3Q19 SRT) / MLUPs/s				
	FP64/ TFLOPs/ s	FP32/ TFLOPs/ s	memory/ GB	bandwidth/ GB/ s	coalesced read	misaligned read	coalesced write	misaligned write	FP64/ FP64	FP64/ FP32	FP32/ FP32	FP32/ FP16S	FP32/ FP16C
AMD Instinct MI250 (1 GCD)	45.26	45.26	64	1638	969	1290	1034	634	3129	6125	5970	8131	7058
AMD Instinct MI100	11.54	46.14	32	1229	812	936	776	567	2675	4976	4753	8374	5995
AMD Radeon VII	3.46	13.83	16	1024	647	825	698	397	2306	3385	4906	6982	4006
Nvidia A100 (40GB)	9.75	19.49	40	1555	1334	1235	1514	207	4461	8417	8816	15455	9531
Nvidia Tesla V100S (32GB)	8.18	16.35	32	1134	911	930	951	153	2874	5205	5561	9208	6937
Nvidia Tesla V100 (16GB)	7.07	14.13	16	900	811	806	888	133	2644	5151	5233	9918	6972
Nvidia Tesla P100 (16GB)	4.76	9.52	16	732	504	306	544	90	1696	3315	3155	4834	3521
Nvidia Tesla P100 (12GB)	4.76	9.52	12	549	426	308	403	47	1258	2431	2390	3695	3107
Nvidia Tesla K40m	1.43	4.29	12	288	158	42	182	42	537	789	1089	1670	814
Nvidia Tesla K80 (1 GPU)	1.37	4.11	12	240	135	42	153	43	465	892	902	1511	856
Nvidia Tesla K20c	1.17	3.52	5	208	131	34	147	35	420	619	860	1322	617
AMD Radeon RX Vega 64	0.83	13.35	8	484	251	368	252	391	987	1424	1890	2956	2515
Nvidia GeForce RTX 3090	0.61	39.05	24	936	886	886	909	204	1337	1002	5493	10888	9347
Nvidia GeForce RTX 3080	0.47	29.77	10	760	692	693	709	171	1087	816	4225	8182	7380
Nvidia Quadro RTX 8000 Server	0.47	14.93	48	624	495	474	433	144	965	762	2658	5543	5645
Nvidia GeForce RTX 2080 Ti	0.42	13.45	11	616	533	531	579	159	1071	850	3285	6752	6717
Nvidia GeForce RTX 2060 S.	0.22	7.18	8	448	392	264	423	82	528	416	2500	4949	3897
Nvidia Tesla T4	0.25	8.14	16	300	259	264	226	85	527	415	1392	2952	2673
Nvidia GeForce GTX 1660 Ti	0.17	5.48	6	288	246	187	263	53	376	297	1469	2995	2783
Nvidia Titan Xp	0.38	12.15	12	548	445	163	514	112	1032	811	2897	5240	4695
Nvidia GeForce GTX 1080	0.31	9.78	8	320	257	201	275	80	740	580	1611	3009	2993
Nvidia Tesla P4	0.18	5.70	8	192	139	121	144	44	424	333	899	1694	1721
Nvidia GeForce GTX 1060M	0.14	4.44	6	192	156	104	165	44	348	274	966	1772	1598
Nvidia GeForce GTX 1050M Ti	0.08	2.49	4	112	98	65	102	27	194	153	622	1156	970
Nvidia Quadro P1000	0.06	1.89	4	82	70	47	73	21	145	114	446	838	723
Nvidia Quadro M4000	0.08	2.57	8	192	154	35	159	23	187	141	885	1493	888
Nvidia Tesla M60 (1 GPU)	0.15	4.82	8	160	143	56	145	39	346	254	842	1476	1532
Nvidia GeForce GTX 960M	0.05	1.51	4	80	73	31	70	14	112	83	434	791	513
Nvidia Quadro K2000	0.03	0.73	2	64	35	7	57	8	59	49	300	342	148
Nvidia GeForce GT 630 (OEM)	0.02	0.46	2	29	13	3	26	4	26	22	129	148	66
AMD Radeon Vega 8 Graphics	0.08	1.23	7	38	30	33	26	33	54	91	103	170	155
Intel UHD Graphics 630	0.12	0.46	7	51	38	34	28	18	80	98	155	183	170
Intel HD Graphics 5500	–	0.35	3	26	16	19	8	9	–	–	62	117	79
Intel HD Graphics 4600	–	0.38	2	26	19	10	11	7	–	–	60	76	30
Samsung Mali-G72 MP18	–	0.24	4	29	22	21	21	21	–	–	13	13	11
Intel Core i9-10980XE	1.81	3.23	128	94	60	65	26	30	110	192	193	152	152
Intel Core i5-9600	0.30	0.60	16	43	35	39	14	16	52	94	99	51	113
Intel Core i7-8700K	0.36	0.71	16	51	38	51	26	18	54	105	108	77	122
Intel Xeon Phi 7210	2.66	5.32	192	102	35	65	62	61	167	241	275	136	124
4x Intel Xeon E5-4620 v4	1.34	2.69	512	273	104	126	52	54	151	239	266	241	139
2x Intel Xeon E5-2630 v4	0.70	1.41	64	137	65	88	66	86	111	152	169	133	74
2x Intel Xeon E5-2623 v4	0.33	0.67	64	137	25	43	24	26	52	69	77	61	34
2x Intel Xeon E5-2680 v3	0.96	1.92	64	137	110	102	43	50	111	194	211	146	156
Intel Core i7-4770	0.22	0.44	16	26	24	26	15	12	37	53	61	13	34
Intel Core i7-4720HQ	0.17	0.33	16	26	22	22	9	11	8	9	11	11	28

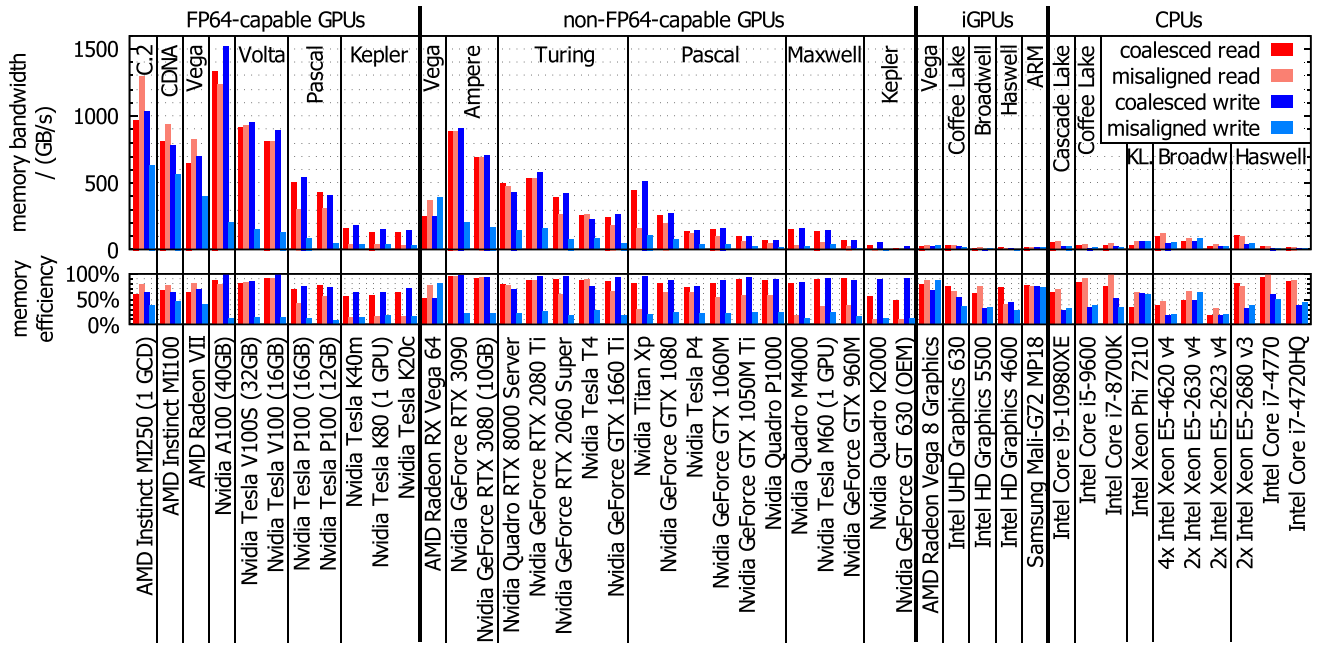


FIG. 22. Synthetic OpenCL memory benchmarks to measure coalesced/misaligned read/write performance. The misaligned write penalty is much larger than the misaligned read penalty across almost all tested devices. Values in Table V.

9. Ultrafast conversion algorithms

```
// FP32/FP32 macros
#define fpXX float // switchable data type
#define load(p,o) p[o] // regular float read
#define store(p,o,x) p[o]=x // regular float write

// FP32/FP16S macros
#define fpXX half // switchable data type
#define load(p,o) vload_half(o,p)*3.0517578E-5f // use OpenCL function
#define store(p,o,x) vstore_half_rte((x)*32768.0f,o,p) // use OpenCL function

// FP32/FP16C macros
#define fpXX ushort // switchable data type
#define load(p,o) half_to_float_custom(p[o]) // call conversion function
#define store(p,o,x) p[o]=float_to_half_custom(x) // call conversion function

// FP32/P160S macros
#define fpXX ushort // switchable data type
#define load(p,o) p160_to_float(p[o])*0.0078125f // call conversion function
#define store(p,o,x) p[o]=float_to_p160((x)*128.0f) // call conversion function

// FP32/P161S macros
#define fpXX ushort // switchable data type
#define load(p,o) p161_to_float(p[o])*0.0078125f // call conversion function
#define store(p,o,x) p[o]=float_to_p161((x)*128.0f) // call conversion function

// FP32/P162C macros
#define fpXX ushort // switchable data type
#define load(p,o) p162C_to_float(p[o]) // call conversion function
#define store(p,o,x) p[o]=float_to_p162C(x) // call conversion function



ushort float_to_half_custom(const float x) {
  const uint b = as_uint(x)+0x00000800; // round-to-nearest-even: add last bit after truncated mantissa
  const uint e = (b&0x7F800000)>>23; // exponent
  const uint m = b&0x007FFFFF; // mantissa; in line below: 0x007FF800 = 0x00800000-0x00000800 = decimal indicator flag - initial rounding
  return (b&0x80000000)>>16 | (e>112)*((((e-112)<<11)&0x7800)|m>>12) | ((e<113)&(e>100))*((((0x007FF800+m)>>(124-e))+1)>>1) | (e>127)*0x7FFF; // sign | normalized | denormalized | saturate
}
float half_to_float_custom(const ushort x) {
  const uint e = (x&0x7800)>>11; // exponent
  const uint m = (x&0x07FF)<<12; // mantissa
  const uint v = as_uint((float)m)>>23; // evil log2 bit hack to count leading zeros in denormalized format
  return as_float((x&0x8000)<<16 | (e!=0)*((e+112)<<23|m) | ((e==0)&(m!=0))*((v-37)<<23|((m<<(150-v))&0x007FF000))); // sign | normalized | denormalized
}



ushort float_to_p160(const float x) {
  const uint b = as_uint(x);
  const int e = ((b&0x7F800000)>>23)-127; // exponent-bias
  int m = (b&0x007FFFFF)>>9; // mantissa
  const int v = abs(e); // shift
  const int r = (e<0 ? 0x0002 : 0xFFFE)<<(13-v); // generate regime bits
  m = ((m>>(v-(e<0)))+1+(e<-13)*0x2)>>1; // rounding: add 1 after truncated position; in case of lowest numbers, saturate
  return (b&0x80000000)>>16 | (e>-16)*((r+m)&0x7FFF) | (e>13)*0x7FFF; // sign | regime+mantissa ("+" handles rounding overflow) | saturate
}
```

```
float p160_to_float(const ushort x) {
  const uint sr = (x>>14)&1; // sign of regime
  ushort t = x<<2; // remove sign and first regime bit
  t = sr ? ~t : t; // positive regime r>=0 : negative regime r<0
  const int r = 142-(as_int((float)t)>>23); // evil log2 bit hack to count leading zeros for regime
  const uint m = (x<<(r+10))&0x007FFFFF; // extract mantissa and bit-shift it in place
  const int rs = sr ? r : -r-1; // negative regime r<0 : positive regime r>=0
  return as_float((x&0x8000)<<16 | (r!=158)*((rs+127)<<23 | m)); // sign | regime | mantissa
}



ushort float_to_p161(const float x) {
  const uint b = as_uint(x);
  const int e = ((b&0x7F800000)>>23)-127; // exponent-bias
  int m = (b&0x007FFFFF)>>10; // mantissa
  const int ae = abs(e);
  const int v = ae>>1; // shift, ">>1" is the same as "/2"
  const int e2 = ae&1; // "&1" is the same as "%2"
  const int r = ((e<0 ? 0x0002 : 0xFFFE<<e2)+e2)<<(13-v-e2); // generate regime bits, merge regime+exponent and shift in
      ↪ place
  m = ((m>>(v-(e<0)*(1-e2)))+(e>-28)+(e<-26)*0x3)>>1; // rounding: add 1 after truncated position; in case of lowest
      ↪ numbers, saturate
  return (b&0x80000000)>>16 | (e>-31)*((r+m)&0x7FFF) | (e>26)*0x7FFF; // sign | regime+exponent+mantissa ("+" handles
      ↪ rounding overflow) | saturate
}
float p161_to_float(const ushort x) {
  const uint sr = (x>>14)&1; // sign of regime
  ushort t = x<<2; // remove sign and first regime bit
  t = sr ? ~t : t; // positive regime r>=0 : negative regime r<0
  const int r = 142-(as_int((float)t)>>23); // evil log2 bit hack to count leading zeros for regime
  const uint e = (x>>(12-r))&1; // extract mantissa and bit-shift it in place
  const uint m = (x<<(r+11))&0x007FFFFF; // extract mantissa and bit-shift it in place
  const int rs = (sr ? r : -r-1)<<1; // negative regime r<0 : positive regime r>=0, "<<1" is the same as "*2"
  return as_float((x&0x8000)<<16 | (r!=158)*((rs+e+127)<<23 | m)); // sign | regime+exponent | mantissa
}



ushort float_to_p162C(const float x) {
  const int b = as_int(x);
  const int e = ((b&0x7F800000)>>23)-127; // exponent-bias
  int m = (b&0x007FFFFF)>>10; // mantissa
  const int ae = -e;
  const int v = ae>>2; // shift, ">>2" is the same as "/4"
  const int r = 0x4000>>v; // generate regime bits, merge regime+exponent and shift in place
  const int e4 = (3-(ae&3))<<(12-v); // generate exponent and shift in place
  m = ((m>>v)+(e>-54)+(e<-51)*0x3)>>1; // rounding: add 1 after truncated position; in case of lowest numbers, saturate
  return (b&0x80000000)>>16 | (e>-59)*((r+e4+m)&0x7FFF) | (e>0)*0x7FFF; // sign | regime+exponent+mantissa ("+" handles
      ↪ rounding overflow) | saturate
}
float p162C_to_float(const ushort x) {
  const int r = 158-(as_int((float)((uint)x<<17))>>23); // remove sign bit, evil log2 bit hack to count leading zeros for
      ↪ regime
  const int e = (x>>(12-r))&3; // extract mantissa and bit-shift it in place
  const int m = (x<<(r+11))&0x007FFFFF; // extract mantissa and bit-shift it in place
  return as_float((x&0x8000)<<16 | (r!=158)*((124-(r<<2)+e)<<23 | m)); // sign | regime+exponent | mantissa
}
```

Listing 1: OpenCL C macros for regular FP32, for FP16S using hardware-accelerated IEEE-754 FP16 floating-point conversion and for our FP16C format with calls to our manual floating-point conversion functions. Manual floating-point conversion functions for FP32 ↔ FP16C (`float↔half`) in OpenCL C. We also provide macros and conversion algorithms for FP32 ↔ $P16_0S/P16_1S/P16_2C$ posit formats. The saturation term in the algorithms can be omitted if it is made sure that larger than maximum numbers are never used, which is the case in this LBM application.

10. LBM core of the FluidX3D OpenCL C implementation (D3Q19 SRT FP32/xx)

```
float __attribute__((always_inline)) sq(const float x) {
  return x*x;
}
uint3 __attribute__((always_inline)) coordinates(const uint n) { // disassemble 1D index to 3D coordinates (n -> x,y,z)
  const uint t = n%(def_sx*def_sy);
  return (uint3)(t%def_sx, t/def_sx, n/(def_sx*def_sy)); // n = x+(y+z*sy)*sx

uint __attribute__((always_inline)) f_index(const uint n, const uint i) { // 32-bit indexing (maximum box size for D3Q19: 608x608x608)
  return i*def_s+n; // SoA (229% faster on GPU compared to AoS)
}
void __attribute__((always_inline)) equilibrium(const float rho, float ux, float uy, float uz, float* feq) { // calculate f_equilibrium
  const float c3 = -3.0f*(sq(ux)+sq(uy)+sq(uz)), rhom1=rho-1.0f; // c3=-2*sq(u)/(2*sq(c))
  ux *= 3.0f;
  uy *= 3.0f;
  uz *= 3.0f;
  feq[ 0] = def_w0*fma(rho, 0.5f*c3, rhom1); // 000 (identical for all velocity sets)
  const float u0=ux+uy, u1=ux+uz, u2=uy+uz, u3=ux-uy, u4=ux-uz, u5=uy-uz;
  const float rhos=def_ws*rho, rhoe=def_we*rho, rhom1s=def_ws*rhom1, rhom1e=def_we*rhom1;
  feq[ 1] = fma(rhos, fma(0.5f, fma(ux, ux, c3), ux), rhom1s); feq[ 2] = fma(rhos, fma(0.5f, fma(ux, ux, c3), -ux), rhom1s); // +00 -00
  feq[ 3] = fma(rhos, fma(0.5f, fma(uy, uy, c3), uy), rhom1s); feq[ 4] = fma(rhos, fma(0.5f, fma(uy, uy, c3), -uy), rhom1s); // 0+0 0-0
  feq[ 5] = fma(rhos, fma(0.5f, fma(uz, uz, c3), uz), rhom1s); feq[ 6] = fma(rhos, fma(0.5f, fma(uz, uz, c3), -uz), rhom1s); // 00+ 00-
  feq[ 7] = fma(rhoe, fma(0.5f, fma(u0, u0, c3), u0), rhom1e); feq[ 8] = fma(rhoe, fma(0.5f, fma(u0, u0, c3), -u0), rhom1e); // ++0 --0
  feq[ 9] = fma(rhoe, fma(0.5f, fma(u1, u1, c3), u1), rhom1e); feq[10] = fma(rhoe, fma(0.5f, fma(u1, u1, c3), -u1), rhom1e); // +0+ -0-
  feq[11] = fma(rhoe, fma(0.5f, fma(u2, u2, c3), u2), rhom1e); feq[12] = fma(rhoe, fma(0.5f, fma(u2, u2, c3), -u2), rhom1e); // 0++ 0--
  feq[13] = fma(rhoe, fma(0.5f, fma(u3, u3, c3), u3), rhom1e); feq[14] = fma(rhoe, fma(0.5f, fma(u3, u3, c3), -u3), rhom1e); // +-0 -+0
  feq[15] = fma(rhoe, fma(0.5f, fma(u4, u4, c3), u4), rhom1e); feq[16] = fma(rhoe, fma(0.5f, fma(u4, u4, c3), -u4), rhom1e); // +0- -0+
  feq[17] = fma(rhoe, fma(0.5f, fma(u5, u5, c3), u5), rhom1e); feq[18] = fma(rhoe, fma(0.5f, fma(u5, u5, c3), -u5), rhom1e); // 0+- 0-+
}
void __attribute__((always_inline)) fields(const float* f, float* rhon, float* uxn, float* uyn, float* uzn) { // calculate density and velocity from fi
  float rho=f[0], ux, uy, uz;
  #pragma unroll
  for(uint i=1; i<def_set; i++) rho += f[i]; // calculate density from f
  rho += 1.0f; // add 1.0f last to avoid digit extinction effects when summing up f
  ux = f[ 1]-f[ 2]+f[ 7]-f[ 8]+f[ 9]-f[10]+f[13]-f[14]+f[15]-f[16]; // calculate velocity from fi (alternating + and - for best accuracy)
  uy = f[ 3]-f[ 4]+f[ 7]-f[ 8]+f[11]-f[12]+f[14]-f[13]+f[17]-f[18];
  uz = f[ 5]-f[ 6]+f[ 9]-f[10]+f[11]-f[12]+f[16]-f[15]+f[18]-f[17];
  *rhon = rho;
  *uxn = ux/rho;
  *uyn = uy/rho;
  *uzn = uz/rho;
}
void __attribute__((always_inline)) neighbors(const uint n, uint* j) { // calculate neighbor indices
  const uint3 xyz = coordinates(n);
  const uint x0 =   xyz.x; // pre-calculate indices (periodic boundary conditions on simulation box walls)
  const uint xp =  (xyz.x       +1)%def_sx;
  const uint xm =  (xyz.x+def_sx-1)%def_sx;
  const uint y0 =   xyz.y                  *def_sx;
```

```
48    const uint yp = ((xyz.y       +1)%def_sy)*def_sx;
49    const uint ym = ((xyz.y+def_sy-1)%def_sy)*def_sx;
50    const uint z0 =   xyz.z                  *def_sy*def_sx;
51    const uint zp = ((xyz.z       +1)%def_sz)*def_sy*def_sx;
52    const uint zm = ((xyz.z+def_sz-1)%def_sz)*def_sy*def_sx;
53    j[0] = n;
54    j[ 1] = xp+y0+z0; j[ 2] = xm+y0+z0; // +00 -00
55    j[ 3] = x0+yp+z0; j[ 4] = x0+ym+z0; // 0+0 0-0
56    j[ 5] = x0+y0+zp; j[ 6] = x0+y0+zm; // 00+ 00-
57    j[ 7] = xp+yp+z0; j[ 8] = xm+ym+z0; // ++0 --0
58    j[ 9] = xp+y0+zp; j[10] = xm+y0+zm; // +0+ -0-
59    j[11] = x0+yp+zp; j[12] = x0+ym+zm; // 0++ 0--
60    j[13] = xp+ym+z0; j[14] = xm+yp+z0; // +-0 -+0
61    j[15] = xp+y0+zm; j[16] = xm+y0+zp; // +0- -0+
62    j[17] = x0+yp+zm; j[18] = x0+ym+zp; // 0+- 0-+
63  }
64  kernel void initialize(global fpXX* fc, global float* rho, global float* u) {
65    const uint n = get_global_id(0); // n = x+(y+z*sy)*sx
66    float feq[def_set]; // f_equilibrium
67    equilibrium(rho[n], u[n], u[def_s+n], u[2*def_s+n], feq);
68    #pragma unroll
69    for(uint i=0; i<def_set; i++) store(fc, f_index(n,i), feq[i]); // write to fc
70  } // initialize()
71  kernel void stream_collide(const global fpXX* fc, global fpXX* fs, global float* rho, global float* u, global uchar* flags
        ↪ ) {
72    const uint n = get_global_id(0); // n = x+(y+z*sy)*sx
73    const uchar flagsn = flags[n]; // cache flags[n] for multiple readings
74    if(flagsn&TYPE_W) return; // if node is boundary node, just return (slight speed up)
75    uint j[def_set]; // neighbor indices
76    neighbors(n, j); // calculate neighbor indices
77    uchar flagsj[def_set]; // cache neighbor flags for multiple readings
78    flagsj[0] = flagsn;
79    #pragma unroll
80    for(uint i=1; i<def_set; i++) flagsj[i] = flags[j[i]];
81    // read from fc in video memory and stream to fhn
82    float fhn[def_set]; // cache f_half_step[n], do streaming step
83    fhn[0] = fc[f_index(n, 0)]; // keep old center population
84    #pragma unroll
85    for(uint i=1; i<def_set; i+=2) { // perform streaming
86      fhn[i  ] = load(fc, flagsj[i+1]&TYPE_W ? f_index(n, i+1) : f_index(j[i+1], i  )); // boundary : regular
87      fhn[i+1] = load(fc, flagsj[i  ]&TYPE_W ? f_index(n, i  ) : f_index(j[i  ], i+1));
88    }
89    // collide fh
90    float rhon, uxn, uyn, uzn; // cache density and velocity for multiple writings/readings
91    fields(fhn, &rhon, &uxn, &uyn, &uzn); // calculate density and velocity fields from f
92    uxn = clamp(uxn, -def_c, def_c); // limit velocity (for stability purposes)
93    uyn = clamp(uyn, -def_c, def_c);
94    uzn = clamp(uzn, -def_c, def_c);
95    float feq[def_set]; // cache f_equilibrium[n]
96    equilibrium(rhon, uxn, uyn, uzn, feq); // calculate equilibrium populations
97    #pragma unroll
98    for(uint i=0; i<def_set; i++) store(fs, f_index(n,i), fma(1.0f-def_w, fhn[i], def_w*feq[i])); // write to fs in video
          ↪ memory
99  } // stream_collide()
```

Listing 2: LBM core of the FluidX3D OpenCL C implementation (D3Q19 SRT FP32/xx).

[1] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen, in *The Lattice Boltzmann Method* (Springer International Publishing, Cham, Switzerland, 2017), Vol. 10, p. 978.

[2] S. Chapman, T. G. Cowling, and D. Burnett, *The Mathematical Theory of Non-Uniform Gases: An Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases* (Cambridge University Press, Cambridge, UK, 1990).

[3] H. Pradipto and A. Purqon, Accuracy and numerical stability analysis of lattice Boltzmann method with multiple relaxation time for incompressible flows, in J. Phys.: Conf. Ser. (IOP Publishing, Bristol, UK, 2017), Vol. 877, p. 012035.

[4] R. Benzi, S. Succi, and M. Vergassola, The lattice Boltzmann equation: Theory and applications, Phys. Rep. **222**, 145 (1992).

[5] P. M. Tekic, J. B. Radjenovic, and M. Rackovic, Implementation of the lattice Boltzmann method on heterogeneous hardware and platforms using OpenCL, Adv. Electr. Comput. Eng. **12**, 51 (2012).

[6] F. Häusl, MPI-based multi-GPU extension of the lattice Boltzmann method, Bachelor's Thesis, University of Bayreuth (2019), https://epub.uni-bayreuth.de/5689/.

[7] F. Häusl, Soft objects in newtonian and non-Newtonian fluids: A computational study of bubbles and capsules in flow, Master's Thesis, University of Bayreuth (2022), https://epub.uni-bayreuth.de/5960/.

[8] M. Lehmann and S. Gekle, Analytic solution to the piecewise linear interface construction problem and its application in curvature calculation for volume-of-fluid simulation codes, Computation **10**, 21 (2022).

[9] M. Lehmann, Esoteric Pull and Esoteric Push: Two simple in-place streaming schemes for the lattice Boltzmann method on GPUs, Computation **10**, 92 (2022).

[10] M. Lehmann, L. M. Oehlschlägel, F. P. Häusl, A. Held, and S. Gekle, Ejection of marine microplastics by raindrops: A computational and experimental study, Microplast. Nanoplast. **1**, 1 (2021).

[11] H. Laermanns, M. Lehmann, M. Klee, M. G. Löder, S. Gekle, and C. Bogner, Tracing the horizontal transport of microplastics on rough surfaces, Microplast. Nanoplast. **1**, 11 (2021).

[12] M. Lehmann, High performance free surface LBM on GPUs, Master's Thesis, University of Bayreuth (2019), https://epub.uni-bayreuth.de/5400/.

[13] M. Schreiber, P. Neumann, S. Zimmer, and H.-J. Bungartz, Free-surface lattice-Boltzmann simulation on many-core architectures, Proc. Comput. Sci. **4**, 984 (2011).

[14] M. Holzer, M. Bauer, and U. Rüde, Highly efficient lattice-Boltzmann multiphase simulations of immiscible fluids at high-density ratios on CPUs and GPUs through code generation, arXiv:2012.06144.

[15] M. Takáč and I. Petráš, Cross-platform GPU-based implementation of lattice Boltzmann method solver using ArrayFire library, Mathematics **9**, 1793 (2021).

[16] M. Q. Ho, C. Obrecht, B. Tourancheau, B. D. de Dinechin, and J. Hascoet, Improving 3D lattice Boltzmann method stencil with asynchronous transfers on many-core processors, in *Proceedings of the 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)* (IEEE, San Diego, California, 2017), pp. 1–9.

[17] J. Habich, C. Feichtinger, H. Köstler, G. Hager, and G. Wellein, Performance engineering for the lattice Boltzmann method on GPGPUs: Architectural requirements and performance results, Comput. Fluids **80**, 276 (2013).

[18] C. Riesinger, A. Bakhtiari, M. Schreiber, P. Neumann, and H.-J. Bungartz, A holistic scalable implementation approach of the lattice Boltzmann method for CPU/GPU heterogeneous clusters, Computation **5**, 48 (2017).

[19] E. O. Aksnes and A. C. Elster, Porous rock simulations and lattice Boltzmann on GPUs, in *Parallel Computing: From Multicores and GPU's to Petascale* (IOS Press, Amsterdam, Netherlands, 2010), pp. 536–545.

[20] A. Kummerländer, M. Dorn, M. Frank, and M. J. Krause, Implicit propagation of directly addressed grids in lattice Boltzmann methods (2021), doi: 10.13140/RG.2.2.35085.87523.

[21] M. Geveler, D. Ribbrock, D. Göddeke, and S. Turek, Lattice-Boltzmann simulation of the shallow-water equations with fluid-structure interaction on multi-and manycore processors, in *Facing the Multicore-Challenge* (Springer, Wiesbaden, Germany, 2010), pp. 92–104.

[22] J. Bény, C. Kotsalos, and J. Latt, Toward full GPU implementation of fluid-structure interaction, in *Proceedings of the 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)* (IEEE, Amsterdam, 2019), pp. 16–22.

[23] G. Boroni, J. Dottori, and P. Rinaldi, Full GPU implementation of lattice-Boltzmann methods with immersed boundary conditions for fast fluid simulations, Int. J. Multiphys. **11**, 1 (2017).

[24] M. Griebel, M. A. Schweitzer *et al.*, *Meshfree Methods for Partial Differential Equations II* (Springer, Cham, Switzerland, 2005).

[25] H.-J. Limbach, A. Arnold, B. A. Mann, and C. Holm, ESPResSo—an extensible simulation package for research on soft matter systems, Comput. Phys. Commun. **174**, 704 (2006).

[26] Institute for Computational Physics, Universität Stuttgart, ESPResSo user's guide, http://espressomd.org/wordpress/wp-content/uploads/2016/07/ug_07_2016.pdf (2016), accessed June 15, 2018.

[27] S. D. Walsh, M. O. Saar, P. Bailey, and D. J. Lilja, Accelerating geoscience and engineering system simulations on graphics hardware, Comput. Geosci. **35**, 2353 (2009).

[28] S. Zitz, A. Scagliarini, and J. Harting, Lattice Boltzmann simulations of stochastic thin film dewetting, Phys. Rev. E **104**, 034801 (2021).

[29] C. Wei, W. Zhenghua, L. Zongzhe, Y. Lu, and W. Yongxian, An improved lbm approach for heterogeneous gpu-cpu clusters, in *Proceedings of the 2011 4th International Conference on Biomedical Engineering and Informatics (BMEI)* (IEEE, Shanghai, China, 2011), Vol. 4, pp. 2095–2098.

[30] M. J. Mawson and A. J. Revell, Memory transfer optimization for a lattice Boltzmann solver on Kepler architecture nVidia GPUs, Comput. Phys. Commun. **185**, 2566 (2014).

[31] J. Tölke and M. Krafczyk, TeraFLOP computing on a desktop PC with GPUs for 3D CFD, Int. J. Comput. Fluid Dyn. **22**, 443 (2008).

[32] G. Herschlag, S. Lee, J. S. Vetter, and A. Randles, GPU data access on complex geometries for D3Q19 lattice Boltzmann method, in *Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (IEEE, Vancouver, BC, Canada, 2018), pp. 825–834.

[33] N. Delbosc, J. L. Summers, A. Khan, N. Kapur, and C. J. Noakes, Optimized implementation of the lattice Boltzmann method on a graphics processing unit towards real-time fluid simulation, Comput. Math. Appl. **67**, 462 (2014).

[34] P. Bailey, J. Myre, S. D. Walsh, D. J. Lilja, and M. O. Saar, Accelerating lattice Boltzmann fluid flow simulations using

graphics processors, in *Proceedings of the 2009 International Conference on Parallel Processing* (IEEE, Vienna, Austria, 2009), pp. 550–557.

[35] C. Obrecht, F. Kuznik, B. Tourancheau, and J.-J. Roux, Multi-GPU implementation of the lattice Boltzmann method, Comput. Math. Appl. **65**, 252 (2013).

[36] W. B. de Oliveira Jr, A. Lugarini, and A. T. Franco, Performance analysis of the lattice Boltzmann method implementation on GPU, in *XL CILAMCE 2019 Ibero-Latin Congress on Computational Methods in Engineering (ABMEC), Natal, Brazil* (2019).

[37] C. Obrecht, F. Kuznik, B. Tourancheau, and J.-J. Roux, A new approach to the lattice Boltzmann method for graphics processing units, Comput. Math. Appl. **61**, 3628 (2011).

[38] N.-P. Tran, M. Lee, and S. Hong, Performance optimization of 3D lattice Boltzmann flow solver on a GPU, Sci. Program. **2017** (2017).

[39] P. R. Rinaldi, E. Dari, M. J. Vénere, and A. Clausse, A lattice-Boltzmann solver for 3D fluid simulation on GPU, Simul. Modell. Pract. Theory **25**, 163 (2012).

[40] P. R. Rinaldi, E. A. Dari, M. J. Vénere, and A. Clausse, Fluid simulation with lattice Boltzmann methods implemented on GPUs using CUDA, in *High-Performance Computing Symposium (HPC2009), San Diego, California, USA* (2009).

[41] J. Beny and J. Latt, Efficient LBM on GPUs for dense moving objects using immersed boundary condition, arXiv:1904.02108.

[42] J. Ames, D. F. Puleri, P. Balogh, J. Gounley, E. W. Draeger, and A. Randles, Multi-GPU immersed boundary method hemodynamics simulations, J. Comput. Sci. **44**, 101153 (2020).

[43] Q. Xiong, B. Li, J. Xu, X. Fang, X. Wang, L. Wang, X. He, and W. Ge, Efficient parallel implementation of the lattice Boltzmann method on large clusters of graphic processing units, Chin. Sci. Bull. **57**, 707 (2012).

[44] H. Zhu, X. Xu, G. Huang, Z. Qin, and B. Wen, An efficient graphics processing unit scheme for complex geometry simulations using the lattice Boltzmann method, IEEE Access **8**, 185158 (2020).

[45] J. Duchateau, F. Rousselle, N. Maquignon, G. Roussel, and C. Renaud, Accelerating physical simulations from a multicomponent lattice Boltzmann method on a single-node multi-GPU architecture, in *Proceedings of the 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)* (IEEE, Krakow, Poland, 2015), pp. 315–322.

[46] C. F. Janßen, D. Mierke, M. Überrück, S. Gralher, and T. Rung, Validation of the GPU-accelerated CFD solver ELBE for free surface flow problems in civil and environmental engineering, Computation **3**, 354 (2015).

[47] J. Habich, T. Zeiser, G. Hager, and G. Wellein, Performance analysis and optimization strategies for a D3Q19 lattice Boltzmann kernel on nVIDIA GPUs using CUDA, Adv. Eng. Software **42**, 266 (2011).

[48] E. Calore, D. Marchi, S. F. Schifano, and R. Tripiccione, Optimizing communications in multi-GPU lattice Boltzmann simulations, in *Proceedings of the 2015 International Conference on High Performance Computing & Simulation (HPCS)* (IEEE, Amsterdam, 2015), pp. 55–62.

[49] P.-Y. Hong, L.-M. Huang, L.-S. Lin, and C.-A. Lin, Scalable multi-relaxation-time lattice Boltzmann simulations on multi-GPU cluster, Comput. Fluids **110**, 1 (2015).

[50] W. Xian and A. Takayuki, Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster, Parallel Comput. **37**, 521 (2011).

[51] C. Obrecht, F. Kuznik, B. Tourancheau, and J.-J. Roux, Global memory access modelling for efficient implementation of the lattice Boltzmann method on graphics processing units, in *Proceedings of the International Conference on High Performance Computing for Computational Science* (Springer, Berkeley, California, 2010), pp. 151–161.

[52] F. Kuznik, C. Obrecht, G. Rusaouen, and J.-J. Roux, LBM based flow simulation using GPU computing processor, Comput. Math. Appl. **59**, 2380 (2010).

[53] C. Feichtinger, J. Habich, H. Köstler, G. Hager, U. Rüde, and G. Wellein, A flexible Patch-based lattice Boltzmann parallelization approach for heterogeneous GPU–CPU clusters, Parallel Comput. **37**, 536 (2011).

[54] E. Calore, A. Gabbana, J. Kraus, E. Pellegrini, S. F. Schifano, and R. Tripiccione, Massively parallel lattice–Boltzmann codes on large GPU clusters, Parallel Comput. **58**, 1 (2016).

[55] A. Horga, With lattice Boltzmann models using CUDA enabled GPGPUs, Master's thesis, University of Timisoara, Romania (2013).

[56] N. Onodera, Y. Idomura, S. Uesawa, S. Yamashita, and H. Yoshida, Locally mesh-refined lattice Boltzmann method for fuel debris air cooling analysis on GPU supercomputer, Mech. Eng. J. **7**, 19-00531 (2020).

[57] G. Falcucci, G. Amati, P. Fanelli, V. K. Krastev, G. Polverino, M. Porfiri, and S. Succi, Extreme flow simulations reveal skeletal adaptations of deep-sea sponges, Nature (London) **595**, 537 (2021).

[58] S. Zitz, A. Scagliarini, S. Maddu, A. A. Darhuber, and J. Harting, Lattice Boltzmann method for thin-liquid-film hydrodynamics, Phys. Rev. E **100**, 033313 (2019).

[59] M. Mohrhard, G. Thäter, J. Bludau, B. Horvat, and M. Krause, An auto-vecotorization friendly parallel lattice Boltzmann streaming scheme for direct addressing, Comput. Fluids **181**, 1 (2019).

[60] F. Gray and E. Boek, Enhancing computational precision for lattice Boltzmann schemes in porous media flows, Computation **4**, 11 (2016).

[61] W. Li, Y. Ma, X. Liu, and M. Desbrun, Efficient kinetic simulation of two-phase flows, ACM Trans. Graphics **41**, 114 (2022).

[62] M. Geier and M. Schönherr, Esoteric twist: An efficient in-place streaming algorithmus for the lattice Boltzmann method on massively parallel hardware, Computation **5**, 19 (2017).

[63] M. Wittmann, T. Zeiser, G. Hager, and G. Wellein, Comparison of different propagation steps for lattice Boltzmann methods, Comput. Math. Appl. **65**, 924 (2013).

[64] M. Wittmann, Ph.D. thesis, Friedrich-Alexander-Universitt Erlangen-Nürnberg, Erlangen, Germany (2016), https://nbn-resolving.org/urn:nbn:de:bvb:29-opus4-74586.

[65] F. Bonaccorso, A. Montessori, A. Tiribocchi, G. Amati, M. Bernaschi, M. Lauricella, and S. Succi, Lbsoft: A parallel open-source software for simulation of colloidal systems, Comput. Phys. Commun. **256**, 107455 (2020).

[66] P. Skordos, Initial and boundary conditions for the lattice Boltzmann method, Phys. Rev. E **48**, 4823 (1993).

[67] G. Wellein, P. Lammers, G. Hager, S. Donath, and T. Zeiser, Towards optimal performance for lattice Boltzmann applications on terascale computers, in *Parallel Computational Fluid Dynamics 2005* (Elsevier, Amsterdam, Netherlands, 2006), pp. 31–40.

[68] M. J. Krause, A. Kummerländer, S. J. Avis, H. Kusumaatmaja, D. Dapelo, F. Klemens, M. Gaedtke, N. Hafen, A. Mink, R. Trunk *et al.*, OpenLB—Open source lattice Boltzmann code, Comput. Math. Appl. **81**, 258 (2021).

[69] J. Latt and M. Krause, OpenLB release 0.3: Open source lattice Boltzmann code (2007).

[70] V. Heuveline and M. Krause, OpenLB: Towards an efficient parallel open source library for lattice Boltzmann fluid flow simulations, in *PARA'08 Workshop on State-of-the-Art in Scientific and Parallel Computing, May 13-16, 2008*, Lecture Notes in Computer Science (LNCS) Vols. No. 6126 and No. 6127 (Springer, Trondheim, Norway, 2011) published online 2011, https://para08.idi.ntnu.no/docs/submission_37.pdf.

[71] M. Krause, S. Avis, H. Kusumaatmaja, D. Dapelo, M. Gaedtke, N. Hafen, M. Haußmann, J. Jeppener-Haltenhoff, L. Kronberg, A. Kummerländer, J. Marquardt, T. Pertzel, S. Simonis, R. Trunk, M. Wu, and A. Zarth, OpenLB release 1.4: Open source lattice Boltzmann code (2020).

[72] J. Latt, O. Malaspinas, D. Kontaxakis, A. Parmigiani, D. Lagrava, F. Brogi, M. B. Belgacem, Y. Thorimbert, S. Leclaire, S. Li *et al.*, Palabos: Parallel lattice Boltzmann solver, Comput. Math. Appl. **81**, 334 (2021).

[73] T. Min, G. Weidong, P. Jingshan, and G. Meng, Performance analysis and optimization of PalaBos on petascale Sunway BlueLight MPP Supercomputer, Procedia Eng. **61**, 241 (2013).

[74] L. Mountrakis, E. Lorenz, O. Malaspinas, S. Alowayyed, B. Chopard, and A. G. Hoekstra, Parallel performance of an IB-LBM suspension simulation framework, J. Comput. Sci. **9**, 45 (2015).

[75] C. Kotsalos, J. Latt, and B. Chopard, Bridging the computational gap between mesoscopic and continuum modeling of red blood cells for fully resolved blood flow, J. Comput. Phys. **398**, 108905 (2019).

[76] C. Kotsalos, J. Latt, and B. Chopard, Palabos-npFEM: Software for the simulation of cellular blood flow (digital blood), J. Open Res. Software **9**, 16 (2021).

[77] G. Wellein, T. Zeiser, G. Hager, and S. Donath, On the single processor performance of simple lattice Boltzmann kernels, Comput. Fluids **35**, 910 (2006).

[78] A. Lintermann and W. Schröder, Lattice–Boltzmann simulations for complex geometries on high-performance computers, CEAS Aeronaut. J. **11**, 745 (2020).

[79] S. Schmieschek, L. Shamardin, S. Frijters, T. Krüger, U. D. Schiller, J. Harting, and P. V. Coveney, LB3D: A parallel implementation of the lattice-Boltzmann method for simulation of Interacting amphiphilic fluids, Comput. Phys. Commun. **217**, 149 (2017).

[80] IEEE Computer Society. Standards Committee and American National Standards Institute, IEEE standard for binary floating-point arithmetic, IEEE Std 754-2019 (Revision of IEEE 754-2008) 754 (1985).

[81] D. Goldberg, What every computer scientist should know about floating-point arithmetic, ACM Comput. Surv. **23**, 5 (1991).

[82] IS Committee and others, 754–2008 IEEE standard for floating-point arithmetic, IEEE Comput. Soc. Std. **2008** (2008).

[83] W. Kahan, IEEE standard 754 for binary floating-point arithmetic, Lect. Notes Status IEEE **754**, 11 (1996).

[84] T. Grützmacher and H. Anzt, A modular precision format for decoupling arithmetic format and storage format, in *European Conference on Parallel Processing* (Springer, Turin, Italy, 2018), pp. 434–443.

[85] H. Anzt, G. Flegar, T. Grützmacher, and E. S. Quintana-Ortí, Toward a modular precision ecosystem for high-performance computing, Int. J. High Perform. Comput. Appl. **33**, 1069 (2019).

[86] M. Krause, Fluid flow dimulation and optimisation with lattice Boltzmann methods on high performance computers: Application to the human respiratory system, Ph.D. thesis, Karlsruhe Institute of Technology (KIT), Universität Karlsruhe (TH), 2010, http://digbib.ubka.uni-karlsruhe.de/volltexte/1000019768.

[87] S. Succi, G. Amati, M. Bernaschi, G. Falcucci, M. Lauricella, and A. Montessori, Towards exascale lattice Boltzmann computing, Comput. Fluids **181**, 107 (2019).

[88] D. d'Humières, Multiple–relaxation–time lattice Boltzmann models in three dimensions, Proc. R. Soc. London, Ser. A **360**, 437 (2002).

[89] M. Klöwer, M. Razinger, J. J. Dominguez, P. D. Düben, and T. N. Palmer, Compressing atmospheric data into its real information content, Nat. Comput. Sci. **1**, 713 (2021).

[90] M. Klöwer, P. Düben, and T. Palmer, Number formats, error mitigation, and scope for 16-bit arithmetics in weather and climate modeling analyzed with a shallow water model, J. Adv. Model. Earth Syst. **12**, e2020MS002246 (2020).

[91] S. Hatfield, M. Chantry, P. Düben, and T. Palmer, Accelerating high-resolution weather models with deep-learning hardware, in *Proceedings of the Platform for Advanced Scientific Computing Conference* (ACM Press, Zurich, Switzerland, 2019), pp. 1–11.

[92] J. Langou, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, and J. Dongarra, Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems), in *SC'06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing* (IEEE, Tampa, Florida, USA, 2006), pp. 50–50.

[93] A. Haidar, H. Bayraktar, S. Tomov, J. Dongarra, and N. J. Higham, Mixed-precision iterative refinement using tensor cores on GPUs to accelerate solution of linear systems, Proc. R. Soc. A **476**, 20200110 (2020).

[94] X. He and L.-S. Luo, Lattice Boltzmann model for the incompressible Navier–Stokes equation, J. Stat. Phys. **88**, 927 (1997).

[95] T. Krüger, Unit conversion in LBM, in LBM Workshop. Dostupné z: http://lbmworkshop.com/wp-content/uploads/2011/08/2011-08-22_Edmonton_scaling.pdf (2011).

[96] G. I. Taylor and A. E. Green, Mechanism of the production of small eddies from large ones, Proc. R. Soc. London, Ser. A **158**, 499 (1937).

[97] T. v. Karman, Ueber den Mechanismus des Widerstandes, den ein bewegter Körper in einer Flüssigkeit erfährt, Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse, 509 (1911).

[98] A. Cattenone, S. Morganti, and F. Auricchio, Basis of the lattice Boltzmann method for additive manufacturing, Arch. Comput. Methods Eng. **27**, 1109 (2020).
[99] U. R. Alim, A. Entezari, and T. Moller, The lattice-Boltzmann method on optimal sampling lattices, IEEE Trans. Visualization Comput. Graphics **15**, 630 (2009).
[100] P. Neumann and T. Neckel, A dynamic mesh refinement technique for lattice Boltzmann simulations on octree-like grids, Comput. Mech. **51**, 237 (2013).
[101] U. Ghia, K. N. Ghia, and C. Shin, High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, J. Comput. Phys. **48**, 387 (1982).
[102] B.-N. Jiang, T. Lin, and L. A. Povinelli, Large-scale computation of incompressible viscous flow by least-squares finite element method, Comput. Methods Appl. Mech. Eng. **114**, 213 (1994).
[103] J.-Y. Yang, S.-C. Yang, Y.-N. Chen, and C.-A. Hsu, Implicit weighted ENO schemes for the three-dimensional incompressible Navier–Stokes equations, J. Comput. Phys. **146**, 464 (1998).
[104] D. Barthes-Biesel, Motion and deformation of elastic capsules and vesicles in flow, Annu. Rev. Fluid Mech. **48**, 25 (2016).
[105] A. Guckenberger, M. P. Schraml, P. G. Chen, M. Leonetti, and S. Gekle, On the bending algorithms for soft objects in flows, Comput. Phys. Commun. **207**, 1 (2016).
[106] A. Guckenberger and S. Gekle, Theory and algorithms to compute Helfrich bending forces: A review, J. Phys.: Condens. Matter **29**, 203001 (2017).
[107] S. Williams, A. Waterman, and D. Patterson, Roofline: An insightful visual performance model for floating-point programs and multicore architectures (Lawrence Berkeley National Lab., Berkeley, CA, 2009).
[108] J. Latt, C. Coreixas, J. Beny, and A. Parmigiani, Efficient supersonic flow simulations using lattice Boltzmann methods based on numerical equilibria, Philos. Trans. R. Soc. A **378**, 20190559 (2020).
[109] N. Frapolli, S. S. Chikatamarla, and I. V. Karlin, Entropic lattice Boltzmann model for gas dynamics: Theory, boundary conditions, and implementation, Phys. Rev. E **93**, 063302 (2016).
[110] M. Atif, M. Namburi, and S. Ansumali, Higher-order lattice Boltzmann model for thermohydrodynamics, Phys. Rev. E **98**, 053311 (2018).
[111] J. L. Gustafson and I. T. Yonemoto, Beating floating point at its own game: posit arithmetic, Supercomputing Front. Innovations **4**, 71 (2017).
[112] NVIDIA Corporation, Parallel Thread Execution ISA Version 7.2 (2021).
[113] J. L. Gustafson, posit arithmetic, Mathematica Notebook describing the posit number system **30** (2017).
[114] F. De Dinechin, L. Forget, J.-M. Muller, and Y. Uguen, posits: The good, the bad and the ugly, in *Proceedings of the Conference for Next Generation Arithmetic 2019, Singapore* (2019), pp. 1–10.
[115] C. Leong, SoftPosit library, accessed Sept, 24, 2019, https://gitlab.com/cerlane/SoftPosit.
[116] Z. Guo, C. Zheng, and B. Shi, Discrete lattice effects on the forcing term in the lattice Boltzmann method, Phys. Rev. E **65**, 046308 (2002).
[117] C. K. Batchelor and G. Batchelor, *An Introduction to Fluid Dynamics* (Cambridge University Press, Cambridge, UK, 2000).
[118] S. Izquierdo, P. Martínez-Lera, and N. Fueyo, Analysis of open boundary effects in unsteady lattice Boltzmann simulations, Comput. Math. Appl. **58**, 914 (2009).
[119] T. Krüger, Introduction to the immersed boundary method, in *LBM Workshop* (Edmonton, Canada, 2011).
[120] A. J. Ladd, Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation, J. Fluid Mech. **271**, 285 (1994).
[121] D. Barthes-Biesel, J. Walter, and A.-V. Salsac, *Flow-Induced Deformation of Artificial Capsules* (CRC Press, Boca Raton, 2010), pp. 35–70.
[122] A. R. Harwood and A. J. Revell, Parallelisation of an interactive lattice-Boltzmann method on an Android-powered mobile device, Adv. Eng. Software **104**, 38 (2017).
[123] P. Neumann and M. Zellner, Lattice Boltzmann flow simulation on Android devices for interactive mobile-based learning, in *European Conference on Parallel Processing* (Springer, Grenoble, France, 2016), pp. 3–15.
[124] NVIDIA Corporation, NVIDIA Turing GPU Architecture (2018), https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf.
[125] NVIDIA Corporation, NVIDIA A100 Tensor Core GPU Architecture (2020), https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf.
[126] R. Gonzales, NVIDIA CUDA force P2 state - performance analysis (off vs. on) (2021), https://babeltechreviews.com/nvidia-cuda-force-p2-state/.
[127] J. Harting, S. Frijters, M. Ramaioli, M. Robinson, D. E. Wolf, and S. Luding, Recent advances in the simulation of particle-laden flows, Eur. Phys. J.: Spec. Top. **223**, 2253 (2014).
[128] T. Krüger, S. Frijters, F. Günther, B. Kaoui, and J. Harting, Numerical simulations of complex fluid-fluid interface dynamics, Eur. Phys. J.: Spec. Top. **222**, 177 (2013).
[129] T. Krüger, B. Kaoui, and J. Harting, Interplay of inertia and deformability on rheological properties of a suspension of capsules, J. Fluid Mech. **751**, 725 (2014).
[130] N. Rivas, S. Frijters, I. Pagonabarraga, and J. Harting, Mesoscopic electrohydrodynamic simulations of binary colloidal suspensions, J. Chem. Phys. **148**, 144101 (2018).

8.4 Publication 4

Esoteric Pull and Esoteric Push: Two Simple In-Place Streaming Schemes for the Lattice Boltzmann Method on GPUs

by

Moritz Lehmann

Computation 10.6 (2022), p. 92
https://doi.org/10.3390/computation10060092

Esoteric Pull and Esoteric Push – two simple in-place streaming schemes for the lattice Boltzmann method on GPUs

Moritz Lehmann[1*]

June 2, 2022

*Correspondence:
moritz.lehmann@uni-bayreuth.de
[1]Biofluid Simulation and Modeling – Theoretische Physik VI, University of Bayreuth

Keywords: lattice Boltzmann method; GPU; in-place streaming; swap algorithm; Esoteric Twist; memory; memory bandwidth; Volume-of-Fluid; FluidX3D; OpenCL

Abstract

I present two novel thread-safe in-place streaming schemes for the lattice Boltzmann method (LBM) on graphics processing units (GPUs), termed Esoteric-Pull and Esoteric-Push, to make the LBM only require one copy of density distribution functions (DDFs) instead of two, greatly reducing memory demand. These build upon the idea of the existing Esoteric-Twist scheme, to stream half of the DDFs at the end of one stream-collide kernel and the remaining half at the beginning of the next, and offer the same beneficial properties over the AA-Pattern scheme – reduced memory bandwidth due to implicit bounce-back boundaries and the possibility to swap pointers between even and odd time steps. However the streaming directions now are chosen in a way that the algorithm can be implemented in about one tenth the amount of code, as two simple loops, compatible with all velocity sets and suitable for automatic code-generation. Performance of the new streaming schemes is even slightly increased over Esoteric-Twist due to better memory coalescence. Benchmarks across a large variety of GPUs and CPUs and show that for most dedicated GPUs, performance differs only insignificantly from the One-Step-Pull scheme, but for integrated GPUs and CPUs, performance is significantly improved. The two proposed algorithms greatly facilitate modifying existing codes to in-place streaming, even with extensions already in place, such as demonstrated for the Free Surface LBM implementation FluidX3D. Their simplicity, together with their ideal performance characteristics, should enable more widespread adoption of in-place streaming across LBM GPU codes.

1 Introduction

The lattice Boltzmann method (LBM) [1] is a type of direct numerical simulation (DNS) to model fluid flow in a physically accurate manner. Its explicit algorithmic structure makes it ideal for parallelization on graphics processing units (GPUs) [2–59]. The LBM works on a mesoscopic scale, representing quantities of fluid molecules by density distribution functions (DDFs) that are exchanged (streamed) between neighboring points on a Cartesian lattice. These DDFs are represented as floating-point numbers and the streaming consists of copying them to the memory locations associated with neighboring lattice points. So the LBM algorithm at its core is copying floating-point numbers in memory with little arithmetic computation in between, meaning its performance is bound by memory bandwidth [3–10, 13–22, 33–46, 59–65].

Since each lattice point holds the same number of DDFs and they need to be exchanged between one another in every time step, a way to avoid the data dependencies on parallel hardware is needed. The most straightforward, and also most common approach [12–44] is to have two copies A and B of the DDFs residing in memory: in even steps read from A and write to B and in odd steps vice versa. With two copies of the DDFs, the memory access can even be chosen in a way to have only partially misaligned reads and only coalesced writes (One-Step-Pull scheme), enabling peak memory efficiency on modern GPUs [13–23]. This solves the data dependencies, but comes at the cost of almost doubling memory demand. Unfortunately, memory capacity is the largest constraint on GPUs [58], limiting maximum possible lattice resolution. To eliminate the higher memory demand and at the same time resolve data dependencies, a class of thread-

safe in-place streaming algorithms have been developed. The first of these is termed the AA-Pattern [3], as it reads from A and at the same time writes to A again in a special manner that does not violate data dependencies. The algorithm however is asymmetric for even and odd time steps, so it has not been widely adopted. A later variant of AA-Pattern is Shift-and-Swap-Streaming [4]. Geier and Schönherr have recently found a more intricate solution termed Esoteric-Twist [2] that is symmetric for even and odd time steps, and moreover rewards with slightly reduced memory bandwidth and thus higher performance compared to AA-Pattern; however its implementation is very complicated, also hindering widespread adoption. Not too many works consider in-place streaming on GPU so far [2–12], and apart from the works introducing the methods [2–5, 66], only few have adopted in-place streaming in their codes [6–8, 11].

This work introduces two new thread-safe in-place streaming schemes – termed Esoteric-Pull and Esoteric-Push – well suitable for GPU implementation. They build upon the same idea as Esoteric-Twist and offer ideal performance characteristics, but at the same time significantly simplify the implementation and even allow for automatic code generation. The very simple and modular implementation is especially well suited to modify existing LBM implementations, even if various extensions such as Free Surface LBM are already in place.

2 Naive implementation - One-Step-Pull and One-Step-Push

The most common LBM implementation uses two copies of the DDFs in memory to resolve data dependencies in a parallel environment [12–44]. There is two variants, One-Step-Pull (figure 1, listing 1) and One-Step-Push (figure 2, listing 2). The pull variant is generally preferred on GPUs as the penalty for non-coalesced reads is smaller than for non-coalesced writes [13–23]. The coloring introduced in figures 1 and 2 illustrates how loading/storing patterns compare to the regular DDF sequence during collision in registers, in other words where exactly each DDF is loaded and stored in memory. This makes especially the later introduced, more sophisticated streaming patterns more comprehensible. In the One-Step-Pull scheme, DDFs are pulled in from neighbors (copy A of the DDFs), collided, and stored at the center node (copy B of the DDFs). In the One-Step-Push scheme, DDFs are loaded from the center node (copy A of the DDFs), collided, and then pushed out to neighbors (copy B of the DDFs). For both schemes, after every time step, the pointers to A and B are swapped.

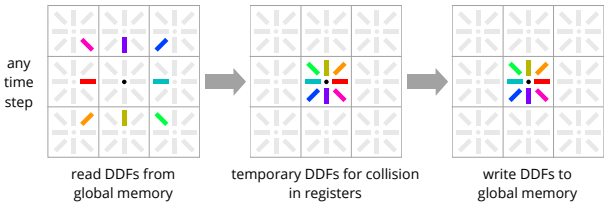


Figure 1: One-Step-Pull streaming scheme. Two copies of the DDFs are used to resolve data dependencies.

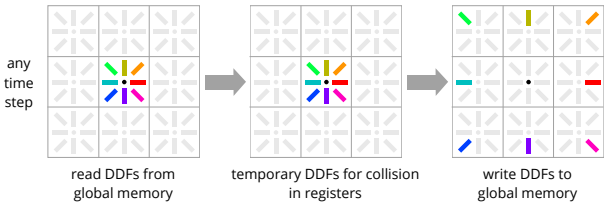


Figure 2: One-Step-Push streaming scheme. Two copies of the DDFs are used to resolve data dependencies.

3 State-of-the-art methods for in-place streaming on GPUs

The data dependency problem with in-place streaming on parallel hardware has been solved by two major approaches already, termed AA-Pattern and Esoteric-Twist. Both provide the great advantage of significantly reducing memory demand for the LBM; however both also pose various difficulties in GPU implementation, hindering widespread adoption.

3.1 AA-Pattern

When performing the LBM streaming step on parallel hardware, the issue arises that neighboring lattice points may be processed in parallel and the exact order of execution is random. One or more threads may not write an updated value to a memory address from which another concurrent thread is reading, because then either the old or the new value may be used by the reading thread. This error is known as a race condition. To perform the LBM streaming step in parallel with only a single buffer for the DDFs, one must write updated values only to the same memory addresses that one thread has previously read the values from. Then, no two threads access the same memory addresses. Bailey et al. [3] have found that this is possible, if for even time steps combining one streaming step, the collision and a second streaming

step, and for odd time steps only performing the collision step. This makes the processed DDFs always end up in the same locations as they were read from, resolving data dependencies on concurrent hardware with only one copy of the DDFs in memory. To make the DDFs actually stream through memory locations, in even time steps, after the collision the DDFs are stored at the neighbor nodes in opposite orientation, and in odd time steps, before the collision the DDFs are loaded from the center node in opposite orientation. The resulting algorithm reads the DDFs from the copy A of the DDFs and writes to the same copy A in-place, so it was termed AA-Pattern (figure 3, listing 3). It is a popular disbelief that the different even and odd time steps would require duplicate implementation of the stream_collide kernel as the pointers to the DDFs cannot be swapped in between time steps. The loading and storing of the DDFs before and after collision can be placed in functions, and when the LBM time step is passed as a parameter, these functions then switch between loading/storing the DDFs from/to neighbors or at the center point (see listing 3). The stream_collide kernel then contains calls to these two functions before and after collision and no duplicate implementation is required. Note that there is also two more modern variants of the AA-Pattern termed Shift-and-Swap-Streaming (SSS) [4] and Periodic-Shift (PS) [5], offering benefits in programming languages where pointer arithmetic is available.

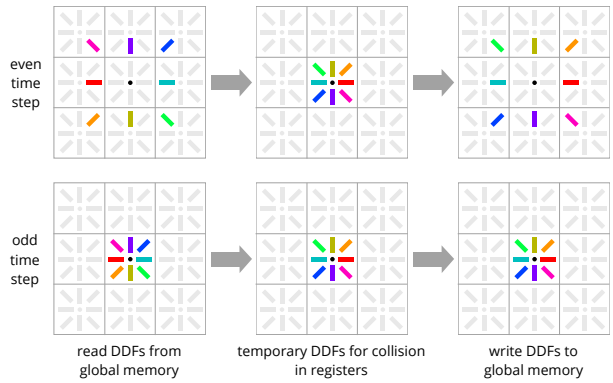


Figure 3: AA-Pattern in-place streaming scheme [3]. Even time steps: DDFs are pulled in from neighbors, collided, and then pushed out to the neighbors again, but stored in opposite orientation. Odd time steps: DDFs are loaded from the center node in opposite orientation, collided, and stored at the center node again in the same orientation as during collision. DDFs are always stored in the same memory locations where they are loaded from, so only one copy of the DDFs is required.

3.2 Esoteric-Twist

The idea of the Esoteric-Twist in-place streaming scheme (figure 4, listing 4) is to pull only DDFs for negative directions, do the collision, and then push only DDFs for positive directions (here: $x, y, z > 0$) [2]. To resolve data dependencies on concurrent hardware, in even steps, after the collision the DDFs are stored in opposite orientation, and in odd time steps, before the collision the DDFs are loaded in opposite orientation.

Esoteric-Twist resembles a criss-cross access pattern shifted north-east by half a node, accessing the DDFs at a total of 4 nodes (in the 2D case) or up to 8 nodes (in 3D) respectively. For certain implementations, this reduces the number of ghost nodes required, but it makes the implementation of the index calculation tedious as it requires manually writing the indices, that are different across velocity sets. On top, for some velocity sets like D3Q15, additional streaming directions must be computed beyond the streaming directions of the velocity set (see listing 4). This is an obstacle for implementing different velocity sets in a modular manner. With some LBM extensions such as Volume-of-Fluid, duplicate (inverse) implementation of the streaming is required, so Esoteric-Twist here becomes rather impractical.

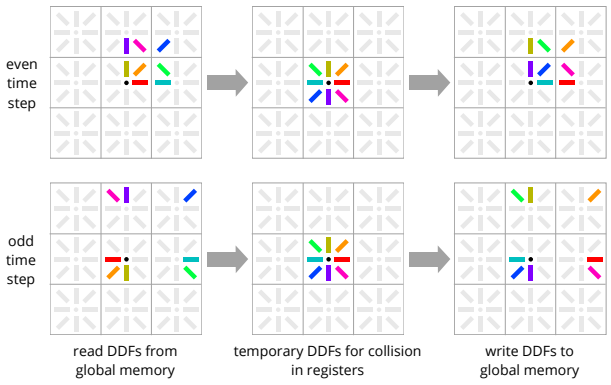


Figure 4: Esoteric-Twist in-place streaming scheme [2]. Even time steps: DDFs are loaded in a criss-cross pattern shifted north-east by half a node. After collision, DDFs are stored in the same pattern but in opposite orientation. Odd time steps: DDFs are loaded in opposite orientation in a shifted criss-cross pattern that covers only DDFs not touched in the even time step. After collision, DDFs are written back in the same pattern but with regular orientation again. DDFs are always stored in the same memory locations where they are loaded from, so only one copy of the DDFs is required.

4 New methods - Esoteric Pull and Esoteric Push

My two novel in-place streaming algorithms are based on the same idea as the Esoteric-Twist scheme, that only DDFs in negative directions (here: even i) are streamed before collision, and after collision only DDFs in positive directions (here: odd i) are pushed; the DDFs are aligned in a way such that even and odd i point in opposite directions. So half of the DDFs are streamed at the end of one stream_collide kernel and the other half is streamed at the beginning of the next.

The important observation I made is that the shifted criss-cross pattern of Esoteric-Twist is not essential for the swap algorithm to work. I waive shifting north the north-west to south-east DDFs by one node in 2D, and waive shifting other diagonal directions in 3D, respectively, abandoning the shifted criss-cross pattern. Instead, the regular streaming direction neighbors are used. This enables trivial index calculation in two four-line loops (unrolled by the compiler) for loading and storing in a way that works with all velocity sets out of the box. This also makes the implementation less redundant, much less prone to errors and it further enables automatic code-generation.

The Esoteric-Pull scheme (figure 5, listing 5) in 2D differs from Esoteric-Twist (figure 4) only in the positions of DDFs for the north-west to south-east directions, that are loaded/stored in their regular locations instead of shifted north by one node. In 3D, other diagonal directions are also not shifted and their regular streaming directions are used instead to determine the streaming neighbor nodes. This has not only the advantage of trivial index calculation, but also improves memory coalescence for the DDFs in these diagonal directions that would be otherwise shifted by one lattice point with Esoteric-Twist, leading to slightly higher performance.

The Esoteric-Push scheme (figure 6, listing 6) essentially is figure 5 flipped by 180 degrees (except for temporary DDFs in registers), highlighting two distinct symmetry flips that can be done independently of each other: a) switching streaming for positive/negative directions and b) switching even/odd time steps.

Both schemes yield bitwise identical simulations as with Esoteric-Twist. If pointer arithmetic is available, the pointers of DDFs in positive and negative directions can be swapped in between time steps such that memory addressing is the same for all time steps, in the very same manner as for Esoteric-Twist.

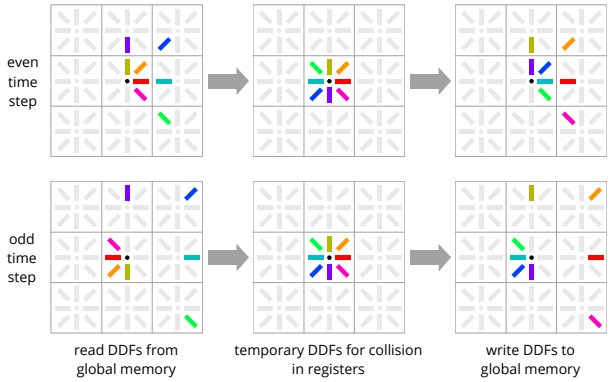


Figure 5: Esoteric-Pull in-place streaming scheme. Even time steps: DDFs in positive directions are loaded from the center node and DDFs from negative directions are pulled in from their regular streaming direction neighbors, are collided, and then DDFs in positive directions are pushed out to neighbors and stored in opposite orientation and DDFs in negative directions are stored at the center node in opposite orientation. Odd time steps: DDFs in positive directions are loaded from the center node in opposite orientation and DDFs from negative directions are pulled in from their regular streaming direction neighbors in opposite orientation, are collided, and then DDFs in positive directions are pushed out to neighbors and DDFs in negative directions are stored at the center node. DDFs are always stored in the same memory locations where they are loaded from, so only one copy of the DDFs is required.

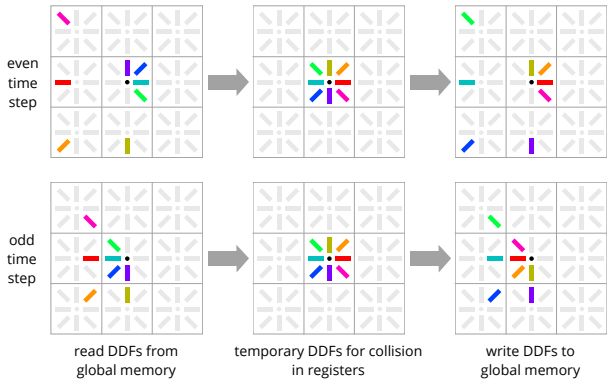


Figure 6: Esoteric-Push in-place streaming scheme. Figure 5 flipped by 180 degrees (except for temporary DDFs in registers).

4.1 Implicit bounce-back

In the very same manner as for the Esoteric-Twist scheme [2], both Esoteric-Pull and Esoteric-Push offer the benefits of the ingeniously emerging implicit bounce-back boundaries. Due to the way the DDFs are flipped in orientation for regular fluid nodes, and because boundary nodes are not processed at all (with a guard clause at the very beginning of the

stream_collide kernel), the DDFs of boundary nodes are not flipped in memory, so for neighboring fluid nodes it appears as if their DDFs are already correctly flipped such that bounce-back boundaries automatically apply.

There is three distinct benefits to this side-effect to the Esoteric streaming schemes: a) fluid nodes do not have to check the flags of their neighbors at all to apply bounce-back boundaries, reducing overall memory bandwidth and increasing performance slightly, b) memory access is more coalesced and c) the implementation is simplified.

4.2 Comparison with existing streaming schemes

When comparing the different streaming algorithms in table 1 for DdQq LBM, the advantages of in-place streaming become evident as both reduced storage and for the Esoteric algorithms also reduced bandwidth. In-place streaming reduces memory demand by $4\,q$ Bytes/node. The Esoteric algorithms further reduce memory bandwidth by $q-1$ Byte/node per time step, as neighbor flags do not have to be checked for implicit bounce-back boundaries. Based on their storage and performance properties, Esoteric-Pull/Push appear identical to Esoteric-Twist, apart from slightly improved memory coalescence for the DDFs in some of the diagonal directions. The main improvements of Esoteric-Pull/Push are located in the much more straightforward implementation that is compatible with all velocity sets. Instead of having to manually unroll the loops over the streaming directions and making sure all indices are typed correctly for each velocity set, the streaming can now be written in a generic way as two short loops that are unrolled by the compiler (compare listings 4 and 5). This also allows for automatic code-generation that many LBM implementations heavily rely on, and significantly improves code maintainability.

A recently proposed method to reduce storage and bandwidth by resorting to FP32/16-bit mixed precision [13] makes in-place streaming with the Esoteric schemes even more compelling, reducing memory storage from 169 to 55 Bytes/node – less than one third – and reducing bandwidth from 171 to 77 Bytes/node per time step for D3Q19. Less memory demand per node in turn enables much larger lattice resolution.

When comparing D3Q19 SRT performance of the different streaming algorithms on the Nvidia A100 40GB GPU with FP32 single-precision floating-point, One-Step-Pull serves as the baseline at 8816 MLUPs/s (100%). One-Step-Push is slightly slower at 8675 MLUPs/s (98%). The AA-Pattern runs with 8269 MLUPs/s (94%), Esoteric-Twist mitigates the efficiency losses with reduced bandwidth due to implicit bounce-back at 8483 MLUPs/s (96%), and Esoteric-Pull/Push offers even slightly higher performance at 8522 MLUPs/s (97%), due to better memory coalescence for the diagonals that are shifted by one lattice point for Esoteric-Twist. Looking at more performance benchmarks across different hardware (figure 7), the benefit of less memory bandwidth usage due to implicit bounce-back approximately cancels out with the drawback of more inefficient, partially misaligned writes due to the inherent symmetry of the memory access. In comparison with the One-Step-Pull streaming scheme [13], although memory efficiency is lower, performance changes only insignificantly on most dedicated GPUs, with some performance increase for FP32/FP16 mixed precision. On integrated GPUs and on CPUs however, there is a significant increase in performance due to more efficient use of on-chip cache with in-place memory-access. The benchmark case used is an empty box with default size of 256^3, with no extensions enabled except bounce-back boundaries, following [13]. For devices where not enough memory was available, the box size was reduced, and for the AMD Radeon VII the box size was increased to 464^3.

algorithm	storage	bandwidth
One-Step-Pull	$8\,q+4\,d+5$	$9\,q$
One-Step-Push	$8\,q+4\,d+5$	$9\,q$
AA-Pattern	$4\,q+4\,d+5$	$9\,q$
Esoteric-Twist	$4\,q+4\,d+5$	$8\,q+1$
Esoteric-Pull	$4\,q+4\,d+5$	$8\,q+1$
Esoteric-Push	$4\,q+4\,d+5$	$8\,q+1$
OSP + FP32/16-bit	$4\,q+4\,d+5$	$5\,q$
AA + FP32/16-bit	$2\,q+4\,d+5$	$5\,q$
ET/EP + FP32/16-bit	$2\,q+4\,d+5$	$4\,q+1$

Table 1: Comparing memory storage (Bytes/node) and bandwidth (Bytes/node per time step) requirements of the different GPU-compatible streaming algorithms for DdQq LBM with FP32 arithmetic precision and 8 available flag bits per node. With in-place streaming and implicit bounce-back of the Esoteric schemes, and with FP32/16-bit mixed precision as proposed in [13], memory demand and bandwidth are significantly reduced.

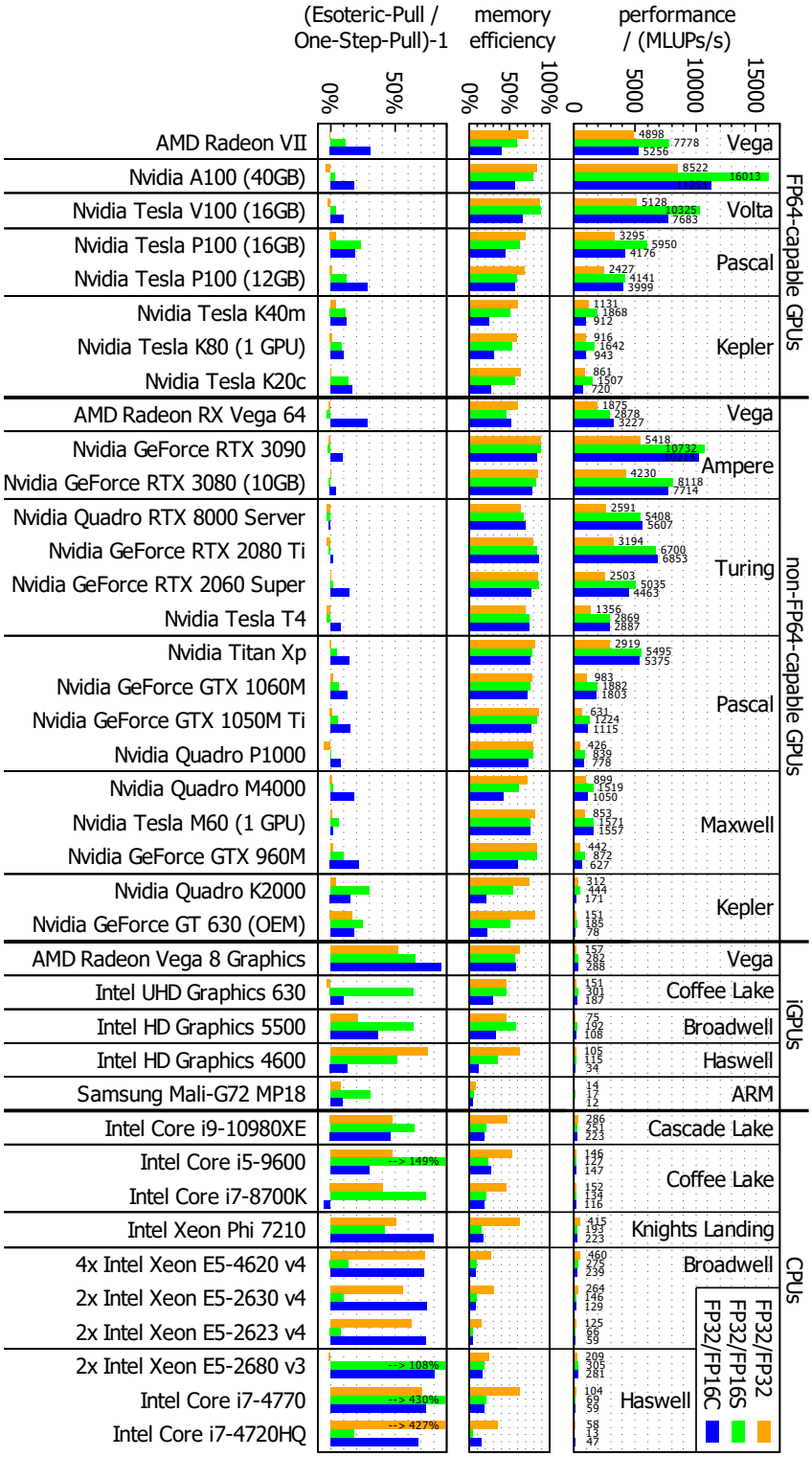


Figure 7: Performance of Esoteric-Pull with D3Q19 SRT on different hardware in the *FluidX3D* OpenCL implementation, in million lattice updates per second (MLUPs/s). Efficiency is calculated by dividing the measured MLUPs/s by the data sheet memory bandwidth times the number of bytes transferred per lattice point and time step (table 1). CPU benchmarks are on all cores. Performance comparison with the One-Step-Pull streaming scheme [13] shows only insignificant differences on most dedicated GPUs, but large gain on integrated GPUs and CPUs.

5 Esoteric Pull for Free Surface LBM on GPUs

To further underline the substantial perks of the added simplicity of Esoteric-Pull over Esoteric-Twist, here the modification of an existing Free Surface LBM (FSLBM) code from One-Step-Pull to Esoteric-Pull in-place streaming is briefly discussed on the example of *FluidX3D* [14].

Although the One-Step-Pull scheme makes an FSLBM implementation the simplest and most efficient, it can easily be modified to the Esoteric-Pull in-place streaming scheme. FSLBM on GPU requires three more kernels additionally to the stream_collide kernel. To distinguish between node types, 3 flag bits are required: fluid (001, F), interface (010, I), gas (100, G), interface→fluid (011, IF), interface→gas (110) and gas→interface (111, GI). The kernels are first introduced for the implementation with One-Step-Pull:

- stream_collide: Immediately return for G nodes. Stream in DDFs from neighbors, but for F and I nodes also load outgoing DDFs from the center node to compute the mass transfer for Volume-of-Fluid [67]. Apply excess mass for F or I nodes by summing it up from all neighboring F and I nodes. Compute the local surface curvature with PLIC [26] and reconstruct DDFs from neighboring G nodes. After collision, compare mass m and post-collision density ρ and along with neighboring flags, mark if the center node should remain I or change to IF or IG. Store post-collision DDFs at the local node.
- surface_1: Prevent neighbors of IF nodes to become/be G nodes; update flags of such neighbors to either I (from IF) or GI (from G).
- surface_2: For GI nodes, reconstruct and store DDFs based on the average density and velocity of all neighboring F, I or IF nodes. For IG nodes, turn all neighboring F or IF nodes to I.
- surface_3: Change IF nodes to F, IG nodes to G and GI nodes to I. Compute excess mass for each case separately as well as for F, I and G nodes, then divide the local excess mass by the number of neighboring F, I, IF and GI nodes, and store the excess mass on the local node.

After modifying the streaming scheme from One-Step-Pull to Esoteric-Pull, a fifth kernel must be added preceding the stream_collide kernel, because in the stream_collide kernel the outgoing DDFs cannot be loaded as neighboring nodes may overwrite them in memory within the same time step (race condition):

- surface_0: Immediately return for G nodes. Stream in DDFs like in figure 5 (incoming DDFs), but also load outgoing DDFs in opposite directions to compute the mass transfer for Volume-of-Fluid. For I nodes, compute the local surface curvature with PLIC and reconstruct DDFs for neighboring G nodes; store these reconstructed DDFs in the locations at the neighbors from which they will be streamed in in the following stream_collide kernel. Apply excess mass for F or I nodes by summing it up from all neighboring F and I nodes.
- stream_collide: Immediately return for G nodes. Stream in DDFs like in figure 5 and do collision. For I nodes, compare mass m and post-collision density ρ and along with neighboring flags, mark if the center node should remain I or change to IF or IG. Stream out the DDFs as in figure 5.
- surface_1, surface_2, surface_3: unchanged

In the surface_0 kernel, additional streaming in inverted directions is necessary to load outgoing DDFs and to store reconstructed DDFs for neighboring G nodes. Having to do manual index calculation here like with Esoteric-Twist would vastly elongate and over-complicate the code and reduce maintainability. With the Esoteric-Pull/Push variants, it is two simple loops of four lines of code each. The full FSLBM OpenCL C implementation with Esoteric-Pull is provided in the appendix in listing 7.

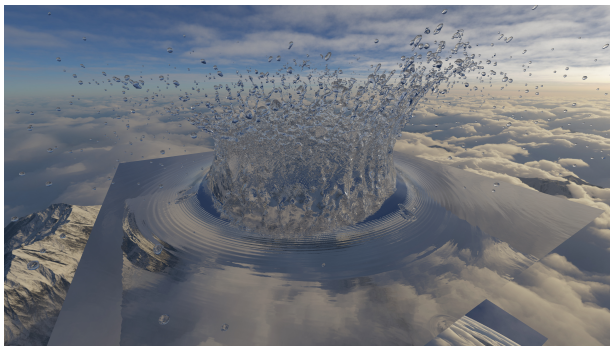


Figure 8: Esoteric-Pull in-place streaming together with FP16C memory compression [13] enables colossal $940\times940\times800$ lattice resolution on a single 48 GB GPU, such as demonstrated here with a raindrop impact simulation. This figure is included in the supplementary files as a video.

The change to the Esoteric-Pull in-place streaming algorithm reduces the memory demand for FSLBM from 181 to 105 Bytes/node, while only decreasing performance by approximately 20% due to the duplicate loading of incoming DDFs and having to store reconstructed DDFs for G nodes. When combined with FP32/16-bit mixed precision [13], the memory

demand is further reduced to 67 Bytes/node or about $\frac{1}{3}$ of vanilla FSLBM, comparable to or even less than the memory requirements of other (Navier-)Stokes solvers [68–71].

This in turn enables colossal lattice resolutions as illustrated with an example in figure 8: A 7 mm (188 lattice point) diameter raindrop, 1.5 ms (7700 time steps) after impacting a deep pool at 9.55 $\frac{\text{m}}{\text{s}}$ [24] and 20° inclination, simulated with the *FluidX3D* implementation with FP32/FP16C mixed precision [13]. Lattice resolution is $940 \times 940 \times 800$ or 707 million lattice points. The simulation was conducted on a Nvidia Quadro RTX 8000 GPU with 48 GB video memory and took 27 minutes, including rendering of the image. The code for this setup is provided in the appendix in listing 8.

6 Conclusions

In-place streaming is essential for any LBM GPU implementation as it significantly reduces memory demand and increases maximum lattice resolution. However, existing thread-safe solutions for GPUs such as AA-Pattern and Esoteric-Twist never gained widespread adoption due to difficult implementation. The new Esoteric-Pull and Esoteric-Push schemes presented in this work should change that. They build upon the same idea as the Esoteric-Twist scheme – streaming half of the DDFs at the end of one stream_collide kernel and the other half at the beginning of the next – but greatly simplify the implementation through trivial index calculation, even allowing for automatic code generation. For existing GPU implementations of the common One-Step-Pull scheme, the switch to Esoteric-Pull requires only moderate modifications to the code, even if several extensions are already implemented, as demonstrated with Free Surface LBM. In contrast, the implementation of Esoteric-Twist would be much more difficult and error-prone here, as the index calculation has to be implemented twice in different variations for regular LBM streaming and for FSLBM mass exchange.

The Esoteric-Pull and Esoteric-Push schemes share the same performance advantage as Esoteric-Twist over AA-Pattern – slightly reduced bandwidth due to implicit bounce-back. Compared to Esoteric-Twist, memory coalescence is even slightly improved on the otherwise shifted diagonal directions. This makes the Esoteric-Pull/Push schemes with only one DDF buffer provide GPU performance on par with the One-Step-Pull scheme with double DDF buffers, despite requiring less efficient misaligned writes. On integrated GPUs and CPUs, performance is even significantly increased.

Acknowledgements

I acknowledge support through the computational resources provided by BZHPC, LRZ and JSC. I acknowledge the NVIDIA Corporation for donating a Titan Xp GPU and an A100 40GB GPU for my research. I acknowledge Stephan Gekle for motivating me to write this paper.

Funding

This study was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project Number 391977956 - SFB 1357.

7 References

[1] Timm Krüger et al. “The lattice Boltzmann method”. In: *Springer International Publishing* 10 (2017), pp. 978–3.

[2] Martin Geier and Martin Schönherr. “Esoteric twist: an efficient in-place streaming algorithmus for the lattice Boltzmann method on massively parallel hardware”. In: *Computation* 5.2 (2017), p. 19.

[3] Peter Bailey et al. “Accelerating lattice Boltzmann fluid flow simulations using graphics processors”. In: *2009 international conference on parallel processing*. IEEE. 2009, pp. 550–557.

[4] M. Mohrhard et al. “An Auto-Vecotorization Friendly Parallel Lattice Boltzmann Streaming Scheme for Direct Addressing”. In: *Computers & Fluids* 181 (2019), pp. 1–7. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2019.01.001.

[5] Adrian Kummerländer et al. *Implicit Propagation of Directly Addressed Grids in Lattice Boltzmann Methods*. 2021.

[6] Martin Schreiber et al. “Free-surface lattice-Boltzmann simulation on many-core architectures”. In: *Procedia Computer Science* 4 (2011), pp. 984–993.

[7] Christoph Riesinger et al. “A holistic scalable implementation approach of the lattice Boltzmann method for CPU/GPU heterogeneous clusters”. In: *Computation* 5.4 (2017), p. 48.

[8] Eirik O Aksnes and Anne C Elster. “Porous rock simulations and lattice Boltzmann on GPUs”. In: *Parallel Computing: From Multicores and GPU's to Petascale*. Amsterdam, Netherlands: IOS Press, 2010, pp. 536–545.

[9] Markus Holzer, Martin Bauer, and Ulrich Rüde. "Highly Efficient Lattice-Boltzmann Multiphase Simulations of Immiscible Fluids at High-Density Ratios on CPUs and GPUs through Code Generation". In: *arXiv preprint arXiv:2012.06144* (2020).

[10] Julien Duchateau et al. "Accelerating physical simulations from a multicomponent Lattice Boltzmann method on a single-node multi-GPU architecture". In: *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*. IEEE. 2015, pp. 315–322.

[11] Wei Li et al. "Efficient Kinetic Simulation of Two-Phase Flows". In: *ACM Transactions on Graphics* 41.4 (2022), p. 114.

[12] Stuart DC Walsh et al. "Accelerating geoscience and engineering system simulations on graphics hardware". In: *Computers & Geosciences* 35.12 (2009), pp. 2353–2364.

[13] Moritz Lehmann et al. "On the accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit and novel 16-bit number formats". In: *arXiv preprint arXiv:2112.08926* (2021).

[14] Moritz Lehmann. *High Performance Free Surface LBM on GPUs*. 2019. URL: https://epub.uni-bayreuth.de/5400/.

[15] Michal Takáč and Ivo Petráš. "Cross-Platform GPU-Based Implementation of Lattice Boltzmann Method Solver Using ArrayFire Library". In: *Mathematics* 9.15 (2021), p. 1793.

[16] Mark J Mawson and Alistair J Revell. "Memory transfer optimization for a lattice Boltzmann solver on Kepler architecture nVidia GPUs". In: *Computer Physics Communications* 185.10 (2014), pp. 2566–2574.

[17] Nicolas Delbosc et al. "Optimized implementation of the Lattice Boltzmann Method on a graphics processing unit towards real-time fluid simulation". In: *Computers & Mathematics with Applications* 67.2 (2014), pp. 462–475.

[18] Nhat-Phuong Tran, Myungho Lee, and Sugwon Hong. "Performance optimization of 3D lattice Boltzmann flow solver on a GPU". In: *Scientific Programming* 2017 (2017).

[19] Christian Obrecht et al. "Multi-GPU implementation of the lattice Boltzmann method". In: *Computers & Mathematics with Applications* 65.2 (2013), pp. 252–261.

[20] Christian Obrecht et al. "A new approach to the lattice Boltzmann method for graphics processing units". In: *Computers & Mathematics with Applications* 61.12 (2011), pp. 3628–3638.

[21] Christian Feichtinger et al. "A flexible Patch-based lattice Boltzmann parallelization approach for heterogeneous GPU–CPU clusters". In: *Parallel Computing* 37.9 (2011), pp. 536–549.

[22] Enrico Calore et al. "Massively parallel lattice–Boltzmann codes on large GPU clusters". In: *Parallel Computing* 58 (2016), pp. 1–24.

[23] Christian Obrecht et al. "Global memory access modelling for efficient implementation of the lattice Boltzmann method on graphics processing units". In: *International Conference on High Performance Computing for Computational Science*. Springer. 2010, pp. 151–161.

[24] Moritz Lehmann et al. "Ejection of marine microplastics by raindrops: a computational and experimental study". In: *Microplastics and Nanoplastics* 1.18 (2021), pp. 1–19.

[25] Hannes Laermanns et al. "Tracing the horizontal transport of microplastics on rough surfaces". In: *Microplastics and Nanoplastics* 1.11 (2021), pp. 1–12.

[26] Moritz Lehmann and Stephan Gekle. "Analytic Solution to the Piecewise Linear Interface Construction Problem and Its Application in Curvature Calculation for Volume-of-Fluid Simulation Codes". In: *Computation* 10.2 (2022), p. 21.

[27] Fabian Häusl. *MPI-based multi-GPU extension of the Lattice Boltzmann Method*. 2019. URL: https://epub.uni-bayreuth.de/5689/.

[28] Fabian Häusl. *Soft Objects in Newtonian and Non-Newtonian Fluids: a Computational Study of Bubbles and Capsules in Flow*. 2022. URL: https://epub.uni-bayreuth.de/5960/.

[29] Hans-Jörg Limbach et al. "ESPResSo – an extensible simulation package for research on soft matter systems". In: *Computer Physics Communications* 174.9 (2006), pp. 704–727.

[30] Institute for Computational Physics, Universität Stuttgart. *ESPResSo User's Guide*. http://espressomd.org/wordpress/wp-content/uploads/2016/07/ug_07_2016.pdf. Accessed: 2018-06-15. 2016.

[31] Pei-Yao Hong et al. "Scalable multi-relaxation-time lattice Boltzmann simulations on multi-GPU cluster". In: *Computers & Fluids* 110 (2015), pp. 1–8.

[32] Wang Xian and Aoki Takayuki. "Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster". In: *Parallel Computing* 37.9 (2011), pp. 521–535.

[33] Minh Quan Ho et al. "Improving 3D Lattice Boltzmann Method stencil with asynchronous transfers on many-core processors". In: *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*. IEEE. 2017, pp. 1–9.

[34] Johannes Habich et al. "Performance engineering for the lattice Boltzmann method on GPGPUs: Architectural requirements and performance results". In: *Computers & Fluids* 80 (2013), pp. 276–282.

[35] Jonas Tölke and Manfred Krafczyk. "TeraFLOP computing on a desktop PC with GPUs for 3D CFD". In: *International Journal of Computational Fluid Dynamics* 22.7 (2008), pp. 443–456.

[36] Gregory Herschlag et al. "GPU data access on complex geometries for D3Q19 lattice Boltzmann method". In: *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2018, pp. 825–834.

[37] Waine B de Oliveira Jr, Alan Lugarini, and Admilson T Franco. "Performance analysis of the lattice Boltzmann method implementation on GPU". In: (2019).

[38] Pablo R Rinaldi et al. "A Lattice-Boltzmann solver for 3D fluid simulation on GPU". In: *Simulation Modelling Practice and Theory* 25 (2012), pp. 163–171.

[39] Pablo R Rinaldi et al. "Fluid Simulation with Lattice Boltzmann Methods Implemented on GPUs Using CUDA". In: 2009.

[40] Jeff Ames et al. "Multi-GPU immersed boundary method hemodynamics simulations". In: *Journal of Computational Science* 44 (2020), p. 101153.

[41] QinGang Xiong et al. "Efficient parallel implementation of the lattice Boltzmann method on large clusters of graphic processing units". In: *Chinese Science Bulletin* 57.7 (2012), pp. 707–715.

[42] Hongyin Zhu et al. "An Efficient Graphics Processing Unit Scheme for Complex Geometry Simulations Using the Lattice Boltzmann Method". In: *IEEE Access* 8 (2020), pp. 185158–185168.

[43] Frédéric Kuznik et al. "LBM based flow simulation using GPU computing processor". In: *Computers & Mathematics with Applications* 59.7 (2010), pp. 2380–2392.

[44] Adrian Horga. "With lattice Boltzmann models using CUDA enabled GPGPUs". In: *Master Thesis* (2013).

[45] Markus Geveler et al. "Lattice-Boltzmann simulation of the shallow-water equations with fluid-structure interaction on multi-and many-core processors". In: *Facing the multicore-challenge*. Wiesbaden, Germany: Springer, 2010, pp. 92–104.

[46] Joel Beny and Jonas Latt. "Efficient LBM on GPUs for dense moving objects using immersed boundary condition". In: *arXiv preprint arXiv:1904.02108* (2019).

[47] Predrag M Tekic, Jelena B Radjenovic, and Milos Rackovic. "Implementation of the Lattice Boltzmann method on heterogeneous hardware and platforms using OpenCL". In: *Advances in Electrical and Computer Engineering* 12.1 (2012), pp. 51–56.

[48] Joël Bény, Christos Kotsalos, and Jonas Latt. "Toward full GPU implementation of fluid-structure interaction". In: *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE. 2019, pp. 16–22.

[49] G Boroni, J Dottori, and P Rinaldi. "FULL GPU implementation of lattice-Boltzmann methods with immersed boundary conditions for fast fluid simulations". In: *The International Journal of Multiphysics* 11.1 (2017), pp. 1–14.

[50] Michael Griebel, Marc Alexander Schweitzer, et al. *Meshfree methods for partial differential equations II*. Cham, Switzerland: Springer, 2005.

[51] Stefan Zitz, Andrea Scagliarini, and Jens Harting. "Lattice Boltzmann simulations of stochastic thin film dewetting". In: *Physical Review E* 104.3 (2021), p. 034801.

[52] Christian F Janßen et al. "Validation of the GPU-accelerated CFD solver ELBE for free surface flow problems in civil and environmental engineering". In: *Computation* 3.3 (2015), pp. 354–385.

[53] Johannes Habich et al. "Performance analysis and optimization strategies for a D3Q19 lattice Boltzmann kernel on nVIDIA GPUs using CUDA". In: *Advances in Engineering Software* 42.5 (2011), pp. 266–272.

[54] Enrico Calore et al. “Optimizing communications in multi-GPU Lattice Boltzmann simulations”. In: *2015 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE. 2015, pp. 55–62.

[55] Naoyuki Onodera et al. “Locally mesh-refined lattice Boltzmann method for fuel debris air cooling analysis on GPU supercomputer”. In: *Mechanical Engineering Journal* 7.3 (2020), pp. 19–00531.

[56] Giacomo Falcucci et al. “Extreme flow simulations reveal skeletal adaptations of deep-sea sponges”. In: *Nature* 595.7868 (2021), pp. 537–541.

[57] Stefan Zitz et al. “Lattice Boltzmann method for thin-liquid-film hydrodynamics”. In: *Physical Review E* 100.3 (2019), p. 033313.

[58] Cao Wei et al. “An improved LBM approach for heterogeneous GPU-CPU clusters”. In: *2011 4th International Conference on Biomedical Engineering and Informatics (BMEI)*. Vol. 4. IEEE. 2011, pp. 2095–2098.

[59] Farrel Gray and Edo Boek. “Enhancing computational precision for lattice Boltzmann schemes in porous media flows”. In: *Computation* 4.1 (2016), p. 11.

[60] G Wellein et al. “Towards optimal performance for lattice Boltzmann applications on terascale computers”. In: *Parallel Computational Fluid Dynamics 2005*. Amsterdam, Netherlands: Elsevier, 2006, pp. 31–40.

[61] Markus Wittmann et al. “Comparison of different propagation steps for lattice Boltzmann methods”. In: *Computers & Mathematics with Applications* 65.6 (2013), pp. 924–935.

[62] Markus Wittmann. “Hardware-effiziente, hochparallele Implementierungen von Lattice-Boltzmann-Verfahren für komplexe Geometrien”. In: (2016).

[63] M.J. Krause. “Fluid Flow Simulation and Optimisation with Lattice Boltzmann Methods on High Performance Computers: Application to the Human Respiratory System”. PhD thesis. Karlsruhe Institute of Technology (KIT), Universität Karlsruhe (TH), 2010. URL: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000019768.

[64] Sauro Succi et al. “Towards exascale lattice Boltzmann computing”. In: *Computers & Fluids* 181 (2019), pp. 107–115.

[65] Dominique d’Humières. “Multiple–relaxation–time lattice Boltzmann models in three dimensions”. In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 360.1792 (2002), pp. 437–451.

[66] Jonas Latt. “Technical report: How to implement your DdQq dynamics with only q variables per node (instead of 2q)”. In: *Tufts University* (2007), pp. 1–8.

[67] Simon Bogner, Ulrich Rüde, and Jens Harting. “Curvature estimation from a volume-of-fluid indicator function for the simulation of surface tension and wetting with a free-surface lattice Boltzmann method”. In: *Physical Review E* 93.4 (2016), p. 043302.

[68] Keenan Crane, Ignacio Llamas, and Sarah Tariq. “Real-time simulation and rendering of 3d fluids”. In: *GPU gems* 3.1 (2007).

[69] Salvadore Gerace. “A model integrated meshless solver (MIMS) for fluid flow and heat transfer”. In: (2010).

[70] C Eric Lynch. *Advanced CFD methods for wind turbine analysis*. Georgia Institute of Technology, 2011.

[71] Andrea Keßler. “Matrix-free voxel-based finite element method for materials with heterogeneous microstructures”. In: (2019).

8 Appendix

8.1 OpenCL C implementation of the different streaming schemes

8.1.1 One-Step-Pull

```
)+R(void load_f(const uint n, float* fhn, const global fpxx* fA, const uint* j, const uchar* flagsj) {
  fhn[0] = load(fA, index_f(n, 0u)); // One-Step-Pull
  for(uint i=1u; i<def_velocity_set; i+=2u) {
    fhn[i  ] = load(fA, flagsj[i+1u]&TYPE_W ? index_f(n, i+1u) : index_f(j[i+1u], i   )); // boundary : regular
    fhn[i+1] = load(fA, flagsj[i   ]&TYPE_W ? index_f(n, i   ) : index_f(j[i   ], i+1u));
  }
}
)+R(void store_f(const uint n, const float* fhn, global fpxx* fB, const uint* j, const ulong t, const uchar* flagsj) {
  for(uint i=0u; i<def_velocity_set; i++) store(fB, index_f(n, i), fhn[i]); // One-Step-Pull
}
```

Listing 1: One-Step-Pull implementation in OpenCL C.

8.1.2 One-Step-Push

```
)+R(void load_f(const uint n, float* fhn, const global fpxx* fA, const uint* j, const uchar* flagsj) {
  for(uint i=0u; i<def_velocity_set; i++) fhn[i] = load(fA, index_f(n, i)); // One-Step-Push
}
)+R(void store_f(const uint n, const float* fhn, global fpxx* fB, const uint* j, const uchar* flagsj) {
  store(fB, index_f(n, 0u), fhn[0]); // One-Step-Push
  for(uint i=1u; i<def_velocity_set; i+=2u) {
    store(fB, flagsj[i   ]&TYPE_W ? index_f(n, i+1u) : index_f(j[i   ], i   ), fhn[i   ]); // boundary : regular
    store(fB, flagsj[i+1u]&TYPE_W ? index_f(n, i   ) : index_f(j[i+1u], i+1u), fhn[i+1u]);
  }
}
```

Listing 2: One-Step-Push implementation in OpenCL C.

8.1.3 AA-Pattern

```
)+R(void load_f(const uint n, float* fhn, const global fpxx* fi, const uint* j, const ulong t, const uchar* flagsj) {
  fhn[0] = load(fi, index_f(n, 0u)); // AA-Pattern
  if(t%2ul) {
    for(uint i=1u; i<def_velocity_set; i+=2u) { // pull
      fhn[i  ] = load(fi, flagsj[i+1u]&TYPE_W ? index_f(n, i+1u) : index_f(j[i+1u], i   )); // boundary : regular
      fhn[i+1] = load(fi, flagsj[i   ]&TYPE_W ? index_f(n, i   ) : index_f(j[i   ], i+1u));
    }
  } else {
    for(uint i=1u; i<def_velocity_set; i+=2u) { // load local (inverse)
      fhn[i  ] = load(fi, index_f(n, i+1u));
      fhn[i+1] = load(fi, index_f(n, i   ));
    }
  }
}
)+R(void store_f(const uint n, const float* fhn, global fpxx* fi, const uint* j, const ulong t, const uchar* flagsj) {
  store(fi, index_f(n, 0u), fhn[0]); // AA-Pattern
  if(t%2ul) {
    for(uint i=1u; i<def_velocity_set; i+=2u) { // push
      store(fi, flagsj[i   ]&TYPE_W ? index_f(n, i   ) : index_f(j[i   ], i+1u), fhn[i   ]); // boundary : regular
      store(fi, flagsj[i+1u]&TYPE_W ? index_f(n, i+1u) : index_f(j[i+1u], i   ), fhn[i+1u]);
    }
  } else {
    for(uint i=1u; i<def_velocity_set; i++) { // store local
      store(fi, index_f(n, i), fhn[i]);
    }
  }
}
```

Listing 3: AA-Pattern implementation in OpenCL C.

8.1.4 Esoteric-Twist

```
)+R(void load_f(const uint n, float* fhn, const global fpxx* fi, const uint* j, const ulong t) {
  fhn[0] = load(fi, index_f(n, 0u)); // Esoteric-Twist
)+"#if defined(D2Q9)"+R(
  fhn[1] = load(fi, index_f(n   , t%2ul ? 1 : 2));
  fhn[2] = load(fi, index_f(j[1], t%2ul ? 2 : 1));
  fhn[3] = load(fi, index_f(n   , t%2ul ? 3 : 4));
  fhn[4] = load(fi, index_f(j[3], t%2ul ? 4 : 3));
  fhn[5] = load(fi, index_f(n   , t%2ul ? 5 : 6));
  fhn[6] = load(fi, index_f(j[5], t%2ul ? 6 : 5));
  fhn[7] = load(fi, index_f(j[3], t%2ul ? 7 : 8)); // attention
  fhn[8] = load(fi, index_f(j[1], t%2ul ? 8 : 7)); // attention
)+"#elif defined(D3Q15)"+R(
  uint x0, xp, xm, y0, yp, ym, z0, zp, zm;
  calculate_indices(n, &x0, &xp, &xm, &y0, &yp, &ym, &z0, &zp, &zm);
  fhn[ 1] = load(fi, index_f(n       , t%2ul ?  1 :  2));
  fhn[ 2] = load(fi, index_f(j[ 1]   , t%2ul ?  2 :  1));
  fhn[ 3] = load(fi, index_f(n       , t%2ul ?  3 :  4));
  fhn[ 4] = load(fi, index_f(j[ 3]   , t%2ul ?  4 :  3));
  fhn[ 5] = load(fi, index_f(n       , t%2ul ?  5 :  6));
  fhn[ 6] = load(fi, index_f(j[ 5]   , t%2ul ?  6 :  5));
  fhn[ 7] = load(fi, index_f(n       , t%2ul ?  7 :  8));
```

```
    fhn[ 8] = load(fi, index_f(j[ 7]   , t%2ul ?  8 :  7));
    fhn[ 9] = load(fi, index_f(j[ 5]   , t%2ul ?  9 : 10)); // attention
    fhn[10] = load(fi, index_f(xp+yp+z0, t%2ul ? 10 :  9)); // attention, additional streaming direction required beyond velocity set
    fhn[11] = load(fi, index_f(j[ 3]   , t%2ul ? 11 : 12)); // attention
    fhn[12] = load(fi, index_f(xp+y0+zp, t%2ul ? 12 : 11)); // attention, additional streaming direction required beyond velocity set
    fhn[13] = load(fi, index_f(j[ 1]   , t%2ul ? 13 : 14)); // attention
    fhn[14] = load(fi, index_f(x0+yp+zp, t%2ul ? 14 : 13)); // attention, additional streaming direction required beyond velocity set
  )+"#elif defined(D3Q19)"+R(
    fhn[ 1] = load(fi, index_f(n    , t%2ul ?  1 :  2));
    fhn[ 2] = load(fi, index_f(j[ 1], t%2ul ?  2 :  1));
    fhn[ 3] = load(fi, index_f(n    , t%2ul ?  3 :  4));
    fhn[ 4] = load(fi, index_f(j[ 3], t%2ul ?  4 :  3));
    fhn[ 5] = load(fi, index_f(n    , t%2ul ?  5 :  6));
    fhn[ 6] = load(fi, index_f(j[ 5], t%2ul ?  6 :  5));
    fhn[ 7] = load(fi, index_f(n    , t%2ul ?  7 :  8));
    fhn[ 8] = load(fi, index_f(j[ 7], t%2ul ?  8 :  7));
    fhn[ 9] = load(fi, index_f(n    , t%2ul ?  9 : 10));
    fhn[10] = load(fi, index_f(j[ 9], t%2ul ? 10 :  9));
    fhn[11] = load(fi, index_f(n    , t%2ul ? 11 : 12));
    fhn[12] = load(fi, index_f(j[11], t%2ul ? 12 : 11));
    fhn[13] = load(fi, index_f(j[ 3], t%2ul ? 13 : 14)); // attention
    fhn[14] = load(fi, index_f(j[ 1], t%2ul ? 14 : 13)); // attention
    fhn[15] = load(fi, index_f(j[ 5], t%2ul ? 15 : 16)); // attention
    fhn[16] = load(fi, index_f(j[ 1], t%2ul ? 16 : 15)); // attention
    fhn[17] = load(fi, index_f(j[ 5], t%2ul ? 17 : 18)); // attention
    fhn[18] = load(fi, index_f(j[ 3], t%2ul ? 18 : 17)); // attention
  )+"#elif defined(D3Q27)"+R(
    fhn[ 1] = load(fi, index_f(n    , t%2ul ?  1 :  2));
    fhn[ 2] = load(fi, index_f(j[ 1], t%2ul ?  2 :  1));
    fhn[ 3] = load(fi, index_f(n    , t%2ul ?  3 :  4));
    fhn[ 4] = load(fi, index_f(j[ 3], t%2ul ?  4 :  3));
    fhn[ 5] = load(fi, index_f(n    , t%2ul ?  5 :  6));
    fhn[ 6] = load(fi, index_f(j[ 5], t%2ul ?  6 :  5));
    fhn[ 7] = load(fi, index_f(n    , t%2ul ?  7 :  8));
    fhn[ 8] = load(fi, index_f(j[ 7], t%2ul ?  8 :  7));
    fhn[ 9] = load(fi, index_f(n    , t%2ul ?  9 : 10));
    fhn[10] = load(fi, index_f(j[ 9], t%2ul ? 10 :  9));
    fhn[11] = load(fi, index_f(n    , t%2ul ? 11 : 12));
    fhn[12] = load(fi, index_f(j[11], t%2ul ? 12 : 11));
    fhn[13] = load(fi, index_f(j[ 3], t%2ul ? 13 : 14)); // attention
    fhn[14] = load(fi, index_f(j[ 1], t%2ul ? 14 : 13)); // attention
    fhn[15] = load(fi, index_f(j[ 5], t%2ul ? 15 : 16)); // attention
    fhn[16] = load(fi, index_f(j[ 1], t%2ul ? 16 : 15)); // attention
    fhn[17] = load(fi, index_f(j[ 5], t%2ul ? 17 : 18)); // attention
    fhn[18] = load(fi, index_f(j[ 3], t%2ul ? 18 : 17)); // attention
    fhn[19] = load(fi, index_f(n    , t%2ul ? 19 : 20));
    fhn[20] = load(fi, index_f(j[19], t%2ul ? 20 : 19));
    fhn[21] = load(fi, index_f(j[ 5], t%2ul ? 21 : 22)); // attention
    fhn[22] = load(fi, index_f(j[ 7], t%2ul ? 22 : 21)); // attention
    fhn[23] = load(fi, index_f(j[ 3], t%2ul ? 23 : 24)); // attention
    fhn[24] = load(fi, index_f(j[ 9], t%2ul ? 24 : 23)); // attention
    fhn[25] = load(fi, index_f(j[ 1], t%2ul ? 25 : 26)); // attention
    fhn[26] = load(fi, index_f(j[11], t%2ul ? 26 : 25)); // attention
  )+"#endif"+R( // D3Q27
  }
  )+R(void store_f(const uint n, const float* fhn, global fpxx* fi, const uint* j, const ulong t) {
    store(fi, index_f(n, 0u), fhn[0]); // Esoteric-Twist
  )+"#if defined(D2Q9)"+R(
    store(fi, index_f(j[1], t%2ul ? 2 : 1), fhn[1]);
    store(fi, index_f(n   , t%2ul ? 1 : 2), fhn[2]);
    store(fi, index_f(j[3], t%2ul ? 4 : 3), fhn[3]);
    store(fi, index_f(n   , t%2ul ? 3 : 4), fhn[4]);
    store(fi, index_f(j[5], t%2ul ? 6 : 5), fhn[5]);
    store(fi, index_f(n   , t%2ul ? 5 : 6), fhn[6]);
    store(fi, index_f(j[1], t%2ul ? 8 : 7), fhn[7]); // attention
    store(fi, index_f(j[3], t%2ul ? 7 : 8), fhn[8]); // attention
  )+"#elif defined(D3Q15)"+R(
    uint x0, xp, xm, y0, yp, ym, z0, zp, zm;
    calculate_indices(n, &x0, &xp, &xm, &y0, &yp, &ym, &z0, &zp, &zm);
    store(fi, index_f(j[ 1]   , t%2ul ?  2 :  1), fhn[ 1]);
    store(fi, index_f(n       , t%2ul ?  1 :  2), fhn[ 2]);
    store(fi, index_f(j[ 3]   , t%2ul ?  4 :  3), fhn[ 3]);
    store(fi, index_f(n       , t%2ul ?  3 :  4), fhn[ 4]);
    store(fi, index_f(j[ 5]   , t%2ul ?  6 :  5), fhn[ 5]);
    store(fi, index_f(n       , t%2ul ?  5 :  6), fhn[ 6]);
    store(fi, index_f(j[ 7]   , t%2ul ?  8 :  7), fhn[ 7]);
    store(fi, index_f(n       , t%2ul ?  7 :  8), fhn[ 8]);
    store(fi, index_f(xp+yp+z0, t%2ul ? 10 :  9), fhn[ 9]); // attention, additional streaming direction required beyond velocity set
    store(fi, index_f(j[ 5]   , t%2ul ?  9 : 10), fhn[10]); // attention
    store(fi, index_f(xp+y0+zp, t%2ul ? 12 : 11), fhn[11]); // attention, additional streaming direction required beyond velocity set
    store(fi, index_f(j[ 3]   , t%2ul ? 11 : 12), fhn[12]); // attention
    store(fi, index_f(x0+yp+zp, t%2ul ? 14 : 13), fhn[13]); // attention, additional streaming direction required beyond velocity set
    store(fi, index_f(j[ 1]   , t%2ul ? 13 : 14), fhn[14]); // attention
  )+"#elif defined(D3Q19)"+R(
    store(fi, index_f(j[ 1], t%2ul ?  2 :  1), fhn[ 1]);
    store(fi, index_f(n    , t%2ul ?  1 :  2), fhn[ 2]);
    store(fi, index_f(j[ 3], t%2ul ?  4 :  3), fhn[ 3]);
    store(fi, index_f(n    , t%2ul ?  3 :  4), fhn[ 4]);
    store(fi, index_f(j[ 5], t%2ul ?  6 :  5), fhn[ 5]);
    store(fi, index_f(n    , t%2ul ?  5 :  6), fhn[ 6]);
    store(fi, index_f(j[ 7], t%2ul ?  8 :  7), fhn[ 7]);
    store(fi, index_f(n    , t%2ul ?  7 :  8), fhn[ 8]);
    store(fi, index_f(j[ 9], t%2ul ? 10 :  9), fhn[ 9]);
    store(fi, index_f(n    , t%2ul ?  9 : 10), fhn[10]);
    store(fi, index_f(j[11], t%2ul ? 12 : 11), fhn[11]);
    store(fi, index_f(n    , t%2ul ? 11 : 12), fhn[12]);
    store(fi, index_f(j[ 1], t%2ul ? 14 : 13), fhn[13]); // attention
    store(fi, index_f(j[ 3], t%2ul ? 13 : 14), fhn[14]); // attention
    store(fi, index_f(j[ 1], t%2ul ? 16 : 15), fhn[15]); // attention
    store(fi, index_f(j[ 5], t%2ul ? 15 : 16), fhn[16]); // attention
    store(fi, index_f(j[ 3], t%2ul ? 18 : 17), fhn[17]); // attention
    store(fi, index_f(j[ 5], t%2ul ? 17 : 18), fhn[18]); // attention
  )+"#elif defined(D3Q27)"+R(
    store(fi, index_f(j[ 1], t%2ul ?  2 :  1), fhn[ 1]);
    store(fi, index_f(n    , t%2ul ?  1 :  2), fhn[ 2]);
    store(fi, index_f(j[ 3], t%2ul ?  4 :  3), fhn[ 3]);
    store(fi, index_f(n    , t%2ul ?  3 :  4), fhn[ 4]);
    store(fi, index_f(j[ 5], t%2ul ?  6 :  5), fhn[ 5]);
    store(fi, index_f(n    , t%2ul ?  5 :  6), fhn[ 6]);
    store(fi, index_f(j[ 7], t%2ul ?  8 :  7), fhn[ 7]);
    store(fi, index_f(n    , t%2ul ?  7 :  8), fhn[ 8]);
```

```
    store(fi, index_f(j[ 9], t%2ul ? 10 :  9), fhn[ 9]);
    store(fi, index_f(n    , t%2ul ?  9 : 10), fhn[10]);
    store(fi, index_f(j[11], t%2ul ? 12 : 11), fhn[11]);
    store(fi, index_f(n    , t%2ul ? 11 : 12), fhn[12]);
    store(fi, index_f(j[ 1], t%2ul ? 14 : 13), fhn[13]); // attention
    store(fi, index_f(j[ 3], t%2ul ? 13 : 14), fhn[14]); // attention
    store(fi, index_f(j[ 1], t%2ul ? 16 : 15), fhn[15]); // attention
    store(fi, index_f(j[ 5], t%2ul ? 15 : 16), fhn[16]); // attention
    store(fi, index_f(j[ 3], t%2ul ? 18 : 17), fhn[17]); // attention
    store(fi, index_f(j[ 5], t%2ul ? 17 : 18), fhn[18]); // attention
    store(fi, index_f(j[19], t%2ul ? 20 : 19), fhn[19]);
    store(fi, index_f(n    , t%2ul ? 19 : 20), fhn[20]);
    store(fi, index_f(j[ 7], t%2ul ? 22 : 21), fhn[21]); // attention
    store(fi, index_f(j[ 5], t%2ul ? 21 : 22), fhn[22]); // attention
    store(fi, index_f(j[ 9], t%2ul ? 24 : 23), fhn[23]); // attention
    store(fi, index_f(j[ 3], t%2ul ? 23 : 24), fhn[24]); // attention
    store(fi, index_f(j[11], t%2ul ? 26 : 25), fhn[25]); // attention
    store(fi, index_f(j[ 1], t%2ul ? 25 : 26), fhn[26]); // attention
)+"#endif"+R( // D3Q27
}
```

Listing 4: Esoteric-Twist implementation in OpenCL C.

8.1.5 Esoteric-Pull

```
)+R(void load_f(const uint n, float* fhn, const global fpxx* fi, const uint* j, const ulong t) {
  fhn[0] = load(fi, index_f(n, 0u)); // Esoteric-Pull
  for(uint i=1u; i<def_velocity_set; i+=2u) {
    fhn[i   ] = load(fi, index_f(n   , t%2ul ? i    : i+1u));
    fhn[i+1u] = load(fi, index_f(j[i], t%2ul ? i+1u : i   ));
  }
}
)+R(void store_f(const uint n, const float* fhn, global fpxx* fi, const uint* j, const ulong t) {
  store(fi, index_f(n, 0u), fhn[0]); // Esoteric-Pull
  for(uint i=1u; i<def_velocity_set; i+=2u) {
    store(fi, index_f(j[i], t%2ul ? i+1u : i   ), fhn[i   ]);
    store(fi, index_f(n   , t%2ul ? i    : i+1u), fhn[i+1u]);
  }
}
```

Listing 5: Esoteric-Pull implementation in OpenCL C.

8.1.6 Esoteric-Push

```
)+R(void load_f(const uint n, float* fhn, const global fpxx* fi, const uint* j, const ulong t) {
  fhn[0] = load(fi, index_f(n, 0u)); // Esoteric-Push
  for(uint i=1u; i<def_velocity_set; i+=2u) {
    fhn[i   ] = load(fi, index_f(j[i+1u], t%2ul ? i+1u : i   ));
    fhn[i+1u] = load(fi, index_f(n      , t%2ul ? i    : i+1u));
  }
}
)+R(void store_f(const uint n, const float* fhn, global fpxx* fi, const uint* j, const ulong t) {
  store(fi, index_f(n, 0u), fhn[0]); // Esoteric-Push
  for(uint i=1u; i<def_velocity_set; i+=2u) {
    store(fi, index_f(n      , t%2ul ? i    : i+1u), fhn[i   ]);
    store(fi, index_f(j[i+1u], t%2ul ? i+1u : i   ), fhn[i+1u]);
  }
}
```

Listing 6: Esoteric-Push implementation in OpenCL C.

8.2 OpenCL C implementation of the FSLBM with Esoteric-Pull

```
#define TYPE_W 0b00000001 // static boundary (wall)
#define TYPE_E 0b00000010 // equilibrium boundary or moving bounce-back boundary (inflow/outflow)
#define TYPE_F 0b00001000 // fluid
#define TYPE_I 0b00010000 // interface
#define TYPE_G 0b00100000 // gas

#define TYPE_IF 0b00011000 // change from interface to fluid
#define TYPE_IG 0b00110000 // change from interface to gas
#define TYPE_GI 0b00111000 // change from gas to interface
#define TYPE_SU 0b00111000 // any flag bit used for SURFACE

#define fpxx float // switchable data type (regular 32-bit float)
#define load(p,o) p[o] // regular float read
#define store(p,o,x) p[o]=x // regular float write

)+R(ulong index_f(const uint n, const uint i) { // 64-bit indexing (maximum 2^32 lattice points (1624^3 lattice resolution, 225GB)
  return (ulong)i*(ulong)def_N+(ulong)n; // SoA (229% faster on GPU)
}

)+R(void load_f(const uint n, float* fhn, const global fpxx* fi, const uint* j, const ulong t) {
  fhn[0] = load(fi, index_f(n, 0u)); // Esoteric-Pull
  for(uint i=1u; i<def_velocity_set; i+=2u) {
    fhn[i   ] = load(fi, index_f(n   , t%2ul ? i    : i+1u));
    fhn[i+1u] = load(fi, index_f(j[i], t%2ul ? i+1u : i   ));
  }
}
)+R(void store_f(const uint n, const float* fhn, global fpxx* fi, const uint* j, const ulong t) {
  store(fi, index_f(n, 0u), fhn[0]); // Esoteric-Pull
  for(uint i=1u; i<def_velocity_set; i+=2u) {
    store(fi, index_f(j[i], t%2ul ? i+1u : i   ), fhn[i   ]);
```

```
      store(fi, index_f(n    , t%2ul ? i    : i+1u), fhn[i+1u]);
    }
  }

  )+"#ifdef SURFACE"+R(
  )+R(void load_f_outgoing(const uint n, float* fon, const global fpxx* fi, const uint* j, const ulong t) { // load outgoing DDFs, even: 1:1 like stream-out odd, odd: 1:1 like stream-out even
    for(uint i=1u; i<def_velocity_set; i+=2u) { // Esoteric-Pull
      fon[i   ] = load(fi, index_f(j[i], t%2ul ? i    : i+1u));
      fon[i+1u] = load(fi, index_f(n    , t%2ul ? i+1u : i   ));
    }
  }
  )+R(void store_f_reconstructed(const uint n, const float* fhn, global fpxx* fi, const uint* j, const ulong t, const uchar* flagsj) { // store reconstructed gas DDFs, even: 1:1 like stream-in even, odd: 1:1 like stream-in odd
    for(uint i=1u; i<def_velocity_set; i+=2u) { // Esoteric-Pull
      if(flagsj[i+1u]&TYPE_G) store(fi, index_f(n    , t%2ul ? i    : i+1u), fhn[i   ]); // only store reconstructed gas DDFs to locations from which
      if(flagsj[i   ]&TYPE_G) store(fi, index_f(j[i], t%2ul ? i+1u : i   ), fhn[i+1u]); // they are going to be streamed in during next stream_collide()
    }
  }
  )+"#endif"+R( // SURFACE



  )+"kernel void initialize(global fpxx* fi, global float* rho, global float* u, global uchar* flags"+R( // ) {
  )+"#ifdef SURFACE"+R(
    , global float* mass, global float* massex, global float* phi // argument order is important
  )+"#endif"+R( // SURFACE
  )+") {"+R( // initialize()
    const uint n = get_global_id(0); // n = x+(y+z*Ny)*Nx
    uchar flagsn = flags[n];
    uint j[def_velocity_set]; // neighbor indices
    neighbors(n, j); // calculate neighbor indices
    uchar flagsj[def_velocity_set]; // cache neighbor flags for multiple readings
    for(uint i=1u; i<def_velocity_set; i++) flagsj[i] = flags[j[i]];
    if(flagsn&TYPE_W) { // node is wall
      bool TYPE_ONLY_W = true; // has only wall neighbors
      for(uint i=1u; i<def_velocity_set; i++) if(!(flagsj[i]&TYPE_W)) TYPE_ONLY_W = TYPE_ONLY_W&&(flagsj[i]&TYPE_W);
      if(TYPE_ONLY_W) {
        u[                      n] = 0.0f; // reset velocity for wall lattice points with only boundary neighbors
        u[    (ulong)def_N+(ulong)n] = 0.0f;
        u[2ul*(ulong)def_N+(ulong)n] = 0.0f;
      }
  )+"#ifndef MOVING_BOUNDARIES"+R(
      if(flagsn&TYPE_W) {
        u[                      n] = 0.0f; // reset velocity for all wall lattice points
        u[    (ulong)def_N+(ulong)n] = 0.0f;
        u[2ul*(ulong)def_N+(ulong)n] = 0.0f;
      }
  )+"#else"+R( // MOVING_BOUNDARIES
    } else if(!(flagsn&TYPE_T)) { // local lattice point is not wall
      bool next_to_moving_boundary = false;
      for(uint i=1u; i<def_velocity_set; i++) {
        next_to_moving_boundary |= (u[j[i]]!=0.0f||u[(ulong)def_N+(ulong)j[i]]!=0.0f||u[2ul*(ulong)def_N+(ulong)j[i]]!=0.0f)&&(flagsj[i]&TYPE_W);
      }
      if(next_to_moving_boundary) flags[n] = flagsn = flagsn|TYPE_E; // only flag lattice points next to a wall with non-zero velocity with TYPE_E
  )+"#endif"+R( // MOVING_BOUNDARIES
    }
    float feq[def_velocity_set]; // f_equilibrium
    calculate_f_eq(rho[n], u[n], u[(ulong)def_N+(ulong)n], u[2ul*(ulong)def_N+(ulong)n], feq);
  )+"#ifdef SURFACE"+R( // automatically generate the interface layer between fluid and gas
    { // separate block to avoid variable name conflicts
      float phin = phi[n];
      if(!(flagsn&(TYPE_W|TYPE_E|TYPE_T|TYPE_F|TYPE_I))) flagsn = (flagsn&~TYPE_SU)|TYPE_G; // change all non-fluid and non-interface flags to gas
      if(flagsn&TYPE_G) { // node is gas
        bool change = false; // check if node has to be changed to interface
        for(uint i=1u; i<def_velocity_set; i++) change = change||(flagsj[i]&TYPE_F); // if neighbor flag fluid is set, the node must be interface
        if(change) { // create interface automatically if phi has not explicitely defined for the interface layer
          flagsn = (flagsn&~TYPE_SU)|TYPE_I; // node must be interface
          phin = 0.5f;
          float rhon, uxn, uyn, uzn; // initialize interface nodes with average density/velocity of fluid neighbors
          average_neighbors_fluid(n, rho, u, flags, &rhon, &uxn, &uyn, &uzn); // get average rho/u from all fluid neighbors
          calculate_f_eq(rhon, uxn, uyn, uzn, feq); // calculate equilibrium DDFs
        }
      }
      if(flagsn&TYPE_G) { // node is still gas
        u[                      n] = 0.0f; // reset velocity for gas nodes
        u[    (ulong)def_N+(ulong)n] = 0.0f;
        u[2ul*(ulong)def_N+(ulong)n] = 0.0f;
        phin = 0.0f;
      } else if((flagsn&TYPE_I) && (phin<0.0f||phin>1.0f)) {
        phin = 0.5f; // node should be interface, but phi was invalid
      } else if(flagsn&TYPE_F) {
        phin = 1.0f;
      }
      phi[n] = phin;
      mass[n] = phin*rho[n];
      massex[n] = 0.0f; // reset excess mass
      flags[n] = flagsn;
    }
  )+"#endif"+R( // SURFACE
    store_f(n, feq, fi, j, 1ul); // write to fi
  } // initialize()



  )+"kernel void stream_collide(global fpxx* fi, global float* rho, global float* u, global uchar* flags, const ulong t"+R( // ) {  // for all but SURFACE, flags is const
  )+"#ifdef SURFACE"+R(
    , global float* mass // argument order is important
  )+"#endif"+R( // SURFACE
  )+") {"+R( // stream_collide()
    const uint n = get_global_id(0); // n = x+(y+z*Ny)*Nx
    const uchar flagsn = flags[n]; // cache flags[n] for multiple readings
    if(flagsn&(TYPE_W|TYPE_G)) return; // if node is boundary or gas, just return

    uint j[def_velocity_set]; // neighbor indices
    neighbors(n, j); // calculate neighbor indices

  )+"#if defined(MOVING_BOUNDARIES)||defined(SURFACE)"+R( // don't leave any spaces in #if arguments
    uchar flagsj[def_velocity_set]; // only required for MOVING_BOUNDARIES, SURFACE or TEMPERATURE
```

```
    flagsj[0] = flagsn;
    for(uint i=1u; i<def_velocity_set; i++) flagsj[i] = flags[j[i]];
  )+"#endif"+R( // MOVING_BOUNDARIES or SURFACE

    float fhn[def_velocity_set]; // local DDFs
    load_f(n, fhn, fi, j, t); // perform streaming (part 1)

  )+"#ifdef MOVING_BOUNDARIES"+R(
    if(flagsn&TYPE_E) { // apply Dirichlet velocity boundaries if necessary (reads velocities of only neighboring boundary nodes, which do not change during simulation)
      uint ji;
      for(uint i=1u; i<def_velocity_set; i+=2u) { // Krueger p.180, rho_wall=1, loop is totally unrolled by compiler
        const float w6 = -6.0f*w(i); // w(i) = w(i+1) if i is odd
        ji = j[i+1u]; fhn[i   ] = flagsj[i+1u]&TYPE_W ? fma(w6, c(i+1u)*u[ji]+c(def_velocity_set+i+1u)*u[(ulong)def_N+(ulong)ji]+c(2u*def_velocity_set+i+1u)*u[2ul*(ulong)def_N+(ulong)ji], fhn[i   ]) : fhn[i   ]; // boundary : regular
        ji = j[i   ]; fhn[i+1u] = flagsj[i   ]&TYPE_W ? fma(w6, c(i   )*u[ji]+c(def_velocity_set+i   )*u[(ulong)def_N+(ulong)ji]+c(2u*def_velocity_set+i   )*u[2ul*(ulong)def_N+(ulong)ji], fhn[i+1u]) : fhn[i+1u];
      }
    }
  )+"#endif"+R( // MOVING_BOUNDARIES

    float rhon, uxn, uyn, uzn; // calculate local density and velocity for collision
  )+"#ifndef EQUILIBRIUM_BOUNDARIES"+R(
    calculate_rho_u(fhn, &rhon, &uxn, &uyn, &uzn); // calculate density and velocity fields from fi
  )+"#else"+R( // EQUILIBRIUM_BOUNDARIES
    if(flagsn&TYPE_E) {
      rhon = rho[                      n]; // apply preset velocity/density
      uxn  = u[                        n];
      uyn  = u[    (ulong)def_N+(ulong)n];
      uzn  = u[2ul*(ulong)def_N+(ulong)n];
    } else {
      calculate_rho_u(fhn, &rhon, &uxn, &uyn, &uzn); // calculate density and velocity fields from fi
    }
  )+"#endif"+R( // EQUILIBRIUM_BOUNDARIES
    float fxn=def_fx, fyn=def_fy, fzn=def_fz; // force starts as constant volume force, can be modified before call of calculate_forcing_terms(...)
    float Fin[def_velocity_set]; // forcing terms

    { // separate block to avoid variable name conflicts
  )+"#ifdef VOLUME_FORCE"+R( // apply force and collision operator, write to fi in video memory
      const float rho2 = 0.5f/rhon; // apply external volume force (Guo forcing, Krueger p.233f)
      uxn = clamp(fma(fxn, rho2, uxn), -def_c, def_c); // limit velocity (for stability purposes)
      uyn = clamp(fma(fyn, rho2, uyn), -def_c, def_c); // force term: F*dt/(2*rho)
      uzn = clamp(fma(fzn, rho2, uzn), -def_c, def_c);
      calculate_forcing_terms(uxn, uyn, uzn, fxn, fyn, fzn, Fin); // calculate volume force terms Fin from velocity field (Guo forcing, Krueger p.233 f)
  )+"#else"+R( // VOLUME_FORCE
      uxn = clamp(uxn, -def_c, def_c); // limit velocity (for stability purposes)
      uyn = clamp(uyn, -def_c, def_c); // force term: F*dt/(2*rho)
      uzn = clamp(uzn, -def_c, def_c);
      for(uint i=0u; i<def_velocity_set; i++) Fin[i] = 0.0f;
  )+"#endif"+R( // VOLUME_FORCE
    }

  )+"#ifdef SURFACE"+R(
    if(flagsn&TYPE_I) { // node was interface, eventually initiate flag change
      bool TYPE_NO_F=true, TYPE_NO_G=true; // temporary flags for no fluid or gas neighbors
      for(uint i=1u; i<def_velocity_set; i++) {
        const uchar flagsji_su = flags[j[i]]&TYPE_SU; // extract SURFACE flags
        TYPE_NO_F = TYPE_NO_F&&flagsji_su!=TYPE_F;
        TYPE_NO_G = TYPE_NO_G&&flagsji_su!=TYPE_G;
      }
      const float massn = mass[n]; // load mass
           if(massn>rhon || TYPE_NO_G) flags[n] = (flagsn&~TYPE_SU)|TYPE_IF; // set flag interface->fluid
      else if(massn<0.0f || TYPE_NO_F) flags[n] = (flagsn&~TYPE_SU)|TYPE_IG; // set flag interface->gas
    }
  )+"#endif"+R( // SURFACE

  )+"#ifndef EQUILIBRIUM_BOUNDARIES"+R(
  )+"#ifdef UPDATE_FIELDS"+R(
    rho[                      n] = rhon; // update density field
    u[                        n] = uxn; // update velocity field
    u[    (ulong)def_N+(ulong)n] = uyn;
    u[2ul*(ulong)def_N+(ulong)n] = uzn;
  )+"#endif"+R( // UPDATE_FIELDS
  )+"#else"+R( // EQUILIBRIUM_BOUNDARIES
  )+"#ifdef UPDATE_FIELDS"+R(
    if(!(flagsn&TYPE_E)) {
      rho[                      n] = rhon; // update density field
      u[                        n] = uxn; // update velocity field
      u[    (ulong)def_N+(ulong)n] = uyn;
      u[2ul*(ulong)def_N+(ulong)n] = uzn;
    }
  )+"#endif"+R( // UPDATE_FIELDS
  )+"#endif"+R( // EQUILIBRIUM_BOUNDARIES

    float feq[def_velocity_set]; // equilibrium DDFs
    calculate_f_eq(rhon, uxn, uyn, uzn, feq); // calculate equilibrium DDFs

  )+"#ifdef VOLUME_FORCE"+R(
    const float c_tau = fma(def_w, -0.5f, 1.0f);
    for(uint i=0u; i<def_velocity_set; i++) Fin[i] *= c_tau;
  )+"#endif"+R( // VOLUME_FORCE
  )+"#ifndef EQUILIBRIUM_BOUNDARIES"+R(
    for(uint i=0u; i<def_velocity_set; i++) fhn[i] = fma(1.0f-def_w, fhn[i], fma(def_w, feq[i], Fin[i])); // perform collision (SRT)
  )+"#else"+R( // EQUILIBRIUM_BOUNDARIES
    for(uint i=0u; i<def_velocity_set; i++) fhn[i] = flagsn&TYPE_E ? feq[i] : fma(1.0f-def_w, fhn[i], fma(def_w, feq[i], Fin[i])); // perform collision (SRT)
  )+"#endif"+R( // EQUILIBRIUM_BOUNDARIES

    store_f(n, fhn, fi, j, t); // perform streaming (part 2)
  } // stream_collide()



  )+"#ifdef SURFACE"+R(
  )+R(kernel void surface_0(global fpxx* fi, const global float* rho, const global float* u, const global uchar* flags, global float* mass, const global float* massex, const global float* phi, const ulong t) { // capture outgoing DDFs before streaming
    const uint n = get_global_id(0); // n = x+(y+z*Ny)*Nx
    const uchar flagsn = flags[n]; // cache flags[n] for multiple readings
    if(flagsn&(TYPE_W|TYPE_G)) return; // node processed here is fluid or interface

```

```
    uint j[def_velocity_set]; // neighbor indices
    neighbors(n, j); // calculate neighbor indices
    float fhn[def_velocity_set]; // incoming DDFs
    load_f(n, fhn, fi, j, t); // load incoming DDFs
    float fon[def_velocity_set]; // outgoing DDFs
    fon[0] = fhn[0]; // fon[0] is already loaded in fhn[0]
    load_f_outgoing(n, fon, fi, j, t); // load outgoing DDFs

    float massn = mass[n];
    for(uint i=1u; i<def_velocity_set; i++) {
      massn += massex[j[i]]; // distribute excess mass from last step which is stored in neighbors
    }
    if(flagsn&TYPE_F) { // node is fluid
      for(uint i=1u; i<def_velocity_set; i++) massn += fhn[i]-fon[i]; // neighbor is fluid or interface node
    } else if(flagsn&TYPE_I) { // node is interface
      float phij[def_velocity_set]; // cache fill level of neighbor lattice points
      for(uint i=1u; i<def_velocity_set; i++) phij[i] = phi[j[i]]; // cache fill level of neighbor lattice points
      float rhon, uxn, uyn, uzn, rho_laplace=0.0f; // no surface tension if rho_laplace is not overwritten later
)+"#ifndef EQUILIBRIUM_BOUNDARIES"+R(
      calculate_rho_u(fon, &rhon, &uxn, &uyn, &uzn); // calculate density and velocity fields from fon (not fhn)
)+"#else"+R( // EQUILIBRIUM_BOUNDARIES
      if(flagsn&TYPE_E) {
        rhon = rho[                      n]; // apply preset velocity/density
        uxn  = u[                        n];
        uyn  = u[    (ulong)def_N+(ulong)n];
        uzn  = u[2ul*(ulong)def_N+(ulong)n];
      } else {
        calculate_rho_u(fon, &rhon, &uxn, &uyn, &uzn); // calculate density and velocity fields from fon (not fhn)
      }
)+"#endif"+R( // EQUILIBRIUM_BOUNDARIES
      uxn = clamp(uxn, -def_c, def_c); // limit velocity (for stability purposes)
      uyn = clamp(uyn, -def_c, def_c);
      uzn = clamp(uzn, -def_c, def_c);
      phij[0] = calculate_phi(rhon, massn, flagsn); // don't load phi[n] from memory, instead recalculate it with mass corrected by excess mass
      rho_laplace = def_6_sigma==0.0f ? 0.0f : def_6_sigma*curvature_calculation(n, phij, phi); // surface tension least squares fit (PLIC, most
          ↪ accurate)
      float feg[def_velocity_set]; // reconstruct f from neighbor gas lattice points
      const float rho2tmp = 0.5f/rhon; // apply external volume force (Guo forcing, Krueger p.233f)
      const float uxntmp = clamp(fma(def_fx, rho2tmp, uxn), -def_c, def_c); // limit velocity (for stability purposes)
      const float uyntmp = clamp(fma(def_fy, rho2tmp, uyn), -def_c, def_c); // force term: F*dt/(2*rho)
      const float uzntmp = clamp(fma(def_fz, rho2tmp, uzn), -def_c, def_c);
      calculate_f_eq(1.0f-rho_laplace, uxntmp, uyntmp, uzntmp, feg); // calculate gas equilibrium DDFs with constant ambient pressure
      uchar flagsj[def_velocity_set]; // cache neighbor flags for multiple readings
      for(uint i=1u; i<def_velocity_set; i++) flagsj[i] = flagsj[j[i]];
      for(uint i=1u; i<def_velocity_set; i+=2u) { // calculate mass exchange between current node and fluid/interface nodes
        massn += flagsj[i   ]&(TYPE_F|TYPE_I) ? flagsj[i   ]&TYPE_F ? fhn[i+1]-fon[i   ] : 0.5f*(phij[i   ]+phij[0])*(fhn[i+1 ]-fon[i   ]) : 0.0f; //
            ↪  neighbor is fluid or interface node
        massn += flagsj[i+1u]&(TYPE_F|TYPE_I) ? flagsj[i+1u]&TYPE_F ? fhn[i  ]-fon[i+1u] : 0.5f*(phij[i+1u]+phij[0])*(fhn[i   ]-fon[i+1u]) : 0.0f; //
            ↪  fluid : interface : gas
      }
      for(uint i=1u; i<def_velocity_set; i+=2u) { // calculate reconstructed gas DDFs
        fhn[i   ] = feg[i+1u]-fon[i+1u]+feg[i   ];
        fhn[i+1u] = feg[i   ]-fon[i   ]+feg[i+1u];
      }
      store_f_reconstructed(n, fhn, fi, j, t, flagsj); // store reconstructed gas DDFs that are streamed in during the following stream_collide()
    }
    mass[n] = massn;
  }
  )+R(kernel void surface_1(global uchar* flags) { // prevent neighbors from interface->fluid nodes to become/be gas nodes
    const uint n = get_global_id(0); // n = x+(y+z*Ny)*Nx
    const uchar flagsn_su = flags[n]&(TYPE_SU|TYPE_W); // extract SURFACE flags
    if(flagsn_su==TYPE_IF) { // flag interface->fluid is set
      uint j[def_velocity_set]; // neighbor indices
      neighbors(n, j); // calculate neighbor indices
      for(uint i=1u; i<def_velocity_set; i++) {
        const uchar flagsji = flags[j[i]];
        const uchar flagsji_su = flagsji&(TYPE_SU|TYPE_W); // extract SURFACE flags
        const uchar flagsji_r = flagsji&~TYPE_SU; // extract all non-SURFACE flags
        if(flagsji_su==TYPE_IG) flags[j[i]] = flagsji_r|TYPE_I; // prevent interface neighbor nodes from becoming gas
        else if(flagsji_su==TYPE_G) flags[j[i]] = flagsji_r|TYPE_GI; // neighbor node was gas and must change to interface
      }
    }
  } // possible types at the end of surface_1(): TYPE_F / TYPE_I / TYPE_G / TYPE_IF / TYPE_IG / TYPE_GI
  )+R(kernel void surface_2(global fpxx* fi, const global float* rho, const global float* u, global uchar* flags, const ulong t) {  // apply flag
      ↪ changes and calculate excess mass
    const uint n = get_global_id(0); // n = x+(y+z*Ny)*Nx
    const uchar flagsn_su = flags[n]&(TYPE_SU|TYPE_W); // extract SURFACE flags
    if(flagsn_su==TYPE_GI) { // initialize the fi of gas nodes that should become interface
      float rhon, uxn, uyn, uzn; // average over all fluid/interface neighbors
      average_neighbors_non_gas(n, rho, u, flags, &rhon, &uxn, &uyn, &uzn); // get average rho/u from all fluid/interface neighbors
      float feq[def_velocity_set];
      calculate_f_eq(rhon, uxn, uyn, uzn, feq); // calculate equilibrium DDFs
      uint j[def_velocity_set];
      neighbors(n, j);
      store_f(n, feq, fi, j, t); // write feq to fi in video memory
    } else if(flagsn_su==TYPE_IG) { // flag interface->gas is set
      uint j[def_velocity_set]; // neighbor indices
      neighbors(n, j); // calculate neighbor indices
      for(uint i=1u; i<def_velocity_set; i++) {
        const uchar flagsji = flags[j[i]];
        const uchar flagsji_su = flagsji&(TYPE_SU|TYPE_W); // extract SURFACE flags
        const uchar flagsji_r = flagsji&~TYPE_SU; // extract all non-SURFACE flags
        if(flagsji_su==TYPE_F||flagsji_su==TYPE_IF) {
          flags[j[i]] = flagsji_r|TYPE_I; // prevent fluid or interface neighbors that turn to fluid from being/becoming fluid
        }
      }
    }
  } // possible types at the end of surface_2(): TYPE_F / TYPE_I / TYPE_G / TYPE_IF / TYPE_IG / TYPE_GI
  )+R(kernel void surface_3(const global float* rho, global uchar* flags, global float* mass, global float* massex, global float* phi) { // apply
      ↪ flag changes and calculate excess mass
    const uint n = get_global_id(0); // n = x+(y+z*Ny)*Nx
    const uchar flagsn_su = flags[n]&(TYPE_SU|TYPE_W); // extract SURFACE flags
    if(flagsn_su&TYPE_W) return;
    const float rhon = rho[n]; // density of node n
    float massn = mass[n]; // mass of node n
    float massexn = 0.0f; // excess mass of node n
    float phin = 0.0f;
    if(flagsn_su==TYPE_F) { // regular fluid node
      massexn = massn-rhon; // dump mass-rho difference into excess mass
      massn = rhon; // fluid node mass has to equal rho
      phin = 1.0f;
```

```
  } else if(flagsn_su==TYPE_I) { // regular interface node
    massexn = massn>rhon ? massn-rhon : massn<0.0f ? massn : 0.0f; // allow interface nodes with mass>rho or mass<0
    massn = clamp(massn, 0.0f, rhon);
    phin = calculate_phi(rhon, massn, TYPE_I); // calculate fill level for next step (only necessary for interface nodes)
  } else if(flagsn_su==TYPE_G) { // regular gas node
    massexn = massn; // dump remaining mass into excess mass
    massn = 0.0f;
    phin = 0.0f;
  } else if(flagsn_su==TYPE_IF) { // flag interface->fluid is set
    flags[n] = (flags[n]&~TYPE_SU)|TYPE_F; // node becomes fluid
    massexn = massn-rhon; // dump mass-rho difference into excess mass
    massn = rhon; // fluid node mass has to equal rho
    phin = 1.0f; // set phi[n] to 1.0f for fluid nodes
  } else if(flagsn_su==TYPE_IG) { // flag interface->gas is set
    flags[n] = (flags[n]&~TYPE_SU)|TYPE_G; // node becomes gas
    massexn = massn; // dump remaining mass into excess mass
    massn = 0.0f; // gas mass has to be zero
    phin = 0.0f; // set phi[n] to 0.0f for gas nodes
  } else if(flagsn_su==TYPE_GI) { // flag gas->interface is set
    flags[n] = (flags[n]&~TYPE_SU)|TYPE_I; // node becomes interface
    massexn = massn>rhon ? massn-rhon : massn<0.0f ? massn : 0.0f; // allow interface nodes with mass>rho or mass<0
    massn = clamp(massn, 0.0f, rhon);
    phin = calculate_phi(rhon, massn, TYPE_I); // calculate fill level for next step (only necessary for interface nodes)
  }
  uint j[def_velocity_set]; // neighbor indices
  neighbors(n, j); // calculate neighbor indices
  uint counter = 0u; // count (fluid|interface) neighbors
  for(uint i=1u; i<def_velocity_set; i++) { // simple model: distribute excess mass equally to all interface and fluid neighbors
    const uchar flagsji_su = flags[j[i]]&(TYPE_SU|TYPE_W); // extract SURFACE flags
    counter += (uint)(flagsji_su==TYPE_F||flagsji_su==TYPE_I||flagsji_su==TYPE_IF||flagsji_su==TYPE_GI); // avoid branching
  }
  massn += counter>0u ? 0.0f : massexn; // if excess mass can't be distributed to neighboring interface or fluid nodes, add it to local mass (
      ↪ ensure mass conservation)
  massexn = counter>0u ? massexn/(float)counter : 0.0f; // divide excess mass up for all interface or fluid neighbors
  mass[n] = massn; // update mass
  massex[n] = massexn; // update excess mass
  phi[n] = phin; // update phi
} // possible types at the end of surface_3(): TYPE_F / TYPE_I / TYPE_G
)+"#endif"+R( // SURFACE
```

Listing 7: OpenCL C FSLBM implementation as in *FluidX3D* [13, 14, 24].

8.3 Setup script for figure 8

```
#include "lbm.hpp"
void main_setup() { // raindrop setup (#define D3Q19, SRT, VOLUME_FORCE, EQUILIBRIUM_BOUNDARIES, SURFACE, FP16C)
  const int box_diameter = 940;
  float drop_diameter = box_diameter/5;
  const int select_drop_size = 12;
  const float alpha_sim = 20.0f;

  if(drop_diameter==-1.0f) drop_diameter = 0.1f*(float)box_diameter;
  const float scale = (float)box_diameter/(10.0f*drop_diameter); // 256.0f/400.0f; // 1.0f;

  // rain drop parameters from "Effects of Altitude on Maximum Raindrop Size and Fall Velocity as Limited by Collisional Breakup, Fig. 3" in SI-
      ↪ units
  float const si_nu = 1.0508E-6f; // kinematic shear viscosity [m^2/s] at 20C and 35g/l salinity
  const float si_rho = 1024.8103f; // fluid density [kg/m^3] at 20C and 35g/l salinity
  const float si_sigma = 73.81E-3f; // fluid surface tension [kg/s^2] at 20C and 35g/l salinity
  const float si_g = 9.81f; // gravitational acceleration [m/s^2]
  const float alpha = alpha_sim; // impact angle [degree], 0 = vertical
  const float si_Ds[] = { 1.0E-3f, 1.5E-3f, 2.0E-3f, 2.5E-3f, 3.0E-3f, 3.5E-3f, 4.0E-3f, 4.5E-3f, 5.0E-3f, 5.5E-3f, 6.0E-3f, 6.5E-3f, 7.0E-3f, 4.1E
      ↪ -3f };
  const float si_us[] = { 4.50f,   5.80f,   6.80f,   7.55f,   8.10f,   8.45f,   8.80f,   9.05f,   9.20f,   9.30f,   9.40f,   9.45f,   9.55f,   7.21
      ↪ f };
  const float si_D = si_Ds[select_drop_size]; // drop diameter [m] (1-7mm)
  const float si_u = si_us[select_drop_size]; // impact velocity [m/s] (4.50-9.55m/s)
  const float si_H  =  4.0f*si_D*scale; // liquid pool height [m] (4*D is sufficient for deep pool)
  const float si_Lx = 10.0f*si_D*scale; // simulation box width [m]
  const float si_Lz =  8.5f*si_D*scale; // simulation box height [m]

  // determine a length, a velocity and the mean density in simulation units
  const float Lx = (float)box_diameter; // simulation box width
  const float u = 0.05f; // impact velocity
  const float rho = 1.0f; // density
  units.set_m_kg_s(Lx, u, rho, si_Lx, si_u, si_rho); // calculate 3 independent conversion factors (m, kg, s)
  print_info("D  = "+to_string(si_D, 6u));
  print_info("Re = "+to_string(units.si_Re(si_D, si_u, si_nu), 6u));
  print_info("We = "+to_string(units.si_We(si_D, si_u, si_rho, si_sigma), 6u));
  print_info("Fr = "+to_string(units.si_Fr(si_D, si_u, si_g), 6u));
  print_info("Ca = "+to_string(units.si_Ca(si_u, si_rho, si_nu, si_sigma), 6u));
  print_info("Bo = "+to_string(units.si_Bo(si_D, si_rho, si_g, si_sigma), 6u));
  print_info("10ms = "+to_string(units.t(0.01f))+" LBM steps");
  const float nu = units.nu(si_nu); // calculate values for remaining parameters in simulation units
  const float sigma = units.sigma(si_sigma);
  const float f = units.f(si_rho, si_g); // force per volume
  const float Lz = 800;//units.x(si_Lz);
  const float H = units.x(si_H);
  const float R = 0.5f*units.x(si_D); // drop radius
  // define simulation box size, viscosity and volume force
  LBM lbm(to_uint(Lx), to_uint(Lx), to_uint(Lz), nu, 0.0f, 0.0f, -f, sigma, 0.5f); // largest box size on Titan Xp with FP32: 384^2, FP16: 464^3

  const uint N=lbm.get_N(), Nx=lbm.get_Nx(), Ny=lbm.get_Ny(), Nz=lbm.get_Nz(); for(uint n=0u, x=0u, y=0u, z=0u; n<N; n++, lbm.coordinates(n, x, y,
      ↪ z)) {
    // define geometry
    lbm.rho[n] = rho; // set density everywhere
    if(sphere(x, y, z, float3(0.5f*Nx, 0.5f*Ny-2.0f*R*tan(alpha*pif/180.0f), H+R+2.5f), R+2.0f)) {
      const float b = sphere_plic(x, y, z, float3(0.5f*Nx, 0.5f*Ny-2.0f*R*tan(alpha*pif/180.0f), H+R+2.5f), R);
      if(b!=-1.0f) {
        lbm.u.y[n] =  sin(alpha*pif/180.0f)*u;//+random_symmetric(0.1f); // break symmetry by initializing with noise
        lbm.u.z[n] = -cos(alpha*pif/180.0f)*u;//+random_symmetric(0.1f); // break symmetry by initializing with noise
        if(b==1.0f) {
          lbm.flags[n] = TYPE_F;
          lbm.phi[n] = 1.0f;
        } else {
```

```
58            lbm.flags[n] = TYPE_I;
59            lbm.phi[n] = b;
60          }
61        }
62      }
63      if(z==0) lbm.flags[n] = TYPE_W;
64      else if(z==H) {
65        lbm.flags[n] = TYPE_I;
66        lbm.phi[n] = 0.5f; // not strictly necessary, but should be clearer (phi is automatically initialized to 0.5f for TYPE_I if not initialized)
67      } else if(z<H) lbm.flags[n] = TYPE_F;
68      else if(z==Nz-1) { // make drops that hit the simulation box ceiling disappear
69        lbm.rho[n] = 0.5f;
70        lbm.flags[n] = TYPE_E;
71      }
72    }
73    lbm.run(0u);
74    key_1 = false; // turn off boundary
75    key_6 = true; // turn on surface raytracing
76    lbm.graphics.set_camera(-30.0f, 20.0f, 100.0f, 1.0f);
77    Clock clock;
78
79    // image
80    lbm.run(7700u/*units.t(0.0015f)*/);
81    print_info("compute time: "+print_time(clock.stop()));
82    clock.start();
83    lbm.graphics.write_frame();
84    print_info("render time: "+to_string(clock.stop(), 3u));
85
86    // video
87    //lbm.graphics.write_frame();
88    //while(units.si_t(lbm.get_t())<0.0015f) {
89    //  lbm.run(units.t(0.0015f)/300u);
90    //  lbm.graphics.write_frame();
91    //}
92    //print_info("compute + render time: "+to_string(clock.stop(), 3u));
93  } /**/
```

Listing 8: C++ setup script for the raindrop impact simulation of figure 8.

8.5 Publication 5

Tracing the horizontal transport of microplastics on rough surfaces

by

Hannes Laermanns, Moritz Lehmann, Marcel Klee,
Martin GJ Löder, Stephan Gekle, and Christina Bogner

Microplastics and Nanoplastics 1.11 (2021), pp. 1–12
https://doi.org/10.1186/s43591-021-00010-2

Laermanns *et al. Microplastics and Nanoplastics* (2021) 1:11
https://doi.org/10.1186/s43591-021-00010-2

Microplastics and Nanoplastics

RESEARCH ARTICLE Open Access

Tracing the horizontal transport of microplastics on rough surfaces

Hannes Laermanns[1], Moritz Lehmann[2], Marcel Klee[1], Martin G. J. Löder[3], Stephan Gekle[2] and Christina Bogner[1*]

Abstract

Occurrence and distribution of microplastics in different ecosystems have recently become subjects of numerous studies. However, to date the research has focused mainly on marine and freshwater ecosystems and widely neglected terrestrial environments. Only recently, first studies investigated the microplastics contamination of soils. Therefore, we know little about the transport mechanisms of microplastics in soils and sediments and virtually nothing about their surface transport. In this study we investigate surface transport mechanisms by tracking fluorescent, irregularly shaped polymethyl methacrylate (PMMA) particles in real time in a laboratory setup. In 108 experimental runs, we vary the irrigation rates, inclinations and surface roughnesses. Additionally, we simulate the small-scale flow patterns to resolve the role of the roughness-induced microrelief. Our results suggest that microplastics are transported along preferential pathways resulting from the micro- and macrorelief, which can be correlated to the flow pattern observed in the computer simulation. Our model study facilitates a deeper insight into microplastic transport on different soil surfaces and serves as a pilot for investigating mechanisms of horizontal microplastic transport. However, microplastics are a diverse group of contaminants with varying shapes, densities and sizes. Therefore, for a full understanding of transport of microplastics in terrestrial environments, it is important to address these properties as well as more variable surfaces for horizontal migration and to include vertical transport mechanisms in future research.

Keywords: Microplastics, Surface transport, Surface roughness, PMMA, sCMOS camera, Particle tracking

Introduction

The large-scale production of synthetic polymers started in the 1950s and over the last seven decades the amount of produced plastic has exceeded eight billion tons [1]. In 2019, plastic production reached 57.9 million tons in Europe and 368 million tons globally [2]. Plastic products are durable and resist to degradation [1], however, the accompanying environmental challenges caused by the increasing amount of plastic waste entering the environment have been neglected for a long time. Due to its poor biodegradability, plastic waste could remain in the environment for centuries leading to an accumulation and a potential threat to our ecosystems [1, 3]. While for a long time only large items of plastic waste have been studied, microplastics came more into focus during the last decade [3]. Microplastics are in general defined as particles smaller than 5 mm [4] and occur as primary and secondary microplastics. Primary microplastics are intentionally produced micro particles like pellets and beads. Secondary microplastics originate from abrasion of plastic products or fragmentation of macroplastics by a combination of e.g. elevated temperature, light, exposure to water, friction from wind and water or exposure to organisms, resulting in thermo-oxidative degradation, photo-oxidation, hydrolysis, mechanical breakdown and biodegradation [1, 5, 6]. Due to their manifold origins and degradation pathways, microplastics are a diverse group of contaminants with a broad range of shapes, densities, sizes, polymers and additives [7]. It is best to think about

*Correspondence: christina.bogner@uni-koeln.de
[1]Ecosystem Research Group, Institute of Geography, Faculty of Mathematics and Natural Sciences, University of Cologne, Albertus Magnus Platz, 50923 Cologne, Germany
Full list of author information is available at the end of the article

their properties as a continuum rather than in discrete categories [8]. Today microplastics are found in every environmental compartment and region of the globe from megacities to even the remotest arctic areas [9–11].

While the contamination of oceans with microplastics and their transport in the marine environment have become a widely considered research field, our knowledge on the distribution and transport of microplastics in terrestrial environments is still limited [10, 12–14]. This is partly due to the analytical challenges because the extraction of microplastics from mineral and organic-rich soils and sediments is much more complicated than from aquatic samples [15–17]. Additionally, we still lack a standard analytical protocol which is a prerequisite for a proper comparison of data from different studies [18, 19]. Furthermore, the distribution of microplastics in the terrestrial environments depends on diverse sources and transport pathways. Sources can be manifold and include e.g. the introduction of microplastics via sewage sludge, compost application or plastic mulching in agriculture, littering or waste mismanagement and the fragmentation and abrasion of plastic particles from plastic products with outdoor use [17, 20, 21]. Once in the environment, microplastics may be transported by e.g. wind and water, and may finally be deposited and potentially accumulate in different environmental compartments [20, 22, 23]. Indeed, terrestrial soils are suggested as a large sink for microplastics [10, 24, 25].

While first experimental evidence on factors affecting vertical transport processes of microplastics in soils and sediments like e.g. particle size or presence of preferential flow paths are published [26–30], studies on horizontal transport are lacking to date [20, 31]. To fill this gap, tracking microplastics during their transport could generate the necessary data and help to understand the relevant transport parameters and mechanisms. Particle tracking, also referred to as particle tracing [32], has a long history in sedimentology [33]. In this discipline it is understood as the analysis of the spatial and temporal distribution of single natural or artificial components of the sediment that are marked and observed during transport [32]. In environmental sciences, sand particles are tracked for example along fluvial trajectories [34], along coasts [35] or on tidal inlets [36]. Furthermore, particles can also be tracked under laboratory condition on a far smaller scale including vertical and horizontal transport pathways with a high temporal and spatial resolution [37–40].

Nowadays specific marking or tagging the particles of interest with an identifiable signature largely facilitates the tracking [32]. Since the use of radioactivity has shown its limitations and is seen critically due to environmental and health concerns, fluorescent dyes are commonly used [32, 41]. Therefore, rhodamine or anthracene serve frequently as fluorescent coatings in combination with additive binding agents for particle tracking [32, 42]. Advances in real-time imaging and image processing allow to trace temporal and spacial distribution of particles, at least under laboratory conditions. Hardy and colleagues [40], for example, developed a real-time approach to trace the movement of sand particles with fluorescent coating during rainfall events by using fluorescent videography techniques. They could visualise and measure transport parameters like direction and travel distance of sand particles. However, most of the studies dealing with particle tracing focus on the spatial distribution as a result of transport processes. Real-time tracking of (natural) particles as demonstrated by Hardy and colleagues in a laboratory and field setup for terrestrial environments remains an exception [40, 43]. Nizzetto and colleagues [31] suggested in their theoretical study that microplastics could be transported similarly to natural particles. However, data to confirm this statement and to parametrize models is lacking [44]. Nevertheless, such data are of utmost importance for a mechanistic understanding of transport processes of microplastics in the environment.

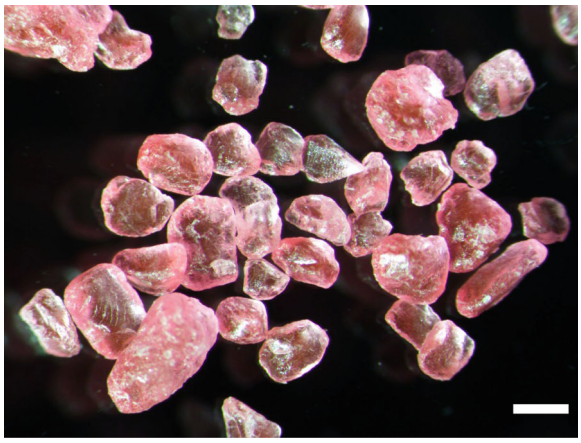


Fig. 1 A subsample of PMMA particles used in the experiments. The figure was cropped, reduced in size and the scale bar enlarged for better visibility in ImageJ [45]. The scale is 1 mm large

In this study, we constructed a laboratory setup for real-time particle tracing and observed fluorescent microplastics in irrigation experiments. We analysed the influence of different irrigation rates, inclinations and surface roughnesses on the movement of irregularly shaped polymethyl methacrylate microplastics.

Material and methods

Microplastic particles

We used UV-fluorescence-labelled polymethyl methacrylate (PMMA) particles. PMMA, also known as acrylic glass, has a density of 1.18 to 1.19 g cm^{-3} and is thus heavier than water. The source material was purchased already fluorescence-labelled from LLV-shop (https://www.llv-

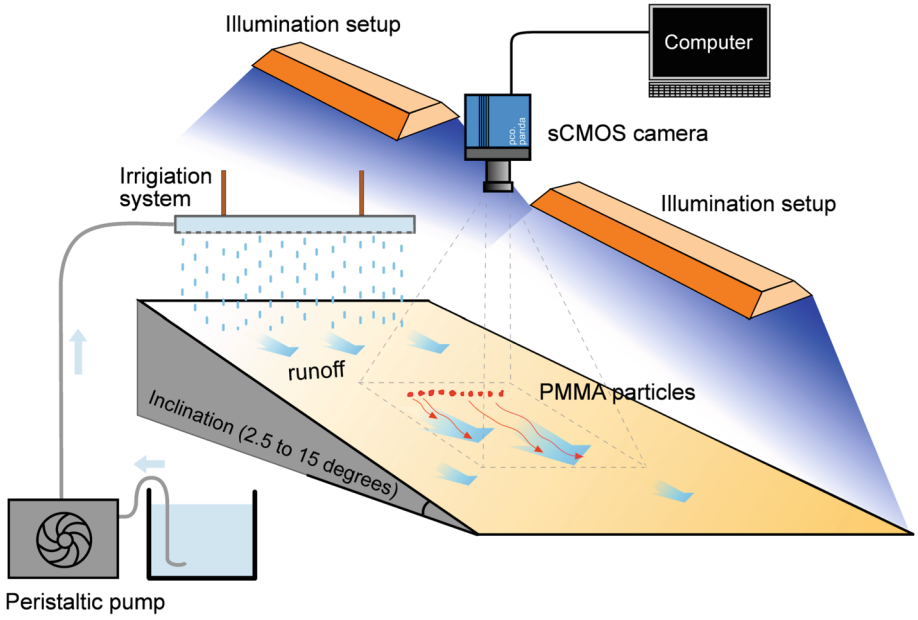


Fig. 2 Laboratory setup with the irrigation and illumination systems. The PMMA particles are positioned in row vertically below the camera

shop.de) in form of raw pellets (Art. Nr. M 13344). Subsequently, microplastic particles in the desired size range were produce by cryo-milling and sieving. We chose PMMA because among the nonfloating polymers, it has a medium density. Additionally, it can be milled relatively easily in contrast to other polymers (glass transition temperature above 100 °C). We determined the size range on a subsample by taking pictures with a stereomicroscope (Leica M50, Leica Microsystems & Olympus DP 26 camera, Olympus Corporation) and the imaging software cellSens (Olympus Corporation). The maximum length and width of 200 particles were measured in the free software ImageJ [45]. The particles were irregularly shaped with a mean particle length of 1215±227 µm (± standard deviation), a mean particle width of 919±186 µm and a mean length-to-width ratio of 1.32 (Fig. 1). The smallest recorded length and width were 842 µm and 485 µm, respectively, the largest recorded length and width were 2456 µm and 1648 µm, respectively. The sizes of particles used during the experiments were estimated via image analysis in Python (c.f. "Image analysis" section).

Experimental setup

The experimental setup consisted of an inclined rough surface of 545 × 400 mm that was irrigated at its upper edge (Fig. 2). To limit all particle movements to surface transport and avoid particle loss by infiltration, we affixed a sand surface with tile glue (Probau GmbH, Cologne, Germany) on two wooden chipboards. For one board we used medium sand (< 630 µm, mean grain size 408 µm), for the other coarse sand (630 − 2000 µm, mean grain size 1,052 µm) to create two different levels of roughness (in the following referred to as 'fine surface' and 'coarse surface', respectively). For each experimental run ten PMMA particles (once 9 and twice 11) were placed in a row spaced by roughly one cm in the center of the surface. To generate runoff on the surface, an irrigation system was built using an ISMATEC ISM930C-IPC04 peristaltic pump (Cole-Parmer GmbH, Wertheim, Germany) equipped with four channels. The irrigation was placed at the upper edge of the surface to avoid any splash effects at the center where the PMMA particles were located. Therefore, the artificial rain did not hit the microplastics directly, but generated a runoff that flowed down the plate (Supplementary Figures S1 and S2). We irrigated the surface with 4.8, 7.2 or 10.44 L h^{-1}. During the experiments, the surface was inclined by 2.5, 5, 7.5, 10, 12.5 and 15°. Altogether 36 different combinations of surface roughness (2), irrigation rates (3) and inclinations (6) were tested in three repetitions amounting to 108 runs in total of 50 seconds each.

Camera and illumination setup

During the experiments, we took pictures with an sCMOS (advanced scientific complementary metal-oxide-semiconductor) high resolution pco.panda 4.2 camera (PCO AG, Kehlheim, Germany) equipped with a monochrome sensor with 2048 × 2048 pixels of 6.5 ×

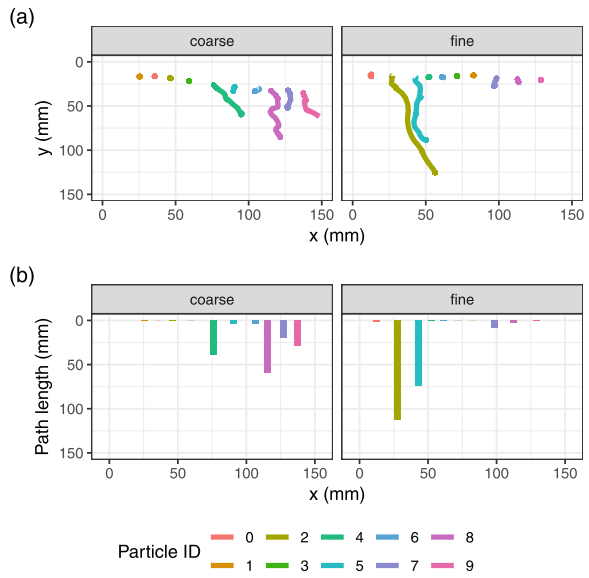


Fig. 3 Examples of (a) trajectories of particles and (b) corresponding path lengths. These are results from one of the runs on the coarse surface with inclination 2.5° and an irrigation rate of 10.44 L h^{-1} and from a run with the same parameters but on the fine surface. Particle ID corresponds to an automatic numbering of PMMA articles sorted by the *y* coordinate

6.5 µm^2 and a Ricoh FL-BC2518-9M 25 mm objective. An sCMOS sensor has the advantage to combine a large field-of-view, low electrical noise and a high frame rate compared to a CCD (charge-coupled device) sensor [46]. The exposure time of the camera was set to 100 ms, resulting in a frame rate of 10 frames per second and 500 pictures during the 50 seconds of recording for each experimental run. The camera was positioned 30 cm vertically above the PMMA particles in the central part of the surface and recorded an area of 150 × 150 mm (Supplementary Figure S2). The camera was equipped with an EFFO-FLR-BN630-M40.5 optical filter (Midwest Optical Systems Inc., Palatine, Illinois, United States) to limit the recorded spectrum to a narrow spectral range around 630 nm, which corresponds to the fluorescence of the PMMA particles. Two EFFI-FLEX-15-465-SD-P2 LED bars (EFFILUX GmbH, Les Ulis, France) with an emission wave length of 465 nm (blue light) were fixed laterally above the rough surface to excite the fluorescence of the PMMA particles. Natural illumination was minimized by darkening the laboratory to reduce noise in the pictures.

Image analysis

We analysed the images using Python 3.8 (Python Software Foundation, https://www.python.org/) and R 4.0.3 [47]. Some of the images contained artefacts due to residual ambient light which resulted in blurred particles' edges. Additionally, some particles moved abruptly and their trajectories appeared as lines in the images (Supplementary Figure S4). To remove the artefacts, we first preprocessed the images by calculating the centre of mass of every particle and then simplified them to equally sized circles centered around their centres of mass. This step increased the accuracy of particle tracking. The latter was done with TrackPy version 0.4.2 [48], a Python package that calculates particle trajectories or paths. Sub-

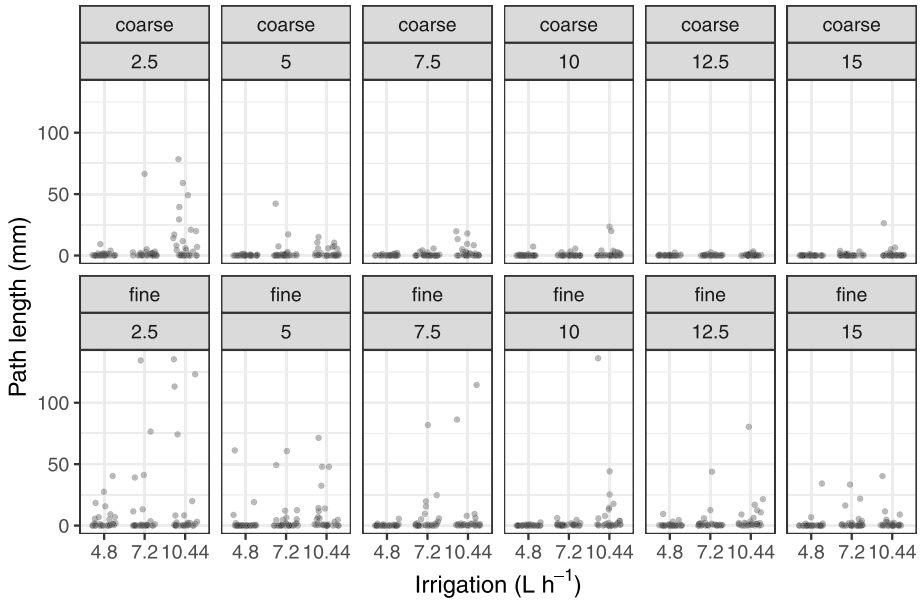


Fig. 4 Variation of path lengths with increasing irrigation and inclination on the coarse and the fine surfaces

Table 1 Estimated parameters of the linear model with confidence intervals shown in parentheses

Model parameter	Estimate (CI)
Intercept (mm)	1.76 (1.2, 2.3)
Equivalent diameter (mm μm^{-1})	-2.96E-03 (-4.49E-03, -1.54E-03)
Inclination (mm $degree^{-1}$)	-0.46 (-0.69, -0.27)
Irrigation (mm L^{-1} h)	0.89 (0.57, 1.3)
Roughness fine (mm)	3.33 (1.97, 5.11)

The parameters correspond to β_i in Eq. 1. They represent slopes for numerical variables and an additive effect for the factor roughness

sequently, we derived the path lengths by calculating the Euclidean distances between particle positions in consecutive images. The calculated centre of mass could vary slightly from image to image by one or two pixels without any particle movement. Therefore, we ignored any trajectory of less than 3 pixels.

From every first image of the experiment, we estimated the sizes of the particles by segmenting the images at the grey value 200 (i.e. any pixel with a grey value equal to or larger than 200 was classified as particle). Based on these sizes in pixels, we calculated particle characteristics 'equivalent diameter' (i.e. the diameter that a completely spherical particle with the same area visible in the image would have) and 'eccentricity' which shows whether a particle resembles a sphere or an ellipse. Eccentricity is defined as the ratio between the focal points of an ellipse to its major axis. To calculate eccentricity, the particle's shape is approximated as an ellipse with the same second-order moments (covariance) as its visible area. Both characteristics were calculated with the Python module 'measure' from the image processing library scikit-image (https://scikit-image.org/).

Statistical analysis

We estimate the influence, i.e. effect sizes and confidence intervals, of the experimental parameters 'roughness', 'inclination', 'irrigation' rate and 'equivalent diameter' on the length of paths that the particles moved in the experimental runs. We follow the advice of the American Statistical Association and refrain from hypothesis testing [49, 50]. Instead, we estimate the effect sizes and their confidence intervals using bootstrap re-sampling [51] in the following linear model:

$$y = \mathrm{Int} + \beta_{\mathrm{ED}} \cdot \mathrm{ED} + \beta_{\mathrm{Inc}} \cdot \mathrm{Inc} + \beta_{\mathrm{Irr}} \cdot \mathrm{Irr} + \beta_{\mathrm{Rough}} \cdot \mathrm{Rough} + \varepsilon \quad (1)$$

where y is the path length in mm, Int the intercept in mm, ED the equivalent diameter in µm, Inc the inclination in degrees, Irr the irrigation rate in L h^{-1}, Rough the roughness (a factor, *coarse* being the reference level), β_i are the model parameters or effects (a slope for numerical variables and an additive effect for factors) and ε are the residuals of the model. The intercept is the expected value of the path length if all other factors are zero. This is not

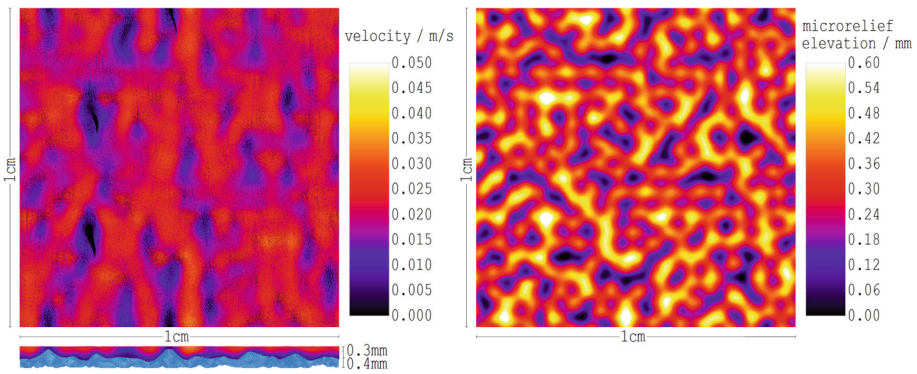


Fig. 5 Velocity profile on a 1 cm × 1 cm section of the fine-grained microrelief illustrated with colored passive tracer particles (left). The 0.3 mm average water film thickness leads to a flow rate of 7.27 L h^{-1}. The flow direction is top to bottom. The flow pattern is stationary and forms streams of increased velocity around teardrop shaped regions of decreased velocity behind bumps in the microrelief. At the bottom left we show a side view of the microrelief and water layer above. At the right we show a heatmap of the microrelief elevation

meaningful for our setting because this would extrapolate the model beyond the measured ranges of the experimental parameters. To have a meaningful intercept, we subtracted the mean from the irrigation, inclination and the equivalent diameter, i.e. we centered all experimental parameters around their respective mean values. Therefore, the intercept now equals the travelled path length in mm for a mean-sized particle (1817 µm) irrigated by the mean irrigation rate (7.5 L h^{-1}), on a mean-inclined (8.7°) coarse surface. This data normalization does not affect the estimation of the effects β_i, but of the intercept only. We note that our model aims at a phenomenological description of particle transport within a certain range of parameters around their mean values rather than a fully detailed physical prediction.

We estimated the confidence intervals by bootstrapping the data (stratified by roughness) 10000 times and re-calculated the model on theses bootstrap samples [52]. Different methods to estimate confidence intervals from bootstraping exist and we compared three of them, namely the percentile, student-*t* and bca (bias-corrected and accelerated) methods, all implemented in the `R` package `rsample` [53]. All calculations were done in `R` [47] using the package collection `tidymodels` [54], V.0.1.0.

Simulation of the water flow

To better understand the water flow on the sediment surfaces, we qualitatively examined the flow patterns in a simulation on a microscopic level of detail. We assumed a free surface flow totally wetting an inclined plate with roughness at the scale of the grains of sand used in preparation of the fine surface. The liquid layer was very thin, about 0.1 − 0.3 mm thickness.

To model the flow, we used the Volume-of-Fluid [55–57] lattice Boltzmann method [58, 59] implemented in the software *FluidX3D* [55, 60, 61]. We simulated an area of 10 mm × 10 mm at 2.5° inclination. The boundaries in the lateral directions were periodic. We used the parameters of water at 20 °C (kinematic shear viscosity $\nu = 1.004 \cdot 10^{-6}$ m^2 s^{-1}, density $\rho = 998.21$ kg m^{-3}, surface tension $\sigma = 72.75 \cdot 10^{-3}$ kg s^{-2}) [62]. We generated the microrelief of the surface caused by its roughness with a 2D periodic Perlin noise [63]. The grain size for the Perlin noise was set to approximately 0.4 mm diameter and the thickness of the water film to 0.3 mm on average. This resulted in a simulated flow rate (scaled up to the entire 400 mm width of the plate in the experiment) of 7.27 L h^{-1} which is in line with experimental parameter of 7.2 L h^{-1}.

Additionally, to visualize the flow patterns, we irrigated the surfaces without any PMMA particles with a methanol solution stained with Nile Red, a fluorescent dye [64, 65]. Using Nile Red allowed us to capture the flow patterns with the same LED lights and camera setup and filter as in the experimental runs with particles. We are aware that methanol has different wetting properties than water. Therefore, we use the videos for visualisation purposes only. We recorded the initial wetting process as well as the flow patterns once stationary flow set in. The coarse surface was irrigated at an inclination of 7.5° with an irrigation rate of 7.2 L h^{-1} and the fine surface at an inclination of 2.5° with an irrigation rate of 7.2 L h^{-1}. The videos are provided in the Additional files 2 and 3.

Results and discussion

Tracking microplastics under laboratory conditions

Our experiments allowed for a high-resolution tracking of microplastic particles under different inclination, surface roughness and irrigation scenarios. Next to their respective trajectories, the image analysis provided information about the characteristics of the particles in the experimental runs. The equivalent diameter of PMMA particles ranged between 453 and 3387 µm (Supplementary Figure S3). They differed from spheres with eccentricities varying between 0.1 and 0.9. An eccentricity of 0 corresponds to a perfect circle, a value of 1 indicates an "extreme ellipse", a parabola. Thus, the larger the value the less circular the PMMA particle. The distribution of both characteristics were comparable between different roughnesses and irrigation rates and shows that the particles in the experiments were comparable (Supplementary Figure S3). Therefore, no differences between the treatments as a consequence of a systematically varying particle morphology are to be expected.

576 out of 1081 particles remained motionless and the other 505 recorded particle trajectories were mostly short (Fig. 3). Most of the recording time, particles did not move at all. However, from time to time they travelled a larger distance, often very quickly. Thus, the overall path lengths and the particle's maximum path length per second correlated (Spearman correlation coefficient of 0.9). The Supplementary Figure S5 shows this relationship. The maximum overall recorded path length equalled 136 mm (fine surface, inclination 10°, irrigation rate 10.44 L h^{-1}) and the maximum recorded path length in one second 135 mm (fine surface, inclination 2.5°, irrigation rate 7.2 L h^{-1}).

Influence of experimental parameters on path lengths

The observed path lengths increased with increasing irrigation rate and were larger on the fine than on the coarse surface (Fig. 4). However, the dot plots show that the most frequent paths were small or zero and that only few particles traveled far.

The estimated parameters of the linear model support this observation (Table 1). Confidence intervals estimated with different bootstrap methods were comparable

(Supplementary Figure S6) and we report the student-t intervals only. All confidence intervals excluded zero. With increasing inclination and increasing equivalent diameter of the microplastic particles, the path length decreased. Indeed, for a constant flow rate, the water film thickness is approximately proportional to the $(\sin(\alpha))^{-1/3}$, with α being the inclination [66][p.183 eq.(4.2.13)] and the larger the film thickness the larger the flow rate (see also "Water flow" section). The effects of inclination and equivalent diameter seem smaller compared to the irrigation rate or the roughness. However, the experimental parameters are measured on different scales and the absolute values of the effects depend on it. Thus, it is better to compare the effects relative to the measurement units of the parameters. If, for example, the inclination is increased from 5 to 10 degrees, the path length would decrease by 2.3 mm. For an increase of irrigation rate from 4 L h^{-1} to 8 L h^{-1}, the path length would increase by 3.6 mm. By increasing the equivalent diameter of the microplastic particles from 1000 to 2000 µm, the path length decreases by 3 mm. And finally, on the fine surface, the particles travelled 3.33 mm farther which is in accordance with several observations from laboratory and environmental studies (e.g. in fluvial context and at valley slopes [67, 68] or on beaches [69, 70]).

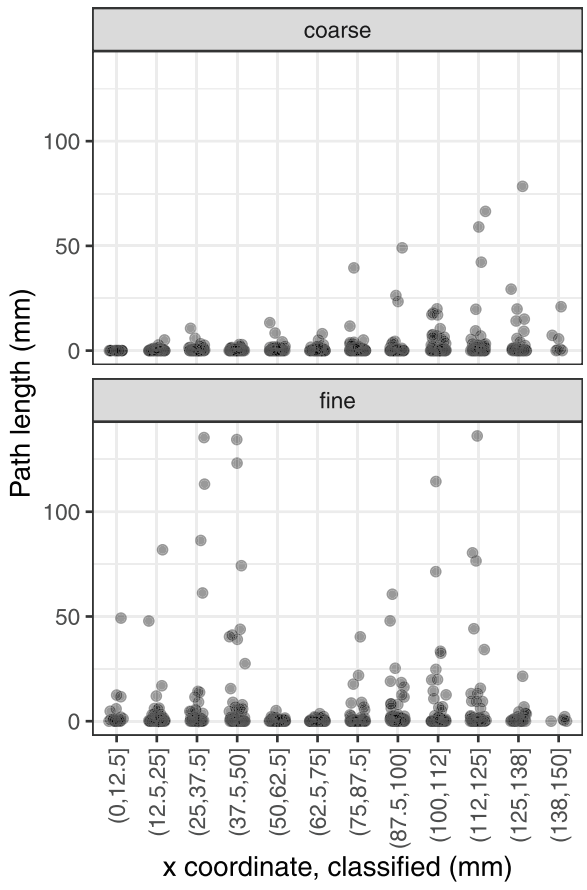


Fig. 6 Variation of path lengths with starting position of the particles on the x axis. The x coordinate of the starting point was split in classes of 12.5 mm length to better show the distribution of path lengths

The overall goodness of fit of the model was low ($R^2_{\text{adj}} = 0.07$) indicating that the assumption of the linear relationship between the experimental parameters and the path length is overly simplistic. However, because our intention is not to predict path lengths but to analyse the influence of experimental parameters (size effect and confidence intervals), the model is still useful.

Water flow

To better understand why most particles are barely advected whilst a few particles are moved across a long distance (Fig. 4), we qualitatively examined the flow pattern in a simulation. For the typical flow rates used in the experiment, the water film thickness is on the same scale as the grain size of the roughness-induced microrelief. The microrelief therefore greatly influences the flow pattern on a microscopic level. For the setup simulated here, where we target 7.2 L h^{-1} to match the experiment, we find that the film thickness is 0.3 mm on average.

For comparison, for the targeted irrigation rates of 4.8 and 10.4 L h^{-1}, we determine the film thickness to 0.252 mm and 0.348 mm respectively. The resulting simulation flow rates are 4.87 and 10.97 L h^{-1}. The flow rate is proportional to film thickness to the third power for a laminar flow over a plane and smooth surface [66][p.183 eq.(4.2.12)]. With increasing inclination, the film thickness decreases. If the film is thinner than the diameter of the microplastic particles, the water surface wetting the particles bulges up on them and pushes them down onto the microrelief, increasing friction and reducing mobility. In contrast, at large film thickness, microplastic particles may be entirely engulfed into the liquid and not touch the microrelief at all, reducing friction and increasing mobility. So although the flow velocity at the surface is faster (proportional to $(\sin(\alpha))^{1/3}$, with α being the inclination) at larger inclination, the friction between the particles and the plate is increased and particles are less mobile overall. The roughness changes the flow conditions and influences the flow rate. Therefore, this relationship holds only approximately.

Figure 5 shows the emerging flow pattern after 0.1 s of simulation, illustrated with colored passive tracer point-particles. These are passively advected points without influence on the flow itself. They are illustrated as colored pixels to show the magnitude of local velocity.

Although the microrelief has no directionality in itself, the flow pattern shows distinctive channels with increased flow velocity on the scale of the microrelief, while in teardrop-shaped regions behind larger bumps of the microrelief, flow velocity is significantly reduced. This pattern is stationary. The asymmetry of the flow pattern indicates deviations from purely laminar flow. There are regions where particles will move down the plate with increased probability (fast flow channels) while in other regions the hydrodynamic force can never overcome friction.

At the low flow rates in the experiment, there are only little surface instabilities (non-stationary, bow-shaped, capillary waves) present. These mainly occur at higher flow rate and larger film thickness [71–77].

On top of the microrelief, the surfaces also possess a macrorelief. The latter emerged during construction of the plates when sand was glued to the chipboards. In regions where there is more glue, the sand layer may be thicker, and in other regions the sand layer may be thinner. This macrorelief also creates preferential flow channels (on a larger scale) where PMMA particles are transported particularly far. Figure 6 shows that on the coarse surface, particles starting on the right moved the farthest, while on the fine surface, preferential flow existed on both the left and right sides.

These macroscopic flow channels would be much more difficult to model in a simulation, since on the one hand both micro- and macrorelief would have to be simultaneously resolved on very different length scales and on the other hand the exact topography is of key importance because it directly affects experimental results. So to confirm those preferential macroscopic flow channels in the experimental setup, we used the recordings of methanol solution stained with Nile Red. Figure 7 confirms the preferential flow paths on the macro scale on the fine surface. The imaged part in the videos is slightly shifted compared to the imaged part in the experiments because the experimental setup had to be moved under the fume hood. The videos are provided in the Supporting Information Additional files 2 and 3.

In the video of the fine surface, we also observed some surface ripples, mainly along the preferential flow channels in the macrorelief. These may be caused by the locally increased flow rate, but could also be a result of the irrigation system perturbing the surface. Such surface ripples could temporarily increase the hydrodynamic force enough to unlock a microplastic particle from the microrelief and initiate the motion.

Transport processes and environmental implications

In our experiments, we observe that the transport of the PMMA particles is based on two different phenomena: transport-inhibiting interaction with the microrelief and transport-enhancing interaction with preferential flow channels of the macrorelief. In the first case flow patterns with areas of larger flow velocities and therefore larger hydrodynamic forces acting on the particles emerge due to the microrelief. The size of PMMA particles is at a similar scale as the microrelief sand grains and generally larger than the water film. This means that the particles on the one hand are able to interlock with the microrelief and are not transported, and on the other hand have constant friction if rolled down the plate by the flow, increasing the probability to lock into the microrelief again and get stuck after a short travel distance. Therefore, most particles did not move at all or travelled only a very short distance. Second, the macrorelief that overlays the microrelief provides macroscopic preferential flow channels for the particles. If a particle is located in or reaches a preferential flow channel, the travelled path lengths are large.

Under natural conditions, the fast flow pathways (both micro and macro) will potentially lead to increased erosion, carving channels into the soil surface [78, 79]. Gomez et al. [78], for example, observed a larger erosion on rougher surfaces and attributed this to "concentration of flow around roughness elements". This is comparable to the flow patterns around rough surface elements that we obtained in the simulation on the microscale and the observed preferential flow paths on the macroscale. This kind of 'flow concentration' could finally lead to rill erosion [80]. Additionally, under natural conditions, interrill erosion due to the detachment of particles by raindrop impact and transport to rills (i.e. preferential pathways), an interplay between overland flow, water infiltration and surface sealing could be important processes affecting particle transport. At least in the beginning, interrill erosion could be size-selective and small particles are eroded preferentially [81, 82]. However, the exact distribution of eroded particles for a particular soil depends on different soil properties like particle size distribution or aggregate stability [83, 84]. The general dependence of soil erodibility is reflected in the well known Revised Universal Soil Loss Equation (RUSLE) [85]. It is still unclear whether erosion processes would affect microplastic particles in the same way as natural sediments because of the broad range of shapes, densities and sizes of microplastics [7, 8].

In the experiments and the simulation investigated here, erosion and infiltration are explicitly prevented. Furthermore it should be considered that the irregular PMMA particles of a mean particle length of 1215 µm that we used to conduct our experiments are only one exemplary polymer and microplastics shape. Because microplastics comprise a large range of different particle sizes, densities, shapes and degradation stages, our results can reflect a small part of the reality only [7, 8]. Particles of different shapes (e.g. spheres, fibres or films) and with different

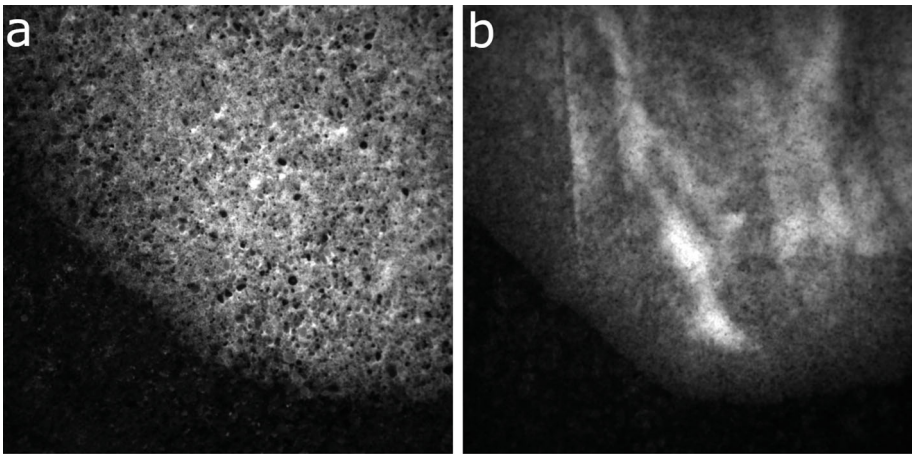


Fig. 7 Runoff stained with Nile red. a: coarse surface irrigated at 7.5° inclination with 7.2 L h^{-1}; b: fine surface irrigated at 2.5° inclination with 7.2 L h^{-1}. Preferential flow paths are visible on the fine surface

surface properties (e.g. due to degradation or biofilm) might interact differently with rough surfaces and be (re)mobilized and transported in a different way.

Nevertheless, the results of our laboratory setup contribute to understanding the horizontal transport of microplastics in nature. Sandy sediment and soil surfaces with sparse vegetation cover resemble to a certain extent to our surfaces. Comparable environments can be found for example in floodplains and on riverbanks [86, 87], on agricultural fields, in clearing areas, in freshly developed building sites and in ruderal habitats [88]. Because rivers are important pathways of microplastics [13, 89], the adjacent riparian zones and floodplains act as an interface between aquatic and terrestrial ecosystems. Therefore, they play an important role in the distribution of microplastics within the "plastic cycle", a term coined by Horton et al. [20] to emphasize the connectivity between environmental compartments. During runoff events caused by floods or precipitation excess, microplastics can be (re-)mobilised, transported and accumulated in these highly dynamic areas [44, 87, 90].

The transport parameters of microplastics under natural conditions such as the flow velocities necessary to move the particles, the influence of land cover (e.g. vegetation) and sedimentation rates [90] are still not deciphered. While Nizzetto and colleagues [31] suggested that microplastics moved similarly to natural particles and organic matter, there is still a significant lack of data to understand the transport patterns of microplastics "across the compartments of hydrological catchments" [44]. Analysing transport patterns under laboratory conditions may facilitate future studies on microplastic distribution and mobility. For future research additional important factors, e.g. topography, hydrology, land cover and the exposure of microplastics to physical, biological and chemical processes should be considered [44] in order to improve the understanding of microplastic transport and the resulting contamination of the environment.

Conclusions

We developed a laboratory setup to reliably trace fluorescent microplastic particles in real-time during irrigation experiments. By limiting the experimental variables to irrigation rate, inclination and surface roughness the driving factors of the particle movement could be deciphered. Especially the roughness and the irrigation rate turned out to be important. In our experiments, we could show that the transport of the microplastics was inhibited by the interaction of the microplastics with the microrelief and enhanced by preferential flow channels of the macrorelief of the rough surfaces. The computer simulated flow patterns showed variable flow velocities on the scale of the microrelief and thus spatially variable hydrodynamic forces acting on the particles. Our laboratory results are a first step to gain a better understanding of the horizontal transport of microplastics on natural sediment and soil surfaces. However, microplastics are a diverse group of contaminants with varying shapes, densities and sizes. This continuity of properties, alteration of the microplastic particles' surface properties as a consequence of biological and chemical processes in the environment and additional factors influencing soil surface, e.g. topography or land cover, could not be taken into account here and need to be part of further research.

Abbreviations

PMMA: Polymethyl methacrylate; sCMOS: Scientific complementary metal–oxide–semiconductor; CCD: Charge-coupled device

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s43591-021-00010-2.

Additional file 1: A pdf file with Supplementary Figures referenced in the main text.

Additional file 2: A video of runoff stained with Nile Red on the coarse surface. The surface was inclined by 7.5 degrees and irrigated with 7.2 L h^{-1}.

Additional file 3: A video of runoff stained with Nile Red on the fine surface. The surface was inclined by 2.5 degrees and irrigated with 7.2 L h^{-1}.

Acknowledgements
The authors thank Julia Horn for support in the laboratory and Florian Steininger for technical assistance.

Authors' contributions
CB, HL and MK designed the study, HL supervised the experimental work, ML simulated the water flow, MK did the irrigation experiments, MGJL prepared and characterized the PMMA particles, CB wrote the code and analysed the image data, all authors discussed the results, HL, CB and ML wrote the first draft of the manuscript. All authors read and approved the final manuscript.

Funding
This project was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), Project Number 391977956, SFB 1357, subprojects B04 and B06. We further acknowledge support through the computational resources provided by the Bavarian Polymer Institute. Open Access funding enabled and organized by Projekt DEAL.

Availability of data and materials
The datasets and code used and/or analysed during the current study are published on Zenodo, https://doi.org/10.5281/zenodo.4911185.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Ecosystem Research Group, Institute of Geography, Faculty of Mathematics and Natural Sciences, University of Cologne, Albertus Magnus Platz, 50923 Cologne, Germany. [2]Biofluid Simulation and Modeling – Theorethische Physik VI, University of Bayreuth, 95448 Bayreuth, Germany. [3]Animal Ecology I, BayCEER, University of Bayreuth, 95440 Bayreuth, Germany.

Received: 4 March 2021 Accepted: 7 June 2021
Published online: 02 July 2021

References

1. Geyer R, Jambeck JR, Law KL. Production, use, and fate of all plastics ever made. Sci Adv. 2017;3(7):1700782. https://doi.org/10.1126/sciadv.1700782.
2. PlasticsEurope. Plastics – the Facts 2020. Technical report. 2020. https://www.plasticseurope.org/en/resources/publications/4312-plastics-facts-2020. Accessed 25 June 2021.
3. Thompson RC, Olsen Y, Mitchell RP, Davis A, Rowland SJ, John AWG, McGonigle D, Russell AE. Lost at Sea: Where Is All the Plastic? Science. 2004;304(5672):838. https://doi.org/10.1126/science.1094559.
4. Arthur C, Baker J, Bamford H. Proceedings of the International Research Workshop on the Occurrence, Effects and Fate of Microplastic Marine Debris. Sept 9-11, 2008. NOAA Technical Memorandum NOS-OR&R-30. 2009.
5. Kataoka T, Nihei Y, Kudou K, Hinata H. Assessment of the sources and inflow processes of microplastics in the river environments of Japan. Environ Pollut. 2019;244:958–65. https://doi.org/10.1016/j.envpol.2018.10.111.
6. Meides N, Menzel T, Poetzschner B, Löder MGJ, Mansfeld U, Strohriegl P, Senker J. Reconstructing the Environmental Degradation of Polystyrene by Accelerated Weathering. Environ Sci Technol. 2021;55(12):7930–7938. https://doi.org/10.1021/acs.est.0c07718.
7. Rochman CM, Brookson C, Bikker J, Djuric N, Earn A, Bucci K, Athey S, Huntington A, McIlwraith H, Munno K, Frond HD, Kolomijeca A, Erdle L, Grbic J, Bayoumi M, Borrelle SB, Wu T, Santoro S, Werbowski LM, Zhu X, Giles RK, Hamilton BM, Thaysen C, Kaura A, Klasios N, Ead L, Kim J, Sherlock C, Ho A, Hung C. Rethinking microplastics as a diverse contaminant suite. Environ Toxicol Chem. 2019;38(4):703–11. https://doi.org/10.1002/etc.4371.
8. Kooi M, Koelmans AA. Simplifying Microplastic via Continuous Probability Distributions for Size, Shape, and Density. Environ Sci Technol Lett. 2019;6(9):551–7. https://doi.org/10.1021/acs.estlett.9b00379.
9. Waller CL, Griffiths HJ, Waluda CM, Thorpe SE, Loaiza I, Moreno B, Pacherres CO, Hughes KA. Microplastics in the Antarctic marine system: An emerging area of research. 2017;598:220–7. https://doi.org/10.1016/j.scitotenv.2017.03.283.
10. Horton AA, Walton A, Spurgeon DJ, Lahive E, Svendsen C. Microplastics in freshwater and terrestrial environments: Evaluating the current understanding to identify the knowledge gaps and future research priorities. Sci Total Environ. 2017;586:127–41. https://doi.org/10.1016/j.scitotenv.2017.01.190.
11. Bergmann M, Mützel S, Primpke S, Tekman MB, Trachsel J, Gerdts G. White and wonderful? Microplastics prevail in snow from the Alps to the Arctic. Sci Adv. 2019;5(8):1157. https://doi.org/10.1126/sciadv.aax1157.
12. Hurley RR, Nizzetto L. Fate and occurrence of micro(nano)plastics in soils: Knowledge gaps and possible risks. Curr Opin Environ Sci Health. 2018;1:6–11. https://doi.org/10.1016/j.coesh.2017.10.006.
13. Rochman CM. Microplastics research–from sink to source. Science. 2018;360(6384):28–9. https://doi.org/10.1126/science.aar7734.
14. Rillig MC, Lehmann A. Microplastic in terrestrial ecosystems. Science. 2020;368(6498):1430–1. https://doi.org/10.1126/science.abb5979.
15. Löder MGJ, Gerdts G. Methodology Used for the Detection and Identification of Microplastics–A Critical Appraisal. In: Bergmann M, Gutow L, Klages M, editors. Marine Anthropogenic Litter. Cham: Springer International Publishing; 2015. p. 201–27.
16. Bläsing M, Amelung W. Plastics in soil: Analytical methods and possible sources. Sci Total Environ. 2018;612:422–35. https://doi.org/10.1016/j.scitotenv.2017.08.086.
17. Piehl S, Leibner A, Löder MGJ, Dris R, Bogner C, Laforsch C. Identification and quantification of macro- and microplastics on an agricultural farmland. Sci Rep. 2018;8(1):17950. https://doi.org/10.1038/s41598-018-36172-y.
18. Möller JN, Löder MGJ, Laforsch C. Finding Microplastics in Soils: A Review of Analytical Methods. Environ Sci Technol. 2020;54(4):2078–90. https://doi.org/10.1021/acs.est.9b04618.
19. Möller JN, Heisel I, Satzger A, Vizsolyi EC, Oster J, Agarwal S, Laforsch C, Löder MGJ. Tackling the Challenge of Extracting Microplastics from Soils: Protocol to Purify Soil Samples for Spectroscopic Analysis. Environ Toxicol Chem. 2021. https://doi.org/10.1002/etc.5024.
20. Horton AA, Dixon SJ. Microplastics: An introduction to environmental transport processes. Wiley Interdiscip Rev: Water. 2018;5(2):1268. https://doi.org/10.1002/wat2.1268.
21. Weithmann N, Möller JN, Löder MGJ, Piehl S, Laforsch C, Freitag R. Organic fertilizer as a vehicle for the entry of microplastic into the environment. Sci Adv. 2018;4(4):8060. https://doi.org/10.1126/sciadv.aap8060.
22. Allen S, Allen D, Phoenix VR, Le Roux G, Durántez Jiménez P, Simonneau A, Binet S, Galop D. Atmospheric transport and deposition of microplastics in a remote mountain catchment. Nat Geosci. 2019;12(5):339–44. https://doi.org/10.1038/s41561-019-0335-5.
23. Laforsch C, Ramsperger AFRM, Mondellini S, Galloway TS. Microplastics: A Novel Suite of Environmental Contaminants but Present for Decades. In: Reichl F-X, Schwenk M, editors. Regulatory Toxicology. Berlin, Heidelberg: Springer; 2021. p. 1–26.
24. Rillig MC. Microplastic in Terrestrial Ecosystems and the Soil? Environ Sci Technol. 2012;46(12):6453–4. https://doi.org/10.1021/es302011r.
25. Van Cauwenberghe L, Devriese L, Galgani F, Robbens J, Janssen CR. Microplastics in sediments: A review of techniques, occurrence and effects. Mar Environ Res. 2015;111:5–17. https://doi.org/10.1016/j.marenvres.2015.06.007.

26. Waldschläger K, Schüttrumpf H. Infiltration Behavior of Microplastic Particles with Different Densities, Sizes, and Shapes–From Glass Spheres to Natural Sediments. Environ Sci Technol. 2020;54(15):9366–73. https://doi.org/10.1021/acs.est.0c01722.
27. Yu M, van der Ploeg M, Lwanga EH, Yang X, Zhang S, Ma X, Ritsema CJ, Geissen V. Leaching of microplastics by preferential flow in earthworm (Lumbricus terrestris) burrows. Environ Chem. 2019;16(1):31. https://doi.org/10.1071/EN18161.
28. Keller AS, Jimenez-Martinez J, Mitrano DM. Transport of Nano- and Microplastic through Unsaturated Porous Media from Sewage Sludge Application. Environ Sci Technol. 2020;54(2):911–20. https://doi.org/10.1021/acs.est.9b06483.
29. Huerta Lwanga E, Gertsen H, Gooren H, Peters P, Salánki T, van der Ploeg M, Besseling E, Koelmans AA, Geissen V. Incorporation of microplastics from litter into burrows of Lumbricus terrestris. Environ Pollut. 2017;220:523–31. https://doi.org/10.1016/j.envpol.2016.09.096.
30. Rillig MC, Ziersch L, Hempel S. Microplastic transport in soil by earthworms. Sci Rep. 2017;7(1):1362. https://doi.org/10.1038/s41598-017-01594-7.
31. Nizzetto L, Bussi G, Futter MN, Butterfield D, Whitehead PG. A theoretical assessment of microplastic transport in river catchments and their retention by soils and river sediments. Environ Sci Pollut Impacts. 2016;18(8):1050–9. https://doi.org/10.1039/C6EM00206D.
32. Black KS, Athey S, Wilson P, Evans D. The use of particle tracking in sediment transport studies: A review. Geol Soc Lond Spec Publ. 2007;274(1):73–91. https://doi.org/10.1144/GSL.SP.2007.274.01.09.
33. White TE. Status of measurement techniques for coastal sediment transport. Coast Eng. 1998;35(1-2):17–45.
34. Liedermann M, Tritthart M, Habersack H. Particle path characteristics at the large gravel-bed river Danube: Results from a tracer study and numerical modelling. Earth Surf Process Landforms. 2013;38:512–522. https://doi.org/10.1002/esp.3338.
35. McComb P, Black K. Detailed Observations of Littoral Transport Using Artificial Sediment Tracer, in a High-Energy, Rocky Reef and Iron Sand Environment. J Coast Res. 2005;212:358–73. https://doi.org/10.2112/03-574.1.
36. Vila-Concejo A, Ferreira Ó, Ciavola P, Matias A, Dias JMA. Tracer studies on the updrift margin of a complex inlet system. Mar Geol. 2004;208(1):43–72. https://doi.org/10.1016/j.margeo.2004.04.020.
37. Katsuragi H. Length and time scales of a liquid drop impact and penetration into a granular layer. J Fluid Mech. 2011;675:552–73. https://doi.org/10.1017/jfm.2011.31.
38. Long EJ, Hargrave GK, Cooper JR, Kitchener BGB, Parsons AJ, Hewett CJM, Wainwright J. Experimental investigation into the impact of a liquid droplet onto a granular bed using three-dimensional, time-resolved, particle tracking. Phys Rev E. 2014;89(3):032201. https://doi.org/10.1103/PhysRevE.89.032201.
39. Hardy RA, Pates JM, Quinton JN, Coogan MP. A novel fluorescent tracer for real-time tracing of clay transport over soil surfaces. CATENA. 2016;141:39–45. https://doi.org/10.1016/j.catena.2016.02.011.
40. Hardy RA, James MR, Pates JM, Quinton JN. Using real time particle tracking to understand soil particle movements during rainfall events. CATENA. 2017;150:32–8. https://doi.org/10.1016/j.catena.2016.11.005.
41. Teleki PG. Fluorescent sand tracers. J Sed Res. 1966;36(2):468–85. https://doi.org/10.1306/74D714EC-2B21-11D7-8648000102C1865D.
42. Rinnan R, Rinnan Å. Application of near infrared reflectance (NIR) and fluorescence spectroscopy to analysis of microbiological and chemical properties of arctic soil. Soil Biol Biochem. 2007;39(7):1664–73. https://doi.org/10.1016/j.soilbio.2007.01.022.
43. Hardy RA, Quinton JN, James MR, Fiener P, Pates JM. High precision tracing of soil and sediment movement using fluorescent tracers at hillslope scale. Earth Surf Process Landf. 2019;44(5):1091–9. https://doi.org/10.1002/esp.4557.
44. Windsor FM, Durance I, Horton AA, Thompson RC, Tyler CR, Ormerod SJ. A catchment-scale perspective of plastic pollution. Glob Chang Biol. 2019;25(4):1207–21. https://doi.org/10.1111/gcb.14572.
45. Schneider CA, Rasband WS, Eliceiri KW. NIH Image to ImageJ: 25 years of image analysis. Nat Methods. 2012;9(7):671–5. https://doi.org/10.1038/nmeth.2089.
46. Saurabh S, Maji S, Bruchez MP. Evaluation of sCMOS cameras for detection and localization of single Cy5 molecules. Opt Express. 2012;20(7):7338–49. https://doi.org/10.1364/OE.20.007338.
47. R Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing; 2020. https://www.R-project.org/.
48. Allan DB, Caswell T, Keim NC, van der Wel CM. soft-matter/trackpy: Trackpy v0.4.2 (Version v0.4.2). Zenodo. 2019. http://doi.org/10.5281/zenodo.3492186.
49. Wasserstein RL, Lazar NA. The ASA Statement on p-values: Context, process, and purpose. Am Stat. 2016;70(2):129–33. https://doi.org/10.1080/00031305.2016.1154108.
50. Wasserstein RL, Schirm AL, Lazar NA. Moving to a World Beyond "p < 0.05". Am Stat. 2019;73(sup1):1–19. https://doi.org/10.1080/00031305.2019.1583913.
51. Efron B, Tibshirani RJ. An Introduction to the Bootstrap: CRC press; 1994.
52. Davison AC, Hinkley DV. Bootstrap Methods and Their Application, Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press; 1997. https://doi.org/10.1017/CBO9780511802843.
53. Kuhn M, Chow F, Wickham H. Rsample: General Resampling Infrastructure. 2020. R package version 0.0.7. https://CRAN.R-project.org/package=rsample.
54. Kuhn M, Wickham H. Tidymodels: a Collection of Packages for Modeling and Machine Learning Using Tidyverse Principles. 2020. https://www.tidymodels.org.
55. Lehmann M. High Performance Free Surface LBM on GPUs. 2019. Master thesis, Biofluid Simulation and Modeling - Theorethische Physik VI, University of Bayreuth. Germany. https://doi.org/10.15495/EPub_UBT_00005400.
56. Körner C, Thies M, Hofmann T, Thürey N, Rüde U. Lattice Boltzmann model for free surface flow for modeling foaming. J Stat Phys. 2005;121(1-2):179–96.
57. Janßen C, Krafczyk M. Free surface flow simulations on GPGPUs using the LBM. Comput Math Appl. 2011;61(12):3549–63.
58. Krüger T, Kusumaatmaja H, Kuzmin A, Shardt O, Silva G, Viggen EM. The lattice Boltzmann method. Springer Int Publ. 2017;10:978–3.
59. Chapman S, Cowling TG, Burnett D. The Mathematical Theory of Non-uniform Gases: an Account of the Kinetic Theory of Viscosity, Thermal Conduction and Diffusion in Gases. Cambridge, United Kingdom: Cambridge university press; 1990.
60. Häusl F. MPI-based multi-GPU extension of the Lattice Boltzmann Method. 2019. Bachelor thesis, Biofluid Simulation and Modeling - Theorethische Physik VI, University of Bayreuth. Germany.
61. Lehmann M, Gekle S. Analytic solution to the piecewise linear interface construction problem and its application in curvature calculation for volume-of-fluid simulation codes. arXiv preprint arXiv:2006.12838. 2020. https://arxiv.org/abs/2006.12838.
62. Edge E. Water – Density Viscosity Specific Weight. https://www.engineersedge.com/physics/water__density_viscosity_specific_weight_13146.htm. Accessed 12 Feb 2021.
63. Gustavson S. Perlin Noise. https://github.com/stegu/perlin-noise. Accessed 12 Feb 2021.
64. Erni-Cassola G, Gibson MI, Thompson RC, Christie-Oleza JA. Lost, but Found with Nile Red: A Novel Method for Detecting and Quantifying Small Microplastics (1 mm to 20 μm) in Environmental Samples. Environ Sci Technol. 2017;51(23):13641–8. https://doi.org/10.1021/acs.est.7b04512.
65. Konde S, Ornik J, Prume JA, Taiber J, Koch M. Exploring the potential of photoluminescence spectroscopy in combination with Nile Red staining for microplastic detection. Mar Pollut Bull. 2020;159:111475. https://doi.org/10.1016/j.marpolbul.2020.111475.
66. Batchelor CK, Batchelor G. An Introduction to Fluid Dynamics. Cambridge: Cambridge university press; 2000.
67. Hassan MA, Reid I. The influence of microform bed roughness elements on flow and sediment transport in gravel bed rivers. Earth Surf Process Landf. 1990;15(8):739–50. https://doi.org/10.1002/esp.3290150807.
68. Chaplot V, Poesen J. Sediment, soil organic carbon and runoff delivery at various spatial scales. CATENA. 2012;88(1):46–56. https://doi.org/10.1016/j.catena.2011.09.004.

69. Bauer BO, Davidson-Arnott RGD, Hesp PA, Namikas SL, Ollerhead J, Walker IJ. Aeolian sediment transport on a beach: Surface moisture, wind fetch, and mean transport. Geomorphology. 2009;105(1-2):106–16,. https://doi.org/10.1016/j.geomorph.2008.02.016.
70. Otvos EG. Rain-Induced Beach Processes; Landforms of Ground Water Sapping and Surface Runoff. J Coast Res. 1999;15(4):16.
71. Liu J, Paul JD, Gollub JP. Measurements of the primary instabilities of film flows. J Fluid Mech. 1993;250:69–101.
72. Ramaswamy B, Chippada S, Joo S. A full-scale numerical study of interfacial instabilities in thin-film flows. J Fluid Mech. 1996;325:163–94.
73. Gjevik B. Occurrence of finite-amplitude surface waves on falling liquid films. Phys Fluids. 1970;13(8):1918–25.
74. Lu C, Jiang S-Y, Duan R-Q. Wave characteristics of falling film on inclination plate at moderate reynolds number. Sci Technol Nucl Installations. 2016;2016:7. Article ID 6586097. https://doi.org/10.1155/2016/6586097.
75. Roy R, Jain S. A study of thin water film flow down an inclined plate without and with countercurrent air flow. Exp Fluids. 1989;7(5):318–28.
76. Yu Y, Cheng X. Experimental study of water film flow on large vertical and inclined flat plate. Prog Nucl Energy. 2014;77:176–86.
77. Ramadurgam S, Chakravarthy R, Tomar G, Govindarajan R. Stability of developing film flow down an inclined surface. Phys Fluids. 2012;24(10):102109.
78. Gómez J, Nearing M. Runoff and sediment losses from rough and smooth soil surfaces in a laboratory experiment. CATENA. 2005;59(3):253–66.
79. Zhao L, Huang C, Wu F. Effect of microrelief on water erosion and their changes during rainfall. Earth Surf Process Landf. 2016;41(5):579–86. https://doi.org/10.1002/esp.3844.
80. Gilley JE. EROSION | Water–Induced. In: Hillel D, editor. Encyclopedia of Soils in the Environment. Oxford: Elsevier; 2005. p. 463–9. https://doi.org/10.1016/B0-12-348530-4/00262-9.
81. Miller WP, Baharuddin MK. Particle Size of Interrill-eroded Sediments from Highly Weathered Soils. Soil Sci Soc Am J. 1987;51(6):1610–5. https://doi.org/10.2136/sssaj1987.03615995005100060037x.
82. Asadi H, Moussavi A, Ghadiri H, Rose CW. Flow-driven soil erosion processes and the size selectivity of sediment. J Hydrol. 2011;406(1):73–81. https://doi.org/10.1016/j.jhydrol.2011.06.010.
83. Wang L, Shi ZH. Size Selectivity of Eroded Sediment Associated with Soil Texture on Steep Slopes. Soil Sci Soc Am J. 2015;79:917–929. https://doi.org/10.2136/sssaj2014.10.0415.
84. Asadi H, Ghadiri H, Rose CW, Rouhipour H. Interrill soil erosion processes and their interaction on low slopes. Earth Surf Process Landf. 2007;32(5):711–24. https://doi.org/10.1002/esp.1426.
85. Renard KG. Predicting Soil Erosion by Water: A Guide to Conservation Planning with the Revised Universal Soil Loss Equation (RUSLE): U.S. Department of Agriculture, Agricultural Research Service Agriculture Hadnbook Number 703; 1997.
86. Scheurer M, Bigalke M. Microplastics in Swiss Floodplain Soils. Environ Sci Technol. 2018;52(6):3591–8. https://doi.org/10.1021/acs.est.7b06003.
87. Weber CJ, Opp C. Spatial patterns of mesoplastics and coarse microplastics in floodplain soils as resulting from land use and fluvial processes. Environ Pollut. 2020;267:115390. https://doi.org/10.1016/j.envpol.2020.115390.
88. Fulazzaky MA, Khamidun MH, Yusof B. Sediment traps from synthetic construction site stormwater runoff by grassed filter strip. J Hydrol. 2013;502:53–61. https://doi.org/10.1016/j.jhydrol.2013.08.019.
89. Lebreton LCM, van der Zwet J, Damsteeg J-W, Slat B, Andrady A, Reisser J. River plastic emissions to the world's oceans. Nat Commun. 2017;8(1):15611. https://doi.org/10.1038/ncomms15611.
90. Lechthaler S, Esser V, Schüttrumpf H, Stauch G. Why analysing microplastics in floodplains matters: Application in a sedimentary context. Environ Sci Pollut Impacts. 2021;23(1):117–31. https://doi.org/10.1039/D0EM00431F.

Publisher's Note

Tracing the horizontal transport of microplastics on rough surfaces

Supporting Information

Hannes Laermanns[a], Moritz Lehmann[b], Marcel Klee[a], Martin G.J. Löder[c], Stephan Gekle[b], and Christina Bogner[a]

[a]Ecosystem Research Group, Institute of Geography, Faculty of Mathematics and Natural Sciences, University of Cologne, Zülpicher Straße 45, 50674 Cologne, Germany
[b]Biofluid Simulation and Modeling – Theorethische Physik VI, University of Bayreuth, 95440 Bayreuth, Germany
[c]Animal Ecology I, BayCEER, University of Bayreuth, 95440 Bayreuth, Germany

List of Figures

Supplementary figures

Figure S1: The coarse surfaces with coarse sand (left) and the fine surface with medium sand (right) (Image: M. Klee).

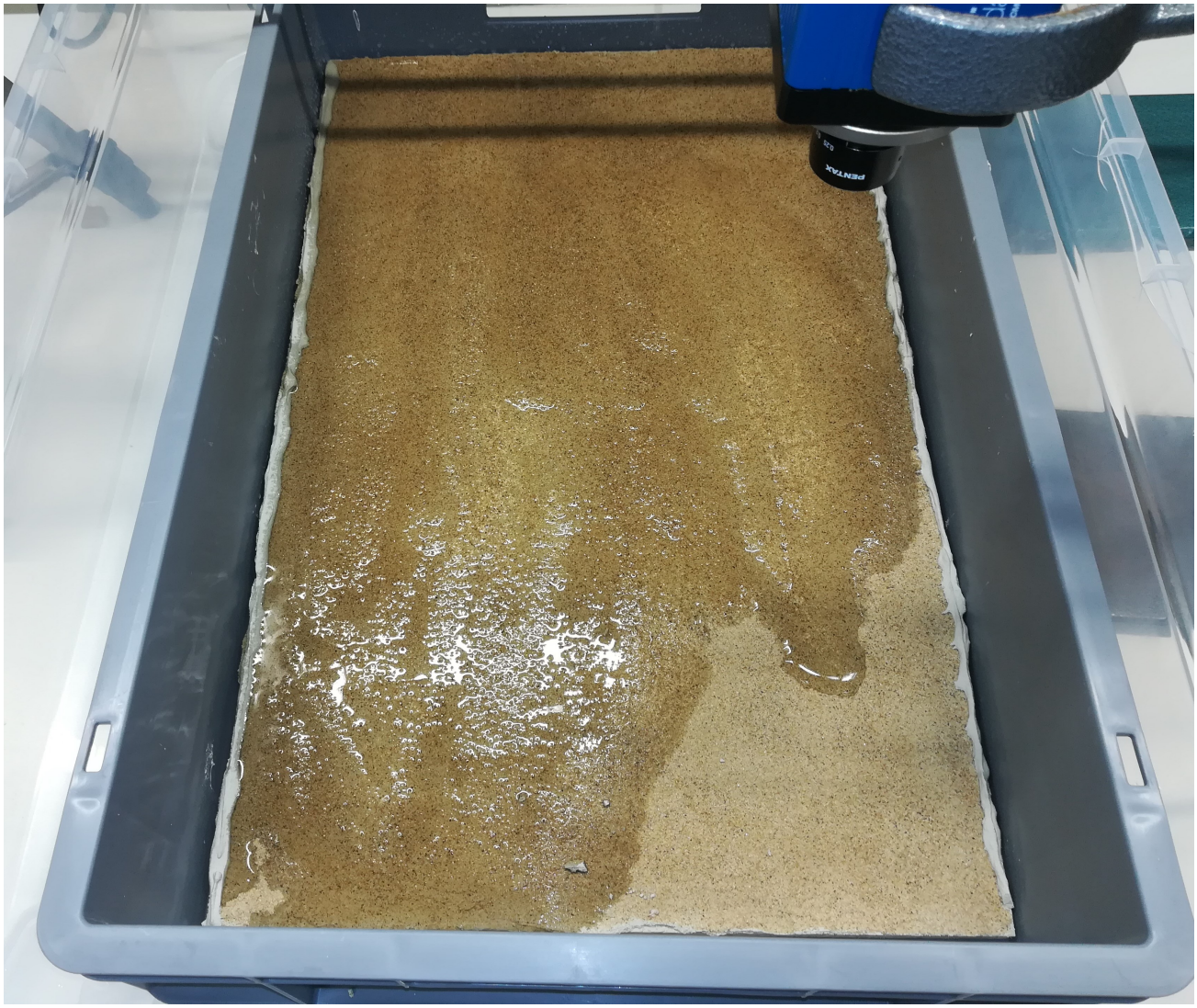


Figure S2: Flow of the water on the fine surface (Image: M. Klee).

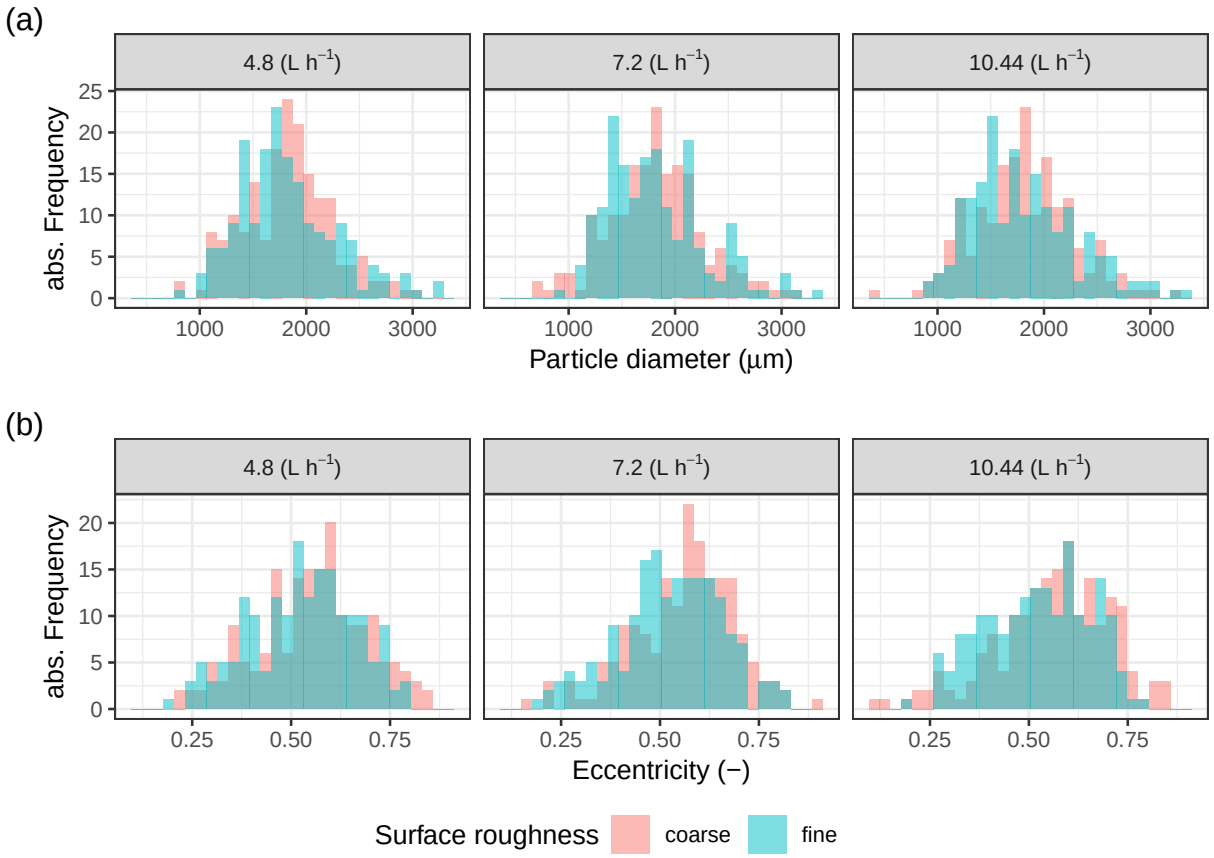


Figure S3: Distribution of (a) the equivalent diameters and (b) eccentricities of PMMA particles on the coarse and the fine surfaces by irrigation rates shown in facets' titles.

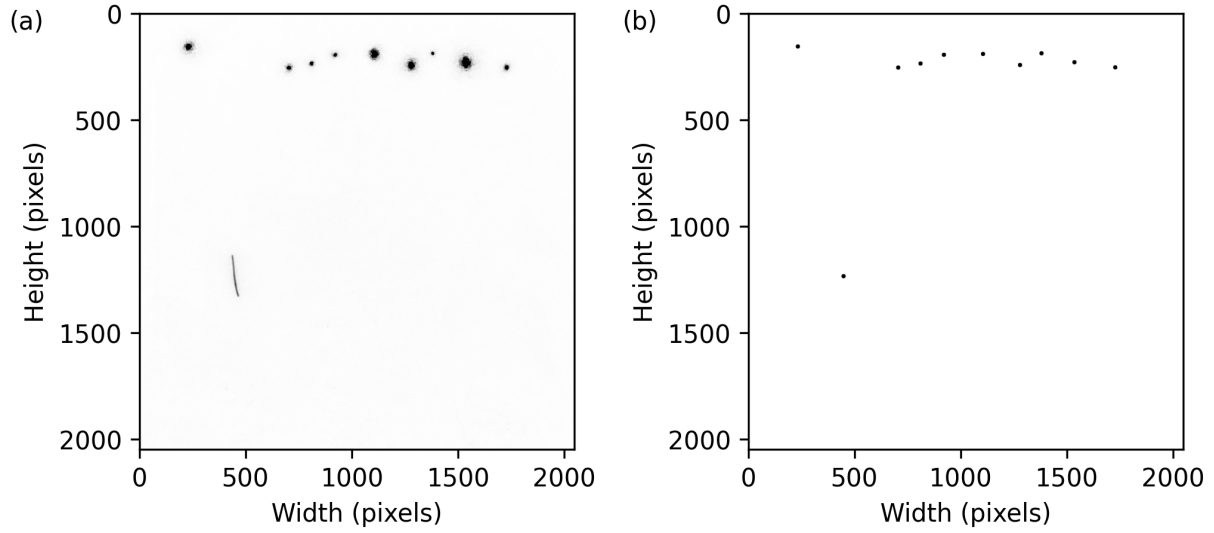


Figure S4: (a) Artefacts on images: abruptly moving particle appearing as a line and blurred particles' edges due to residual ambient light. Original image from the second run on the fine surface at an inclination of 7.5° and an irrigation rate of 10.44 L h^{-1}. (b) Preprocessed image with particles transformed to equally sized circles centered around particles' centers. Additionally, background noise is removed, i.e. the image is binary (black for particles, white for background).

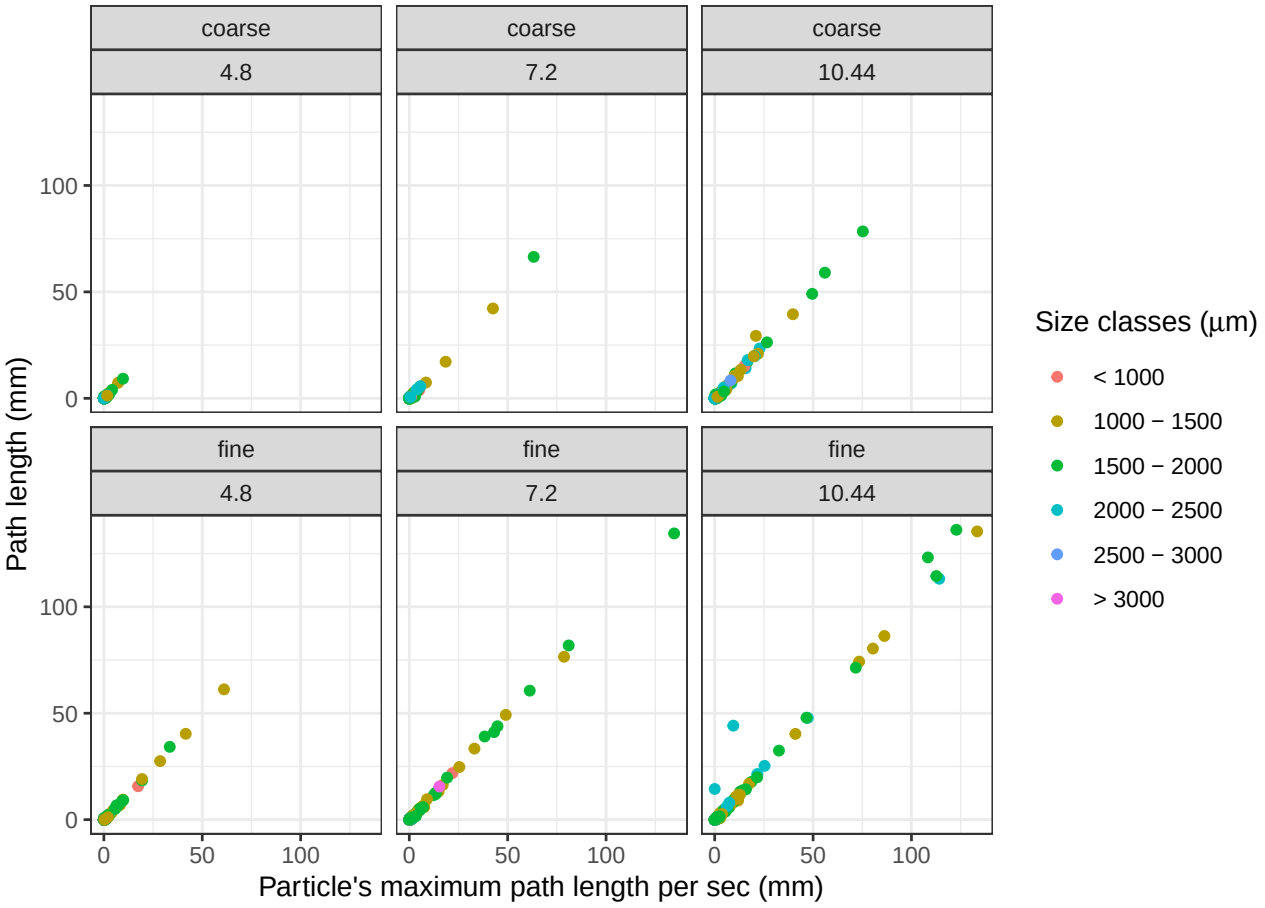


Figure S5: Relationship between particle's maximum path length per second and the overall path length by roughness and irrigation rate (in L h^{-1}).

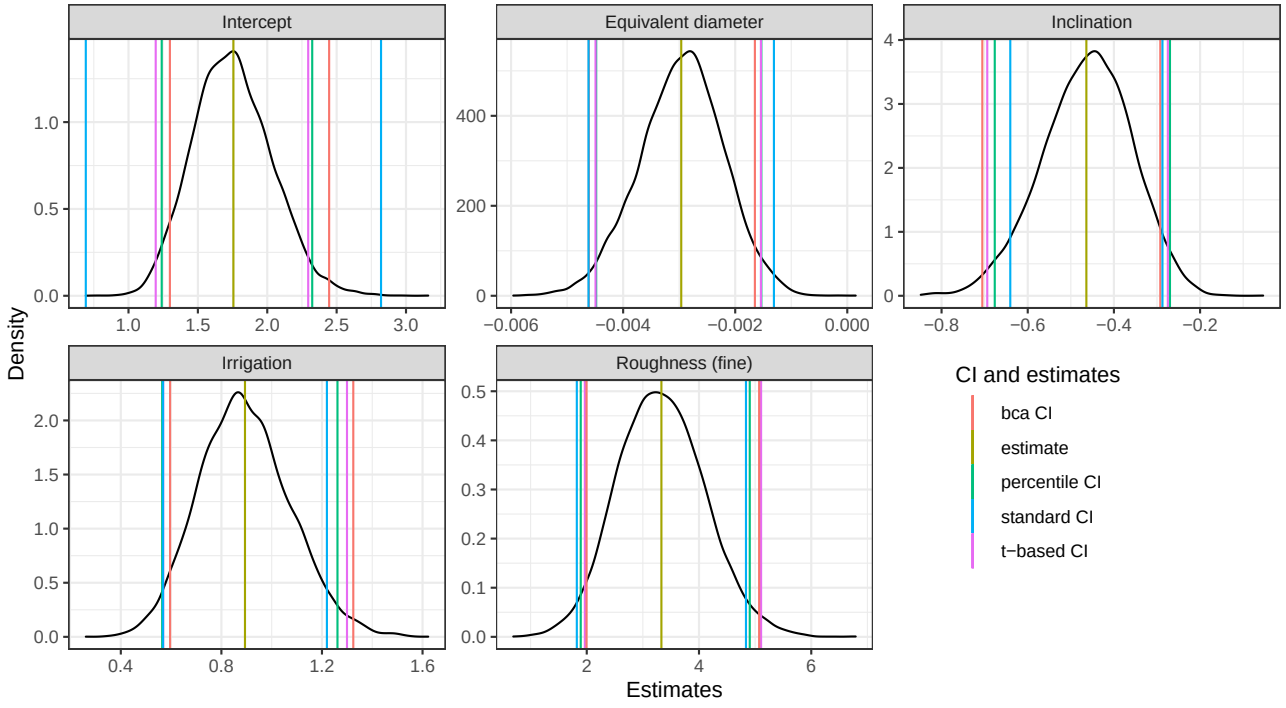


Figure S6: Results of the bootstrapping the linear model. Confidence intervals (CI) were estimated with different methods for comparison keeping $\alpha = 0.05$ for all. estimate: the actual estimated from the linear model, bca CI: bias-corrected and accelerated CI, percentile CI: basic percentile CI, standard CI: CI obtained from the linear model based on the assumption of normal residuals, t-based: t CI using bootstrap standard errors (Davison and Hinkley, 1997).

References

Davison, A. C. and D. V. Hinkley (1997). *Bootstrap Methods and Their Application*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press. DOI: 10.1017/CBO9780511802843.

8.6 Publication 6

Comparison of free surface and conservative Allen-Cahn phase field lattice Boltzmann method

by

Christoph Schwarzmeier, Markus Holzer, Travis Mitchell, Moritz Lehmann, Fabian Häusl, and Ulrich Rüde

Journal of Computational Physics 111753 (2022)
https://doi.org/10.1016/j.jcp.2022.111753

Journal of Computational Physics 473 (2023) 111753

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Comparison of free-surface and conservative Allen–Cahn phase-field lattice Boltzmann method

Christoph Schwarzmeier [a,*], Markus Holzer [a,b], Travis Mitchell [c], Moritz Lehmann [d], Fabian Häusl [d], Ulrich Rüde [a,b]

[a] Chair for System Simulation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstraße 11, 91058 Erlangen, Germany
[b] CERFACS, 42 Avenue Gaspard Coriolis, 31057 Toulouse Cedex 1, France
[c] School of Mechanical and Mining Engineering, The University of Queensland, St Lucia, QLD 4072, Australia
[d] Biofluid Simulation and Modeling - Theoretische Physik VI, University of Bayreuth, 95447 Bayreuth, Germany

ARTICLE INFO

Article history:
Received 22 June 2022
Received in revised form 8 September 2022
Accepted 2 November 2022
Available online 9 November 2022

Keywords:
Lattice Boltzmann method
Free-surface flow
Conservative Allen–Cahn phase-field
Standing wave
Rising bubble
Drop impact

ABSTRACT

This study compares the free-surface lattice Boltzmann method (FSLBM) with the conservative Allen–Cahn phase-field lattice Boltzmann method (PFLBM) in their ability to model two-phase flows in which the behavior of the system is dominated by the heavy phase. Both models are introduced and their individual properties, strengths and weaknesses are thoroughly discussed. Six numerical benchmark cases were simulated with both models, including (i) a standing gravity and (ii) capillary wave, (iii) an unconfined rising gas bubble in liquid, (iv) a Taylor bubble in a cylindrical tube, and (v) the vertical and (vi) oblique impact of a drop into a pool of liquid. Comparing the simulation results with either analytical models or experimental data from the literature, four major observations were made. Firstly, the PFLBM selected was able to simulate flows purely governed by surface tension with reasonable accuracy. Secondly, the FSLBM, a sharp interface model, generally requires a lower resolution than the PFLBM, a diffuse interface model. However, in the limit case of a standing wave, this was not observed. Thirdly, in simulations of a bubble moving in a liquid, the FSLBM accurately predicted the bubble's shape and rise velocity with low computational resolution. Finally, the PFLBM's accuracy is found to be sensitive to the choice of the model's mobility parameter and interface width.

1. Introduction

Multiphase flows are important in a wide range of natural and industrial applications. For instance, they can manifest as undesired foam in the food industry [1,2], in the transport of hydrocarbons from subsurface environments [3], or in the transfer of micro-particles into the environment during rainfall [4]. The laboratory experiments associated with studying the fundamental dynamics of multiphase flows can be expensive and time-consuming, while only supplying limited insight into the governing fluid mechanics. With advances of computational infrastructure, it has now become common to supplement physical with numerical experiments through the use of computational fluid dynamics (CFD). This tends to provide a cheaper, time-efficient solution to flow problems and allows direct insights to the flow field, as every arbitrary location inside the fluid can be monitored.

* Corresponding author.
E-mail address: christoph.schwarzmeier@fau.de (C. Schwarzmeier).

https://doi.org/10.1016/j.jcp.2022.111753
0021-9991/

This work aims to present and analyze a numerical method that can be used to supplement experiments by numerical simulations of immiscible two-phase flows, in which the flow dynamics of the lighter phase are assumed to have a negligible influence on the heavier phase, and overall dynamics of the system. In such cases, the flow in the lighter phase is commonly neglected, reducing the two-phase flow to a flow with a free boundary, or more commonly referred to as free-surface flow [5]. It has been previously shown that such simplification is valid in simulations for e.g., single gas bubbles rising in a liquid [6,7] and has been applied to simulate foaming [8].

While the flow in the lighter phase is assumed to be negligible, the simulation of the flow in the heavier phase often requires a highly resolved computational grid to capture all the relevant flow structures. Therefore, the numerical methods presented here are designed and targeted for massively parallel computing environments. For efficient numerical fluid simulations on such hardware, the lattice Boltzmann method (LBM) has established as a modern alternative to classical approaches for CFD that are based on the discretization of the Navier–Stokes equations. As all operations require only information of a local neighborhood, the LBM is inherently suitable for parallel computing and has been extended with models for simulating a variety of different physics including multiphase flows [9–11], particulate flows [12,13], thermal effects [14] and others.

There are several multiphase LBM models available in the literature that can be distinguished by the representation of the interface between the phases. Models having a sharp interface representation include the free-surface lattice Boltzmann method (FSLBM) [8], the level-set method [15], the front-tracking approach [16], and the color gradient model [17]. In contrast, the interface is represented in a diffuse manner in the pseudopotential model [10], the free-energy model [18], and phase-field models. The latter are either based on solving the Cahn–Hilliard [19,20] or Allen–Cahn [21] equation to model the interfacial dynamics. In this article, the comparative study is restricted to the FSLBM and the conservative Allen–Cahn phase-field LBM (PFLBM) [21,22]. Both of these models have well-optimized parallel implementations, and have been shown to be capable of simulating systems with high density and viscosity contrasts corresponding to liquid–gas systems.

The FSLBM extends the LBM with a volume-of-fluid approach [23] where the sharp interface between the two phases is captured by an indicator field [8]. The fluid dynamics of the lighter phase are entirely neglected, and only the effect of pressure forces at the interface is modeled. It is implicitly assumed that the density and viscosity ratio between the two fluid phases is infinite. The sharp interface formulation and avoiding computations in the lighter phase lead to high computational efficiency with low memory requirements. Although the algorithm's implementation is challenging, it is also well suited for parallel hardware like graphics processing units (GPUs) [4,24].

The conservative Allen–Cahn equation [11,25] is the basis of the conservative Allen–Cahn phase-field LBM [21,22], a model designed to simulate two-phase flow problems with high density and viscosity contrasts. The algorithm is simpler than that of the FSLBM, where different equations must be solved depending on the type of cell. In contrast, the PFLBM can be purely formulated via the standard lattice Boltzmann equation with additional force terms [21]. As in the single-phase LBM, all operations are restricted to a local cell-neighborhood making the PFLBM well-suited for parallel computing. While prior phase-field models were not capable of simulating multiphase flows with large density and viscosity ratios [26–28], the PFLBM has been successfully used in simulations with density ratios of up to 10^3 and viscosity ratios of up to 10^2 [14, 29–32]. This is equivalent to an air–water system and makes the model a possible alternative for free-surface flows where the dynamics are governed by the heavier phase. The Allen–Cahn phase-field equation tracks the dynamics of the interface. The diffusivity of the interface suggests that a PFLBM simulation must have a higher resolution than an FSLBM simulation. On the other hand, due to its algorithmic simplicity, it is easier to optimize the implementation for different architectures, including accelerator hardware like GPUs [33].

In this work, the models are compared with respect to their algorithmic properties and ability to simulate two-phase flows in which the lighter phase has negligible impact on the flow dynamics. First, the numerical foundations of the LBM, FSLBM and PFLBM are introduced in more detail. Then, the models are compared with respect to methodology and numerical implementation. Based on six numerical experiments, the accuracy and the required computational grid resolution of the models are compared. For all numerical experiments, each model's simulation results were cross-validated with independent implementations from other code bases, as listed below. The choice of these tests is discussed, as the test cases must be reasonably applicable to both models. The initial test case features a standing surface wave governed by gravitational forces and is referred to as a gravity wave. Surface tension effects are not modeled in this test case. In the second benchmark, the same standing wave setup is used, however, the flow is governed by surface tension rather than gravitational forces. With respect to the gravity wave test case, the capillary wave allows one to exclusively evaluate the models' capability to capture the effects of surface tension. For both the gravity and capillary wave, there exist analytical models that can be used to validate the simulation accuracy. In the third and fourth test case, an unconfined single rising gas bubble in liquid, and a confined Taylor bubble in a cylindrical tube are simulated and compared with experimental data from the literature. Finally, the fifth and sixth benchmark case feature dynamic coalescence, i.e., the formation of a splash crown caused by the impact of a droplet into liquid. The results are qualitatively compared with experimental data from the literature. Both models are regularly applied to simulate capillary waves [7,8,34], rising bubbles [6,7,34–37], and drop impacts [4,34,36–38], however, a direct comparison between them is missing from the literature. Finally, it is concluded that the PFLBM is more accurate in simulating flows governed purely by surface tension forces compared to the FSLBM used in this article. However, in flow problems governed by surface tension and gravitational acceleration, the FSLBM required less computational resolution than the PFLBM while having more accuracy in the tests performed here. Additionally, the PFLBM was sensitive to the choice of the model's mobility parameter and interface width, affecting accuracy and numerical stability.

In this work, general properties related to computational performance such as the grid's resolution and memory requirements are discussed, while quantitative data are not presented. Data such as these would only represent the state of the implementations used here and would not allow a general conclusion to be made.

The FSLBM simulations were performed using the open source C++ framework waLBerla [39] and cross-validated with FluidX3D [4,40]. The PFLBM simulations were also performed using waLBerla together with the code generation framework *lbmpy* [41]. These simulations were cross-validated using TCLB [42]. The implementations used in this work and all simulation setups are freely available online, as described in Appendix B.

2. Numerical methods

The first part of this section briefly introduces the LBM, before presenting the numerical foundations of the FSLBM and the PFLBM. The section is concluded by comparing both models focusing on their computational properties.

2.1. The lattice Boltzmann method

The classical approach to CFD is to simulate the evolution of a flow problem via the discretization of the Navier–Stokes equations. Contrary to this, the LBM is based on the lattice Boltzmann equation (LBE) and has gained popularity in the last few decades. The LBE is given by,

$$f_i\left(\boldsymbol{x}+\boldsymbol{c}_i\Delta t, t+\Delta t\right)-f_i\left(\boldsymbol{x}, t\right)=\Omega_i\left(\boldsymbol{x}, t\right)+F_i\left(\boldsymbol{x}, t\right), \tag{1}$$

with $f_i\left(\boldsymbol{x}, t\right) \in \mathbb{R}$ being a discrete particle distribution function (PDF) that describes the probability that there exists a virtual fluid particle at position $\boldsymbol{x} \in \mathbb{R}^d$ and time $t \in \mathbb{R}^+$ traveling with discrete lattice velocity $\boldsymbol{c}_i \in \Delta x/\Delta t\,\{-1,0,1\}^d$ [43]. The domain is discretized using a uniformly spaced Cartesian grid with spacing $\Delta x \in \mathbb{R}^+$ where the macroscopic fluid velocity in each cell is discretized using a DdQq velocity set such that $i \in \{0,1,\ldots,q-1\}$. Here, $d \in \mathbb{N}$ refers to the number of dimensions in space and $q \in \mathbb{N}$ refers to the number of discrete lattice velocities. In each velocity set, the so-called lattice speed of sound, $c_s=\sqrt{1/3}\,\Delta x/\Delta t$, defines the relation between density, $\rho(\boldsymbol{x}, t) \in \mathbb{R}^+$, and pressure, $p(\boldsymbol{x}, t)=c_s^2\rho(\boldsymbol{x}, t)$, with $\Delta t \in \mathbb{R}^+$ denoting the temporal resolution. External forces are included in the LBM by $F_i(\boldsymbol{x}, t) \in \mathbb{R}$.

The collision operator, $\Omega_i\left(\boldsymbol{x}, t\right) \in \mathbb{R}$, models particle collisions and redistributes PDFs. In this study, collision operators are based on the multiple relaxation time (MRT) scheme [44] that can be written as,

$$\boldsymbol{\Omega}\left(\boldsymbol{x}, t\right)=\boldsymbol{M}^{-1}\cdot\hat{\boldsymbol{S}}\cdot\boldsymbol{M}\cdot\left(\boldsymbol{f}^{\mathrm{eq}}\left(\rho, \boldsymbol{u}\right)-\boldsymbol{f}\left(\boldsymbol{x}, t\right)\right), \tag{2}$$

where $\boldsymbol{M} \in \mathbb{R}^{q\times q}$ denotes a $q \times q$ Matrix, constructed from a set of q moments, that transforms the PDFs to the moment space [44]. In the moment space, the collision is resolved by subtracting the PDFs' equilibria, $\boldsymbol{f}^{\mathrm{eq}}(\rho, \boldsymbol{u}) \in \mathbb{R}^q$, from the PDFs and applying the diagonal relaxation matrix $\hat{\boldsymbol{S}} \in \mathbb{R}^{q\times q}$. It contains the relaxation rate, $\omega_i < 2/\Delta t$, the inverse of which is referred to as the relaxation time, $\tau_i = 1/\omega_i$. For the MRT employed here, the relaxation time corresponding to second-order moments, τ, is directly related to the kinematic viscosity of the fluid through,

$$\nu=c_s^2\left(\tau-\frac{\Delta t}{2}\right). \tag{3}$$

The equilibrium PDF is given as,

$$f_i^{\mathrm{eq}}(\rho, \boldsymbol{u})=\rho w_i+\rho_0 w_i\left(\frac{\boldsymbol{c}_i\cdot\boldsymbol{u}}{c_s^2}+\frac{(\boldsymbol{c}_i\cdot\boldsymbol{u})^2}{2c_s^4}-\frac{\boldsymbol{u}\cdot\boldsymbol{u}}{2c_s^2}\right), \tag{4}$$

and can be derived from the continuous Maxwell–Boltzmann distribution [45] using the macroscopic velocity, $\boldsymbol{u} \equiv \boldsymbol{u}\left(\boldsymbol{x}, t\right) \in \mathbb{R}^d$, density, $\rho \equiv \rho\left(\boldsymbol{x}, t\right)$, and lattice weight, $w_i \in \mathbb{R}$. When setting the LBM reference density $\rho_0 = 1$ in Equation (4), the incompressible LBM formulation is obtained, whereas $\rho_0 = \rho$ reveals the LBM in compressible form [46].

If the collision operator's moment set is constructed with the so-called raw moments and all moments are relaxed with the same relaxation rate, $\omega = 1/\tau$, the commonly used Bhatnagar–Gross–Krook (BGK), also referred to as single relaxation time (SRT) collision operator is obtained [43],

$$\boldsymbol{\Omega}\left(\boldsymbol{x}, t\right)=\omega\left(\boldsymbol{f}^{\mathrm{eq}}\left(\rho, \boldsymbol{u}\right)-\boldsymbol{f}\left(\boldsymbol{x}, t\right)\right). \tag{5}$$

A major contributor to the LBM's popularity is its formulation as an explicit time-stepping scheme and the fact that all non-linear operations (collision) are local to a computational cell, while the advection (streaming) is linear [47]. This means that Equation (1) can be separated into the subsequent steps of collision and streaming denoted by,

$$f_i^{\star}\left(\boldsymbol{x}, t\right)=f_i\left(\boldsymbol{x}, t\right)+\Omega_i\left(\boldsymbol{x}, t\right), \tag{6}$$

$$f_i\left(\boldsymbol{x}+\boldsymbol{c}_i\Delta t, t+\Delta t\right)=f_i^{\star}\left(\boldsymbol{x}, t\right), \tag{7}$$

174

where $f_i^\star(\boldsymbol{x},t)$ indicates the post-collision status of the PDFs. This illustrates that the resulting scheme can be parallelized well and is therefore excellently-suited for massively parallel, large-scale simulations [33]. In practice usually both steps are combined, as shown in Equation (1), to achieve the best parallel performance [48].

2.2. Free-surface lattice Boltzmann method

The free-surface lattice Boltzmann method (FSLBM) used in this work is based on the approach from Körner et al. [8]. It allows the simulation of a moving interface between two immiscible fluids and assumes that the heavier phase completely governs the flow dynamics of the system. As the flow dynamics of the lighter phase are ignored, the problem reduces to a single-phase flow with a free boundary. This assumption applies to two-phase systems with substantial density and viscosity ratios between the phases. In the following, the heavier and lighter phases will be called liquid and gas phases, respectively.

The boundary between the two phases is treated in a volume-of-fluid approach [23]. As such, the fill level, $\varphi(\boldsymbol{x},t)$, in a cell is defined as the ratio of its liquid volume to its total volume, and acts as an indicator to describe the affiliation to a phase. Using this definition, all cells belonging to the fluid domain are either categorized as liquid ($\varphi(\boldsymbol{x},t)=1$), gas ($\varphi(\boldsymbol{x},t)=0$) or interface ($\varphi(\boldsymbol{x},t)\in(0,1)$). The latter type assembles a sharp interface, i.e., a closed layer of single interface cells that separates liquid from gas cells. In terms of the LBM implementation, liquid and interface cells are treated as normal cells that contain PDFs and participate in the collision and streaming described in Section 2.1. As opposed to this, gas cells neither contain PDFs nor participate in the LBM update.

The fill level, $\varphi(\boldsymbol{x},t)$, fluid density, $\rho(\boldsymbol{x},t)=\sum_i f_i(\boldsymbol{x},t)$, and volume, Δx^3, of a cell are used to define its liquid mass as,

$$m(\boldsymbol{x},t)=\varphi(\boldsymbol{x},t)\,\rho(\boldsymbol{x},t)\,\Delta x^3. \tag{8}$$

The mass flux, $\Delta m_i(\boldsymbol{x},t)$, is tracked for interface cells and computed from the LBM streaming step as,

$$\frac{\Delta m_i(\boldsymbol{x},t)}{\Delta x^3}=\begin{cases}0 & \boldsymbol{x}+\boldsymbol{c}_i\Delta t\in\text{gas}\\ f_{\bar{i}}^\star(\boldsymbol{x}+\boldsymbol{c}_i\Delta t,t)-f_i^\star(\boldsymbol{x},t) & \boldsymbol{x}+\boldsymbol{c}_i\Delta t\in\text{liquid}\\ \frac{1}{2}\left(\varphi(\boldsymbol{x},t)+\varphi(\boldsymbol{x}+\boldsymbol{c}_i\Delta t,t)\right)\left(f_{\bar{i}}^\star(\boldsymbol{x}+\boldsymbol{c}_i\Delta t,t)-f_i^\star(\boldsymbol{x},t)\right) & \boldsymbol{x}+\boldsymbol{c}_i\Delta t\in\text{interface}\end{cases} \tag{9}$$

where $\boldsymbol{c}_{\bar{i}}=-\boldsymbol{c}_i$ is the inversion of the lattice direction i.

An interface cell is converted to gas or liquid when it gets emptied, $\varphi(\boldsymbol{x},t)<0-\varepsilon_\varphi$, or filled, $\varphi(\boldsymbol{x},t)>1+\varepsilon_\varphi$, with respect to the heuristically chosen threshold, $\varepsilon_\varphi=10^{-2}$, that is defined to prevent oscillatory conversions [49]. It is important to note that liquid or gas cells can not be converted directly into one another. Instead, when converting an interface cell, surrounding liquid and gas cells are converted to interface cells to maintain a closed interface layer. In the case of conflicting conversions, the separation of liquid and gas cells is prioritized.

In the course of the simulation, unnecessary interface cells may appear that either have no liquid or no gas neighbor. In that case, the mass flux from Equation (9) is modified as suggested in Reference [38] to either force these cells to fill or empty.

When converting an interface cell with fill level, $\varphi^{\text{conv}}(\boldsymbol{x},t)$, to liquid or gas, the fill level of the converted cell is set to $\varphi(\boldsymbol{x},t)=1$ or $\varphi(\boldsymbol{x},t)=0$, respectively. This leads to an excess mass, $m_{\text{ex}}(\boldsymbol{x},t)$, of,

$$\frac{m_{\text{ex}}(\boldsymbol{x},t)}{\rho(\boldsymbol{x},t)\Delta x^3}=\begin{cases}\varphi^{\text{conv}}(\boldsymbol{x},t)-1 & \text{if } \boldsymbol{x} \text{ is converted to liquid}\\ \varphi^{\text{conv}}(\boldsymbol{x},t) & \text{if } \boldsymbol{x} \text{ is converted to gas}\end{cases} \tag{10}$$

that must be distributed to neighboring cells. In this work, excessive mass is distributed evenly among surrounding interface cells, or evenly among surrounding interface and liquid cells in the implementation in FluidX3D.

A cell conversion from liquid to interface and vice-versa does not modify the PDFs of the cell. In contrast, the PDFs in cells converted from gas to interface are not yet available. They are initialized using Equation (4) with ρ, and $\boldsymbol{u}$ averaged from all surrounding liquid and non-newly converted interface cells.

The LBM collision as in Equation (6) is applied to all liquid and interface cells with Equation (4) being used in compressible form. Unlike suggested in Reference [8], the gravitational force is not weighted with the fill level of an interface cell in the implementation used here.

During the LBM streaming step, according to Equation (7), PDFs streaming from gas cells to interface cells do not exist and must be reconstructed. This is accomplished using an anti-bounce-back pressure boundary condition at the interface,

$$f_i^\star(\boldsymbol{x}-\boldsymbol{c}_i\Delta t,t)=f_i^{\text{eq}}\left(\rho^{\text{G}},\boldsymbol{u}\right)+f_{\bar{i}}^{\text{eq}}\left(\rho^{\text{G}},\boldsymbol{u}\right)-f_{\bar{i}}^\star(\boldsymbol{x},t)\quad\forall i:\boldsymbol{x}-\boldsymbol{c_i}\Delta t\in\text{gas} \tag{11}$$

where $\boldsymbol{u}\equiv\boldsymbol{u}(\boldsymbol{x},t)$ is the velocity of the interface cell and $\rho^{\text{G}}\equiv\rho^{\text{G}}(\boldsymbol{x},t)=p^{\text{G}}(\boldsymbol{x},t)/c_s^2$ is the gas density computed from the pressure of the gas phase, $p^{\text{G}}(\boldsymbol{x},t)$. In the original model [8], it was suggested to reconstruct PDFs based on their orientation with respect to the interface normal. However, this approach overwrites existing information and was observed to lead to anisotropic artifacts [50,51]. Here, as suggested in Reference [50], only missing PDFs are reconstructed, and no information is dropped.

The gas pressure,

$$p^{\mathrm{G}}(\boldsymbol{x},t) = p^{\mathrm{V}}(t) - p^{\mathrm{L}}(\boldsymbol{x},t), \tag{12}$$

consists of the volume pressure, $p^{\mathrm{V}}(t) \in \mathbb{R}^+$, and the Laplace pressure, $p^{\mathrm{L}}(\boldsymbol{x},t) \in \mathbb{R}^+$. The volume pressure can be either atmospheric in which $p^{\mathrm{V}}(t) = \text{constant}$ or result from the change of the volume, $V(t) \in \mathbb{R}^+$, of an enclosed gas volume, i.e., bubble with,

$$p^{\mathrm{V}}(t) = p^{\mathrm{V}}(0)\,\frac{V(0)}{V(t)}. \tag{13}$$

The Laplace pressure,

$$p^{\mathrm{L}}(\boldsymbol{x},t) = 2\sigma\kappa(\boldsymbol{x},t), \tag{14}$$

incorporates the surface tension, $\sigma \in \mathbb{R}^+$, and the interface curvature, $\kappa(\boldsymbol{x},t) \in \mathbb{R}$. There exist different approaches for computing the interface curvature that are based on the finite difference method (FDM) or a local triangulation of the interface [40,50]. The simulation results shown here are obtained using the FDM as described in Reference [50]. The interface normal, as required by the FDM curvature model, is modified near solid obstacle cells according to Reference [52]. Other curvature computation models have been tested and will be discussed in Section 3.1.2.

In applications where bubbles must be properly simulated, an additional bubble model extension is required for the FSLBM. Since gas volumes can coalesce and divide, this algorithm must keep track of the volume pressure of individual bubbles and handle coalescence and segmentation accordingly. Such algorithms are referred to as bubble models and are algorithmically challenging when applied in parallel computing environments. Here, the bubble model from Reference [49] is used to simulate bubble coalescence correctly and in parallel environments. It is based on the combination of the interface normal and the *seed-fill* algorithm [53]. In contrast, in FluidX3D, the bubble model is based on the *Hoshen–Kopelman* algorithm [54].

2.3. Conservative Allen–Cahn model

The conservative Allen–Cahn model is described in several other publications [22,32,36]. Here, the governing equations and their discretization with the LBM are only briefly introduced.

2.3.1. Governing equations

The phase-field model studied in this work is built on the following macroscopic equations,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \boldsymbol{u} = 0, \tag{15}$$

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = -\nabla p + \nabla\cdot\left(\mu\left[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right]\right) + \boldsymbol{F}_s + \boldsymbol{F}_b, \tag{16}$$

$$\frac{\partial \phi}{\partial t} + \boldsymbol{\nabla}\cdot(\phi\boldsymbol{u}) = \boldsymbol{\nabla}\cdot M\left(\boldsymbol{\nabla}\phi - \frac{1-(\phi-\phi_0)}{\xi}\boldsymbol{n}\right), \tag{17}$$

the first of which represents the continuity equation. Equation (16) is the momentum equation with the hydrodynamic pressure, $p \equiv p(\boldsymbol{x},t)$, and Equation (17) is the Allen–Cahn equation used for the tracking of the interface. Here, the mobility is denoted by $M \in \mathbb{R}^+$, the interface width by $\xi \in \mathbb{N}^+$, $\boldsymbol{n} \equiv \boldsymbol{n}(\boldsymbol{x},t) = \nabla\phi/|\nabla\phi|$ is the unit vector normal to the liquid–gas interface, and $\mu \in \mathbb{R}^+$ is the fluid's dynamic viscosity.

The principle behind phase-field models is to allocate an additional scalar field for the phase indicator parameter, $\phi \equiv \phi(\boldsymbol{x},t) \in [0,1]$. This phase indicator represents the fluid with higher density by $\phi_H = 1$ and the lower density fluid by $\phi_{\mathrm{L}} = 0$. The bounds of ϕ_{H} and ϕ_{L} can be seen to vary in the literature, and is generally a point of contention. Nonetheless, the authors specify the bounds as $(0,1)$ to minimize issues that may otherwise arise in the light phase fluid.

The forces acting on the fluid include the body force associated with gravity, and the surface tension forces resulting from the liquid-gas interface. These are given as,

$$\boldsymbol{F}_b \equiv \boldsymbol{F}_b(\boldsymbol{x},t) = \rho(\boldsymbol{x},t)\boldsymbol{g}, \tag{18}$$

$$\boldsymbol{F}_s \equiv \boldsymbol{F}_s(\boldsymbol{x},t) = \mu_\phi\nabla\phi(\boldsymbol{x},t), \tag{19}$$

respectively, with gravitational acceleration, $\boldsymbol{g} \in \mathbb{R}^d$ and chemical potential, $\mu_\phi \in \mathbb{R}$.

176

2.3.2. Lattice Boltzmann equations

Discretizing the conservative Allen–Cahn equation with the LBM yields,

$$h_i(\boldsymbol{x}+\boldsymbol{c}_i\Delta t, t+\Delta t) - h_i(\boldsymbol{x},t) = \Omega_i^h \left[h_i^{\mathrm{eq}}(\phi,\boldsymbol{u}) - h_i(\boldsymbol{x},t)\right]|_{(\boldsymbol{x},t)}, \tag{20}$$

where the collision operator of the phase-field LBE is given by $\Omega_i^h(\boldsymbol{x},t) \in \mathbb{R}$, the phase-field PDFs by $h_i(\boldsymbol{x},t) \in \mathbb{R}$, and the phase-field relaxation time by,

$$\tau_\phi = M/c_s^2. \tag{21}$$

Thus, the mobility of the interface defines the behavior of the interface tracking LBM step. The density

$$\rho(\boldsymbol{x},t) = \rho(\phi) = \rho_{\mathrm{L}} + (\phi(\boldsymbol{x},t) - \phi_{\mathrm{L}})(\rho_{\mathrm{H}} - \rho_{\mathrm{L}}) \tag{22}$$

as used in the PDF equilibrium in Equation (4), is computed via interpolation from the phase indicator $\phi(\boldsymbol{x},t)$ as suggested in Reference [21]. Using this formulation of the LBM step, the zeroth-order moment,

$$\phi(\boldsymbol{x},t) = \sum_q h_q(\boldsymbol{x},t), \tag{23}$$

computes $\phi(\boldsymbol{x},t)$. The conservative Allen–Cahn equation is recovered by applying,

$$\boldsymbol{F}^\phi(\boldsymbol{x},t) = \frac{4\phi(1-\phi)}{\xi}\cdot\boldsymbol{n}, \tag{24}$$

in the collision space according to Guo's forcing scheme [29,55].

The LBE for the hydrodynamics is given by,

$$g_i(\boldsymbol{x}+\boldsymbol{c}_i\Delta t, t+\Delta t) - g_i(\boldsymbol{x},t) = \Omega_i^g \left[g_i^{\mathrm{eq}}(p^*,\boldsymbol{u}) - g_i(\boldsymbol{x},t)\right]|_{(\boldsymbol{x},t)}, \tag{25}$$

with collision operator, $\Omega_i^g(\boldsymbol{x},t) \in \mathbb{R}$, for the hydrodynamic PDFs, $g_i(\boldsymbol{x},t) \in \mathbb{R}$, and normalized pressure, $p^* \equiv p^*(\boldsymbol{x},t) = p(\boldsymbol{x},t)/(\rho(\boldsymbol{x},t)c_s^2)$. Note here that the LBE is formulated such that the zeroth-order moment recovers the normalized pressure,

$$p^*(\boldsymbol{x},t) = \sum_i g_i(\boldsymbol{x},t). \tag{26}$$

Additionally, it is important to notice that for $g_i^{\mathrm{eq}}(p^*,\boldsymbol{u}) \in \mathbb{R}$, the incompressible formulation of the equilibrium PDFs is used.

The forcing term to recover the Navier–Stokes equation is,

$$\boldsymbol{F}(\boldsymbol{x},t) = \boldsymbol{F}_s + \boldsymbol{F}_b + \boldsymbol{F}_p + \boldsymbol{F}_\mu, \tag{27}$$

which consists of terms to recover the correct pressure gradient term, $\boldsymbol{F}_p \equiv \boldsymbol{F}_p(\boldsymbol{x},t) \in \mathbb{R}$, the viscous forces, $\boldsymbol{F}_\mu \equiv \boldsymbol{F}_\mu(\boldsymbol{x},t) \in \mathbb{R}$, the surface forces, $\boldsymbol{F}_s \equiv \boldsymbol{F}_s(\boldsymbol{x},t) \in \mathbb{R}$, and the body forces, $\boldsymbol{F}_b \equiv \boldsymbol{F}_b(\boldsymbol{x},t) \in \mathbb{R}$. The force vector is directly applied in the collision space according to Guo's forcing scheme [29,55]. The pressure and viscous forces are given as,

$$\boldsymbol{F}_p(\boldsymbol{x},t) = -p^* c_s^2(\rho_{\mathrm{H}} - \rho_{\mathrm{L}})\nabla\phi, \tag{28}$$

$$\boldsymbol{F}_\mu(\boldsymbol{x},t) = \nu(\rho_{\mathrm{H}} - \rho_{\mathrm{L}})\left[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right]\cdot\nabla\phi, \tag{29}$$

where $\rho_{\mathrm{H}} \equiv \rho_{\mathrm{H}}(\boldsymbol{x},t)$, and $\rho_{\mathrm{L}} \equiv \rho_{\mathrm{L}}(\boldsymbol{x},t)$ denote the density in the heavy and light phase, respectively [36]. The kinematic viscosity, $\nu \equiv \nu(\boldsymbol{x},t)$, is computed with Equation (3) using the linearly interpolated relaxation time

$$\tau(\boldsymbol{x},t) = \tau(\phi) = \tau_{\mathrm{L}} + (\phi(\boldsymbol{x},t) - \phi_{\mathrm{L}})(\tau_{\mathrm{H}} - \tau_{\mathrm{L}}), \tag{30}$$

where $\tau_{\mathrm{H}} \equiv \tau_{\mathrm{H}}(\boldsymbol{x},t)$ is the relaxation time of the heavy phase and $\tau_{\mathrm{L}} \equiv \tau_{\mathrm{L}}(\boldsymbol{x},t)$ is the relaxation time of the light phase. It is noted here that the deviatoric stress tensor can be obtained from moments of the non-equilibrium distribution to avoid the need for finite difference approximations in the velocity field.

2.4. Comparison of methodology and numerical implementation

In this section, the FSLBM and PFLBM are compared in terms of various aspects ranging from methodology to implementation. This is done to illustrate the various assumptions made in each model, and provide an understanding for the quantitative comparisons made in the later sections.

177

2.4.1. Treatment of the low density phase

One of the major differences between the FSLBM and the PFLBM presented is the treatment of the lighter fluid phase. Contrary to the PFLBM, the flow dynamics of the lighter phase are ignored in the FSLBM. Although this allows the PFLBM to be applicable to a broader range of applications, this work focuses on flows where the lighter phase is believed to have negligible influence on the system. In this case, the computations in the second phase are assumed to be unnecessary when using the PFLBM. Further to this, the lighter phase has a lower viscosity than the heavier phase. Consequently, the flow is more likely to become turbulent, and Δx and Δt must be chosen to avoid instabilities in the lighter phase. Concerning the heavier phase, Δx and Δt tend to be much smaller than necessary for stability. This impacts the efficiency of the simulation and is one of the driving motivations of the FSLBM. While not considered here, these drawbacks could be moderately compensated by using adaptive refinement of the computational grid [37].

2.4.2. Representation of the interface

Another significant difference between the two models is the representation of the interface between the phases. In the FSLBM, the interface layer has a width of one cell and is therefore referred to as a sharp interface. The fill level in the cell captures the interfacial movement. On the other hand, the PFLBM represents the interface layer in a diffuse manner with a width of typically around five lattice cells [22]. The Allen–Cahn equation describes the advection of the interface. In general, it is preferential that the interface width is more than a magnitude smaller than the smallest characteristic length scale of the system [43].

2.4.3. Conservation of mass

Both models in their originally proposed states are mass conserving [8,11]. However, it was observed that single interface cells can become trapped in liquid or gas in the FSLBM [38]. It is argued that these artifacts do not perturb the fluid simulation but are visible as artifacts. To resolve these artifacts, it is suggested to forcefully convert these cells to the cell type in their surrounding, leading to a loss in mass. Following this approach, the current FSLBM implementation does not fully conserve mass.

2.4.4. Numerical implementation

This section focuses on implementation-related aspects of the FSLBM and PFLBM, such as their applicability to code generation, parallel computing, and memory requirements.

Code generation With metaprogramming techniques, it is possible to describe the complete PFLBM model in an abstract symbolic form embedded in a high-level programming language, e.g., Python [33]. Highly optimized code in a performance-oriented programming language, e.g., C or CUDA, is generated automatically. Furthermore, performance optimizations, including spatial blocking, common subexpression elimination, and simultaneous instructions on multiple data (SIMD) vectorization, are applied by the code generator. This provides portability to different computing architectures, such as accelerator hardware like GPUs, and reduces code complexity while increasing the maintainability of the code base. The PFLBM consists of essentially only two continuous LBM steps, making it perfectly applicable for code generation. Here, the entire model, including boundary conditions, forces, and inter-process communication, is implemented using the code generation framework *lbmpy* [41].

In contrast, the FSLBM is not expected to be as well suited to code generation directly. While a compute kernel for the LBM step can be generated, many other model components are not inherently suitable to code generation. In various parts of the model, the type and direction of a neighboring cell define the operation. For instance, in the mass exchange algorithm, Equation (9), different lattice directions have to be treated according to the type of the neighboring cell in this direction. The abstract form of the code might then be similar to the direct implementation in a performance-oriented programming language. Therefore, future work remains to evaluate the applicability of the FSLBM to code generation techniques.

Parallelization The common requirement of a highly resolved computational grid can often not be sufficiently computed on a single processor or compute node of a cluster for practically relevant simulations. The PFLBM scales almost perfectly [33] on parallel computing environments and inherently tracks coalescence and segmentation of gas volumes through the Allen–Cahn equation.

Without modeling bubble coalescence and segmentation, parallelization of the FSLBM is straightforward on any parallel hardware, scaling just as well as the single-phase LBM. It must be remarked that this is sufficient for a wide variety of applications such as the standing wave and drop impact test cases presented in Section 3. However, when tracking individual gas volumes, a bubble model is required that monitors information such as the bubble's identifiers, the gas pressure, and the identifier of the process on which parts of the bubble reside. The parallel implementation of a bubble model is challenging and the models presented in Reference [49] rely on either global all-to-all communication or global sequential communication in each LBM time step. As an extension to that, Reference [56] presented more complicated bubble models where regional all-to-all or sequential communication is sufficient. However, Reference [56] has shown that neither of the mentioned bubble models scale ideally on a parallel computing environment relying on inter-process communication. In the implementation used in this study, the model from Reference [49] with global all-to-all communication is used.

178

Memory requirements The FSLBM requires similar memory allocation to a single-phase LBM implementation, making it well suited for systems with limited memory like GPUs. In contrast, the PFLBM requires a second LBM step with separate PDFs for the phase field, approximately doubling the amount of memory required to be stored at each lattice cell, making it less attractive for hardware with constrained memory. In particular, in setups where high surface detail (e.g., a large number of small droplets) needs to be resolved, with the FSLBM, droplets can have a minimum diameter of three cells. With the PFLBM on the other hand, the minimum droplet diameter is increased to at least 10 cells. To match resolved surface details, the lateral increase in lattice resolution for the PFLBM combined with the higher memory requirements per cell leads to approximately a $2 \cdot (10/3)^3 \approx 74$-fold increase in required memory, making the FSLBM clearly the better choice in such use-cases.

3. Numerical experiments

This section compares the FSLBM and the PFLBM using numerical experiments. Choosing the proper test case for comparing two distinct models in terms of accuracy and computational performance is a challenging task. One must select test cases to which both models are applicable, and keep in mind that each model may be subjected to different forms of errors. Additionally, it is crucial to select a benchmark where the correct solution is known *a priori*, either from experimental data or analytical models to give a point of reference for the modeled results.

While many references provide experimental data for different kinds of multiphase flows, comparing two models based on only experimental measurements can be misleading. Every experiment is subject to uncertainties that can not be considered in numerical simulations, and if both models disagree with experimental observations in contradicting form, no meaningful conclusion can be drawn. Therefore, it is always preferable to base the initial comparison on test cases, for which the exact solution is known from analytical calculations.

Numerical tools used for fluid simulations generally consist of various, coupled models responsible for certain physical aspects. For instance, each of the models discussed here has multiple approaches for including wetting effects [50,52,57,58]. In an initial comparison, a suitable test case should only include the minimally required components of the models to avoid drawing incorrect conclusions caused by a single component in one of the models.

Here, six numerical experiments were used to compare the FSLBM and PFLBM. Citations have been provided to literature in which each of these models has been applied to the chosen test cases, arguably showing that they are both applicable modeling procedures for the cases. Two of these tests simulated a standing wave with analytical models available in the literature. The two cases differed by only the driving force in the flow. While a gravity wave oscillates due to a body force, a capillary wave does so due to the forces resulting from surface tension. In each of the test cases, the respective other force was neglected. The third and fourth test cases featured unconfined and confined buoyancy driven flows. That is, simulations of a gas bubble rising in a large pool of liquid and a Taylor bubble traveling through a cylindrical pipe, both of which were compared with experimental data from the literature. In the final test cases, dynamic coalescence was investigated by simulating the impact of a vertical and oblique drop into a pool of liquid. The results were qualitatively compared to photographs of the laboratory experiments from the literature.

In all simulations with the FSLBM, the SRT collision model from Equation (5) was used. To improve numerical stability in the PFLBM, a weighted orthogonal MRT collision model according to Equation (2) was employed and the individual moments in both LBM steps were relaxed according to Reference [34]. It is important to note here that also the second-order moments for the interface tracking LBM step were relaxed with τ_ϕ. Within all test cases, the specified relaxation rates were constant across the various resolutions leading to what is also known as diffusive scaling in the LBM. Setting the second-order moments directly to the equilibrium led to nonphysical results. Both models used the D2Q9 velocity set for the standing wave simulations. For all other simulations, a D3Q19 velocity set was employed by the FSLBM while the PFLBM was set up with two D3Q27 lattices for the two LBM steps. It is common to introduce the Cahn number, $\mathrm{Cn} = \xi/L$, to describe the PFLBM's interface width ξ. It is highlighted in this work that for convergence assessments, the value of ξ remained constant rather than Cn, as solutions are desired to tend towards a sharp interface result. In the FSLBM, body forces were modeled according to Guo et al. [55]. The forcing terms applied to the LBM steps in the PFLBM model were according to Reference [21]. In the simulations of both models, no-slip boundary walls were realized through the bounce-back boundary condition [43]. In agreement with the usual choice in the LBM literature, the reference density was chosen to be $\rho_0 = \rho_H = 1$ in all simulations.

In the FSLBM, the fill level was initialized with a Monte Carlo-like sampling method. A two-dimensional grid consisting of equally spaced, 101×101, sample points was created in each cell. The ratio of samples within the specified initial profile to the total number of samples per cell gave the initial fill level. In the PFLBM, the diffuse interface was initialized with,

$$\phi_x = \phi_0 \pm \frac{\phi_\mathrm{H} - \phi_\mathrm{L}}{2} \tanh\left(\frac{x - x_0}{\xi/2}\right), \tag{31}$$

in the direction normal to an interface located at x_0.

The surface meshes visualized for the bubbles and drop impacts were obtained using a marching cube algorithm with destination value $\varphi = 0.5$ and $\phi = 0.5$ for the FSLBM and PFLBM, respectively. If not explicitly specified otherwise, all quantities but non-dimensional numbers are denoted in the lattice Boltzmann unit system. All simulations shown in this article were performed with double-precision floating-point arithmetic.

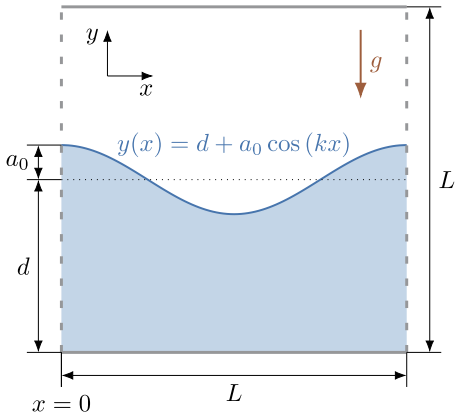


Fig. 1. Simulation setup of the two-dimensional standing gravity and capillary wave with wavelength, L, liquid depth, d, and initial wave amplitude, a_0. There were periodic boundary conditions at the domain's side-walls in x-direction and no-slip boundary conditions at the top- and bottom walls in y-direction. The gravitational acceleration, g, was only present in the gravity wave test.

3.1. Standing waves

In this section, both models' simulation results for a standing gravity and capillary wave are presented and compared with their analytical solutions.

3.1.1. Gravity wave

A gravity wave is a standing wave that oscillates at the phase boundary between two immiscible fluids. Its fluid dynamics are entirely governed by gravitational forces, with surface tension forces being negligible in comparison.

Simulation setup A gravity wave with wavelength, L, was simulated in a quadratic domain of size $L \times L \times 1$ (x-, y-, z-direction). As illustrated in Fig. 1, a free boundary was initialized with the profile, $y(x) = d + a_0 \cos(kx)$, with liquid depth, $d = 0.5L$, initial amplitude, $a_0 = 0.01L$, and wavenumber, $k = 2\pi/L$. There were no-slip boundary conditions at both walls in the y-direction and periodic boundary conditions at all other domain walls. Due to the gravitational acceleration, g, the initial profile evolved into a standing wave oscillating around the liquid depth, d, and dampened by viscous forces. The Reynolds number,

$$\mathrm{Re} = \frac{a_0 \omega_0 L}{\nu}, \tag{32}$$

is defined by the angular frequency of the wave,

$$\omega_0 = \sqrt{gk \tanh(kd)}. \tag{33}$$

In both models, the heavier phase was initialized with hydrostatic pressure according to g such that the LBM pressure at $y(x) = d$ equaled the constant atmospheric volume pressure $p^{\mathrm{V}}(t) = p_0 = \rho_0 c_s^2 = 1/3$.

The surface elevation, $a^*(x,t) = a(x,t)/a_0$, and the time, $t^* = t\omega_0$, are non-dimensionalized to ease comparison. The simulations were run until $t^* = 80$ and the surface elevation, i.e., amplitude $a(x,t)$, was monitored at $x = 0$ every $t^* = 0.1$. It was computed by the sum of all cells' fill levels in the y-direction at $x = 0$ in the FSLBM. In the PFLBM, the surface elevation was evaluated by interpolating the position at which the phase-field value is $\phi = 0.5$.

The simulations were carried out with $\mathrm{Re} = 10$ and $L \in \{50, 100, 200, 400, 800\}$ for the FSLBM and $L \in \{50, 100, 200, 400\}$ for the PFLBM. The FSLBM's gas phase was considered to be the atmosphere, having a constant atmospheric volume pressure of $p^{\mathrm{V}}(t) = p_0$ defined by the LBM reference density $\rho_0 = 1$. In the PFLBM, the density ratio, $\tilde{\rho} = \rho_{\mathrm{H}}/\rho_{\mathrm{L}} = 1000$, and kinematic viscosity ratio, $\tilde{\nu} = \nu_{\mathrm{H}}/\nu_{\mathrm{L}} = 1$, mimic a liquid–gas system and were chosen to conform with the analytical solution of the capillary wave in Section 3.1.2. The relaxation rate was set to $\omega = 1.8$ and $\omega_{\mathrm{H}} = 1.99$, for the FSLBM and for the heavy phase in the PFLBM, respectively. The mobility, $M = 0.02$, and interface width, $\xi = 5$, were chosen in the PFLBM conforming to usual choices in the literature [29].

Analytical model The analytical model for the gravity wave is derived by linearization of the continuity and Euler equations with a free-surface boundary condition [59]. The surface elevation, i.e., the amplitude of the standing wave,

$$a(x,t) = a_{\mathrm{D}}(t) \cos(kx - \omega_0 t) + d, \tag{34}$$

is obtained under the assumption of an inviscid fluid resulting in zero damping with $a_{\mathrm{D}}(t) = a_0$. Viscous damping is considered by,

180

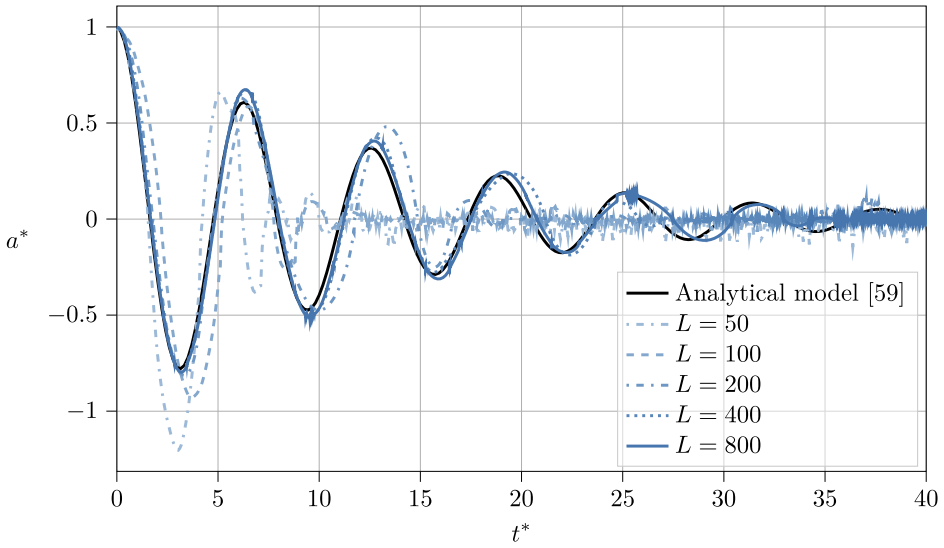


Fig. 2. Gravity wave as simulated by the FSLBM with $L \in \{50, 100, 200, 400, 800\}$ in terms of non-dimensional amplitude, $a^*(0, t^*)$, and time, t^*. Due to the small initial amplitude, $a_0 = 0.01L$, the FSLBM's resolution must be sufficiently high to capture the movement of the interface.

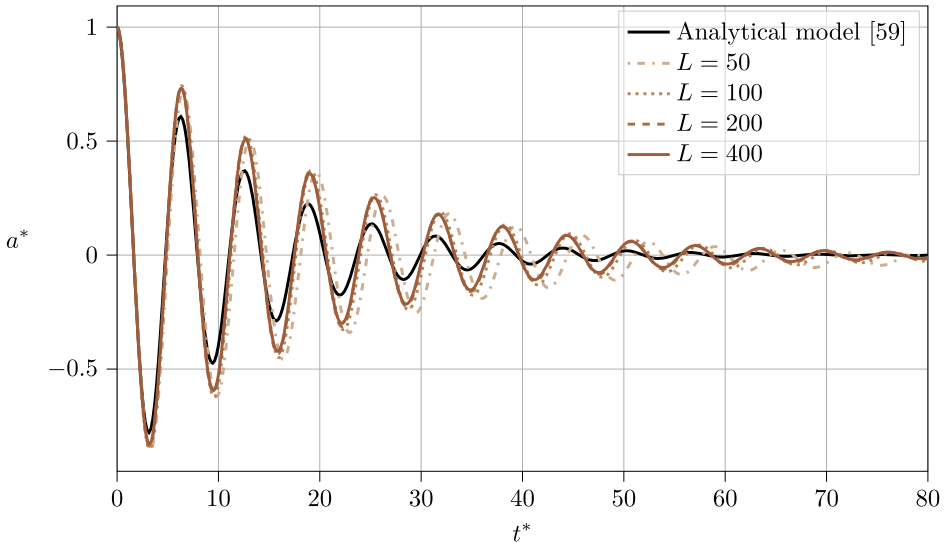


Fig. 3. Gravity wave as simulated by the PFLBM with $L \in \{50, 100, 200, 400\}$ in terms of non-dimensional amplitude, $a^*(0, t^*)$, and time, t^*. The model was able to capture small interface movement even with low resolution.

$$a_D(t) = a_0 e^{-2\nu k^2 t}, \tag{35}$$

as provided in Reference [60]. The model is valid for $k|a_0| \ll 1$ and $k|a_0| \ll kd$ [59], which is applicable in this study with $k|a_0| = 0.02\pi \ll 1 < kd = \pi$.

Results and discussion Fig. 2 shows the amplitude, $a^*(0, t^*)$, over time, t^*, for different wavelengths, L, simulated with the FSLBM. As immediately evident, the FSLBM could not reasonably simulate the gravity wave setup chosen here with small resolutions. This is caused by the requirement of a small initial amplitude, $a_0 = 0.01L$, to be consistent with the analytical solution. The surface of the wave moves only in a range of a few LBM cells or even purely within one cell. This could not be simulated with sufficient accuracy with the FSLBM due to its sharp interface representation on a fixed Cartesian grid. On the other hand, with higher resolution, the amplitudes span more cells and the FSLBM converged well with reasonable accuracy to the analytical model. In particular, a resolution of $L = 50$ did not allow a meaningful simulation, however, $L \in \{100, 200, 400, 800\}$ allowed $2, 3, 4, 5$ periods to be simulated.

Due to the diffuse interface of the PFLBM, the model was capable of simulating even very small amplitudes as shown in Fig. 3. The simulations converged well and from $L = 100$ on, the phase of the wave was predicted accurately. However, the model clearly underestimated the wave's damping for the parameters used in this study.

In Fig. 4, the FSLBM and PFLBM are compared directly. The resolution of the FSLBM was chosen such that a sufficient number of periods have been simulated to allow a meaningful comparison. The computational grid had to be resolved to a very high level to have the amplitude to span over multiple cells (note that many fewer periods were resolved by the FSLBM

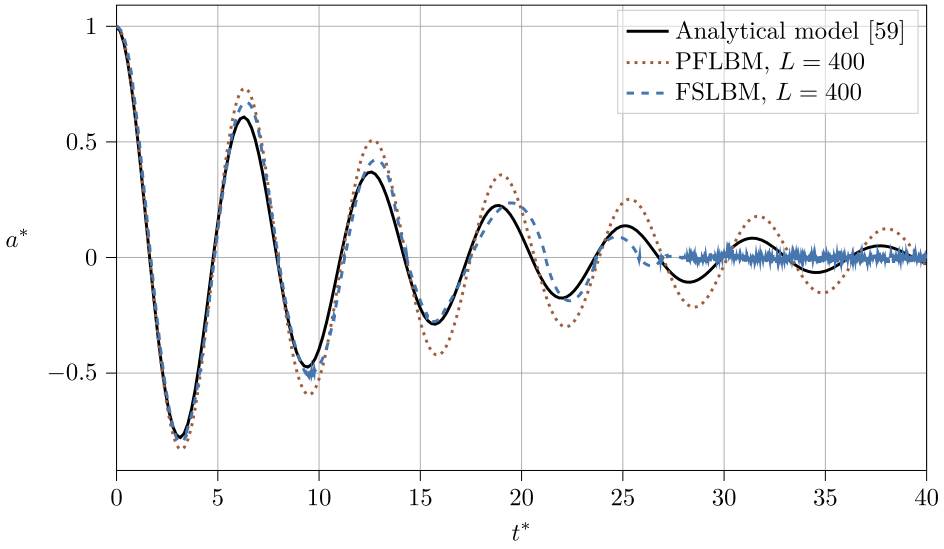


Fig. 4. Amplitude of the gravity wave as simulated by the FSLBM and PFLBM at $L = 400$ in terms of non-dimensional amplitude, $a^*(0, t^*)$, and time, t^*. The PFLBM was able to capture significantly smaller amplitudes.

for the same resolution). However, in this test case, there was only a single fluid and a single gas domain divided by one interface with little curvature. Therefore, the width of PFLBM's diffuse interface was less significant here and did not enforce a highly resolved computational grid. While this is representative for a variety of applications, it is not for many others in which the minimal diffuse interface width imposes a higher computational resolution. It must be also noted that the size of the amplitude was chosen for consistency of the analytical model. This highlights a limit case for the FSLBM where difficulties arise due to only small surface movement. In this particular case where the surface oscillates back and forth around the same lattice cells, the amplitude of the oscillation can only be sufficiently resolved when cell conversions are triggered, i.e., when the surface movement extends beyond a single layer of cells. In other setups where there is persistent directional movement of the surface, this is not a problem and the surface position is resolved well anywhere between lattice cells.

3.1.2. Capillary wave

In contrast to the gravity wave, the fluid dynamics of the capillary wave are purely dominated by surface tension forces, while gravitational forces are neglected.

Simulation setup The simulation setup was equivalent to the one of the gravity wave in Section 3.1.1. As for the gravity wave, a standing capillary wave evolves, oscillating around a liquid depth, d, because of surface tension forces. The decay of the wave is again caused by energy dissipation due to viscous friction. While the definition of Re in Equation (32) is also used for the capillary wave, the angular frequency of the wave is given by,

$$\omega_0 = \sqrt{\frac{\sigma k^3}{\rho_H + \rho_L}}. \tag{36}$$

Here, it can be seen that it is now defined with the surface tension, σ, and the densities of the heavy, ρ_H, and light phase, ρ_L. Except for hydrostatic pressure, which is not present due to the absence of gravity, the simulation parameters and evaluation procedure were identical to those presented in Section 3.1.1.

All simulations were performed with $\mathrm{Re} = 10$ and $L \in \{50, 100, 200, 400, 800\}$ for the FSLBM and $L \in \{50, 100, 200, 400\}$ for the PFLBM. In the latter, the density ratio, $\tilde{\rho} = 1000$, and kinematic viscosity ratio, $\tilde{\nu} = 1$, mimic a liquid–gas system and conform with the capillary wave's analytical model. The relaxation rate was set to $\omega = 1.8$ and $\omega_\mathrm{H} = 1.99$, for the FSLBM and for the heavy phase in the PFLBM, respectively. As in Section 3.1.1, the mobility, $M = 0.02$, and interface width, $\xi = 5$, were used in the PFLBM.

Analytical model Prosperetti [61] presented an analytical model for small-amplitude capillary waves in viscous fluids. The model assumes that there is either a single fluid with a free-surface ($\rho_\mathrm{L} = 0$, $\mu_\mathrm{L} = 0$) or two fluids with equal kinematic viscosity such that $\tilde{\nu} = 1$. It is derived from the linearized Navier–Stokes equations and therefore only valid in the limit of infinitesimally small wave amplitudes.

Assuming no gravitational forces and no initial velocity, the capillary wave amplitude, $a(t)$, with respect to time, t, is described by,

182

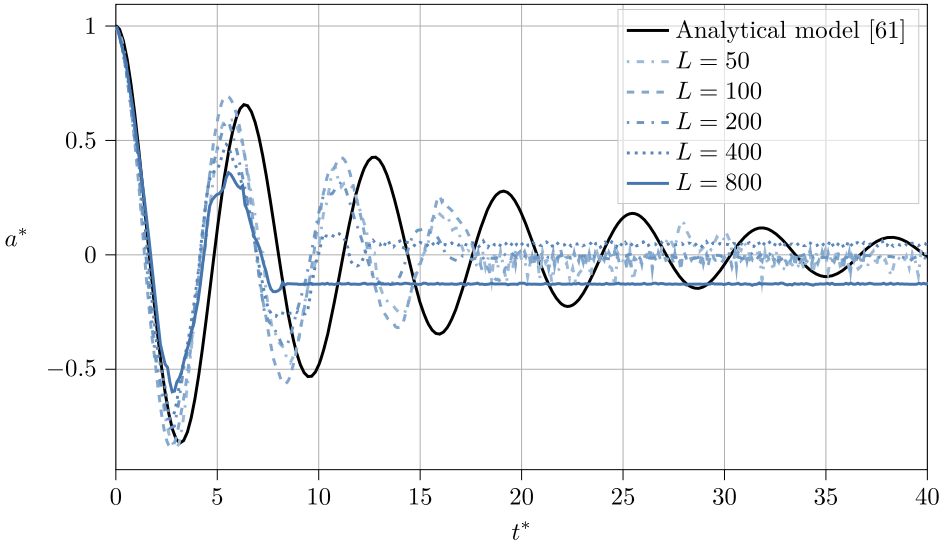


Fig. 5. Capillary wave as simulated by the FSLBM with $L \in \{50, 100, 200, 400, 800\}$ in terms of non-dimensional amplitude, $a^*(t^*)$, and time, t^*. The FSLBM did not converge with higher resolution due to deficiencies in all investigated curvature computation models.

$$a(t) = \frac{4(1-4\beta)\nu^2k^4}{8(1-4\beta)\nu^2k^4+\omega_0^2}a_0\mathrm{erfc}\sqrt{\nu k^2 t} + \sum_{i=1}^{4}\frac{z_i}{Z_i}\left(\frac{w_0^2 a_0}{z_i^2-\nu k^2}\right)\cdot\exp\left(\left(z_i^2-\nu k^2\right)t\right)\cdot\mathrm{erfc}\left(z_i\sqrt{t}\right), \tag{37}$$

where z_i are the roots of the polynomial,

$$z^4 - 4\beta\left(k^2\nu\right)^{\frac{1}{2}}z^3 + 2(1-6\beta)k^2\nu z^2 + 4(1-3\beta)\left(k^2\nu\right)^{\frac{3}{2}}z + (1-4\beta)\nu^2k^4 + \omega_0^2 = 0, \tag{38}$$

and Z_i are computed by circular permutation of the index i in z_i,

$$Z_i = \prod_{1\le j\le 4,\, j\neq i}\left(z_j - z_i\right). \tag{39}$$

The expression $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$ is the complementary error function and β is a dimensionless parameter defined by,

$$\beta = \frac{\rho_\mathrm{L}\rho_\mathrm{H}}{(\rho_\mathrm{L}+\rho_\mathrm{H})^2}. \tag{40}$$

The analytical model is only applicable for small amplitudes such that a correction factor was proposed extending the validity of the model to amplitudes of up to $a_0 \lesssim 0.1L$ [62]. For $a_0 = 0.01L$ as chosen here, this correction factor is only 1.0023 and it can be assumed that the original analytical model is valid to be used as a reference in this study.

Results and discussion As illustrated in Fig. 5, the simulations of the FSLBM did not converge with increasing resolution of the computational grid. The only difference between the gravity wave test case and the capillary wave test case is the driving force, which is a body force in the former and the surface tension in the latter. In the gravity wave test case, the FSLBM simulation converged and the results agreed well with the analytical model. This suggests the potential existence of errors in the surface tension model used within the FSLBM. There, surface tension forces are incorporated by the Laplace pressure, p^L, from Equation (14), with the interface curvature κ being the only non-constant parameter in the equation. Therefore, it is apparent that the diverging behavior must be caused by a diverging interface curvature computation.

As described in Section 2.2, the simulations and results shown here are based on a curvature computation using the finite difference method (FDM) [50]. A similar result was obtained when computing the interface curvature using a local triangulation model and the algorithm from Taubin [63] in waLBerla, as suggested by Reference [49]. This is in agreement with Reference [50], where both approaches were found to diverge with increasing resolution when computing the curvature of a resting spherical gas bubble. On the other hand, a curvature model based on local triangulation and a least squares fit optimization (LSQR) was found to be second-order convergent in the same test case [50]. However, using a similar LSQR approach [40] in FluidX3D, also no convergent behavior could be obtained in the capillary wave test case at the largest resolutions.

It is vital to remark that the absolute value of the curvature decreases when increasing the resolution. With the parametrization chosen here, also the absolute numerical value of the surface tension decreases with increasing L. Therefore, although the LSQR curvature model converges, the model's constant error in curvature has increasingly more influence at higher resolution.

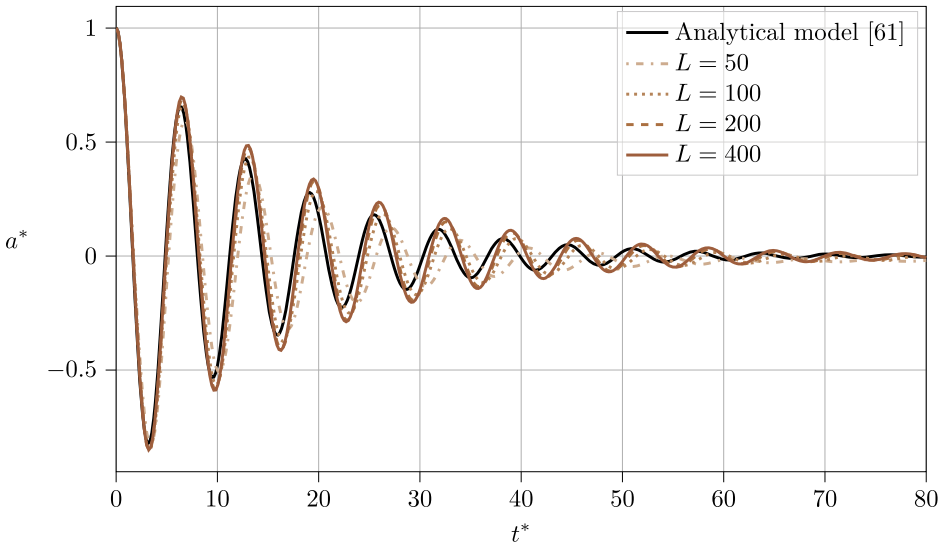


Fig. 6. Capillary wave as simulated by the PFLBM with $L \in \{50, 100, 200, 400\}$ in terms of non-dimensional amplitude, $a^*(t^*)$, and time, t^*. The model captured the wave's phase and damping with reasonable accuracy.

The capillary wave has been previously simulated and compared to a different analytical model [64], where gravitational forces are also considered [7]. There, simulations were performed with the curvature computation using the algorithm of Taubin but the authors did not present a convergence study. Using the same capillary wave setup and resolution as in Reference [7], a moderate agreement with the analytical model was observed with the FSLBM implementations used in this work. However, in a convergence study, again the FSLBM did not converge to the analytical model regardless of the curvature computation model used. In the work of Körner et al. [8], the FSLBM has also been used to simulate a capillary wave and found to agree well with the analytical solution. The authors did not present a convergence study but verbally argued that the error decreases linearly with increasing resolution. The curvature model used there is based on the two-dimensional template–sphere method [65] that uses a neighborhood of 25 cells to compute the curvature. However, the implementations presented here are explicitly targeted at parallel computing environments, in which such a calculation is not feasible. To maintain reasonable parallel efficiency, only information from nearest neighbor cells is desired for curvature computation.

In contrast, as illustrated in Fig. 6, the PFLBM converged well towards the analytical solution but slightly underestimated the analytical model's damping with the parameters from this study. A comparable capillary wave test case has been simulated with the PFLBM in Reference [34]. However, compared to the parameters chosen here, the initial amplitude and Reynolds numbers were significantly smaller in Reference [34], leading to a more accurate prediction of the damping.

It can be concluded that special attention must be paid when simulating surface tension dominated flows with very low curvature with the FSLBM. While the capillary wave resembles an extreme case with small amplitudes leading to infinitesimal values of absolute curvature, other test cases with major surface tension influence have been simulated with good accuracy with the FSLBM [4,40]. On the other hand, the PFLBM accurately simulates this test case and as in Section 3.1.1, it has to be pointed out explicitly that the PFLBM is capable of also simulating very small amplitudes.

3.2. Buoyancy driven flows

This section presents numerical simulations of buoyancy driven flows. The first test case is an unconfined flow, where a single gas bubble rises in liquid. In the second test case, a large gas bubble rises in liquid contained in a cylindrical tube. This large gas bubble in the confined, buoyancy driven flow is commonly referred to as Taylor bubble.

3.2.1. Rising bubble

The more practically oriented third test case is an unconfined buoyancy driven flow, i.e., the rise of a single gas bubble in a liquid column. In order to correctly simulate the bubble shape and rise velocity, the balance between buoyancy, viscous, and surface tension forces must be correct. As there are no analytical models available predicting a rising bubble's shape and velocity, the comparison is drawn using experimental data from Bhaga and Weber [66].

Simulation setup As shown in Fig. 7, a gas bubble was initialized as a sphere of diameter, D, centered at $(4D, 4D, 1D)$ in a computational domain of size $8D \times 8D \times 20D$ (x-, y-, z-direction). Gravity was applied in the negative z-direction causing the bubble to rise due to buoyancy. The top and bottom walls (in z-direction) were realized as no-slip boundaries, while the side walls of the domain were periodic. The size of the domain was tested, and determined to be sufficiently large so as not to influence the results of the simulations. Hydrostatic pressure was initialized such that the reference density, $\rho_0 = 1$, was positioned at $10D$ in the z-direction.

184

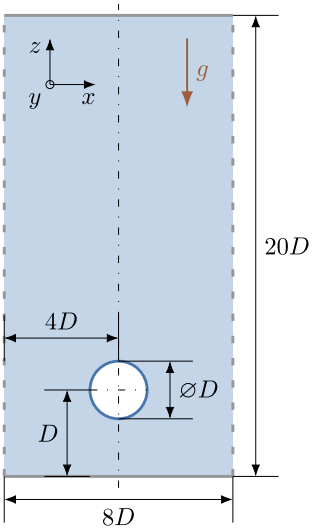


Fig. 7. Simluation setup of the three-dimensional rising bubble test case with initial bubble diameter, D, and gravitational acceleration, g. The domain's side walls in x-direction are periodic, whereas at the top and bottom walls in z-direction, no-slip boundary conditions are applied.

Table 1
The rising bubble test cases are defined by the Bond number, Bo, and the Morton number, Mo. The FSLBM's relaxation rate, ω, and the PFLBM's relaxation rate in the heavy phase, ω_{H}, are kept constant for all resolutions to achieve diffusive scaling.

Case	1	2	3	4
Bo	32.2	115	243	339
Mo	$8.2 \cdot 10^{-4}$	$4.63 \cdot 10^{-3}$	266	43.1
ω (FSLBM)	1.95	1.95	1.65	1.8
ω_{H} (PFLBM)	1.97	1.98	1.83	1.92

The rise of a single gas bubble in liquid is characterized by the Morton number,

$$\mathrm{Mo} = \frac{g\mu^4}{\rho\sigma^3}, \tag{41}$$

which describes the ratio of viscous to surface tension forces, and the Bond number,

$$\mathrm{Bo} = \frac{gD^2\rho}{\sigma}, \tag{42}$$

which describes the ratio of gravitational forces, i.e., buoyancy, to surface tension forces. It is commonly also referred to as the Eötvös number (Eo). The definitions of these dimensionless numbers are taken from Reference [66] and the density, ρ, and dynamic viscosity, μ, refer to the heavier fluid.

The bubble shape and position in terms of its center of mass, were monitored at every reference time interval,

$$t^* = t\sqrt{\frac{g}{D}}. \tag{43}$$

From the bubble position in the z-direction at time, $t^* = 5$, and $t^* = 10$, the rise velocity u and Reynolds number,

$$\mathrm{Re} = \frac{\rho D u}{\mu}, \tag{44}$$

were evaluated. The simulations were stopped at $t^* = 10$. The bubble shape and the Reynolds number were then compared with experimental observations from Reference [66].

The simulations were carried out with $D \in \{8, 16, 32, 64\}$ for both models. Additionally, as in Reference [29], a fixed mobility, $M = 0.02$, and interface width, $\xi = 5$, were used for the PFLBM. Furthermore, to close the system parameters, the density of the liquid phase was specified as $\rho_H = 1$, and the density ratio, $\tilde{\rho} = 1000$, and dynamic viscosity ratio, $\tilde{\mu} = \mu_{\mathrm{H}/\mu_{\mathrm{L}}} = 100$, were chosen to mimic an air–water system. For the FSLBM, the initial pressure of the bubble was set to the reference pressure, $p_0 = \rho_0 c_s^2 = 1/3$, with reference density, $\rho_0 = 1$. The dimensionless numbers that define the four cases tested, and the employed LBM relaxation rates are listed in Table 1. Hydrostatic pressure was initialized in the domain such that the pressure is equivalent to the LBM reference density, $\rho_0 = 1$, in the center of the domain in z-direction.

185

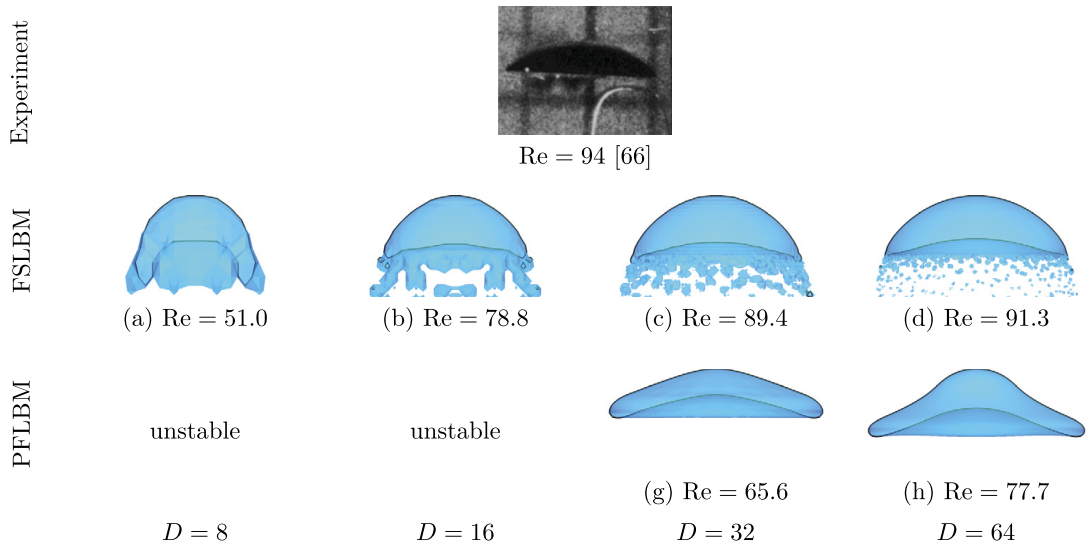


Fig. 8. Simulated bubble shape and Reynolds number, Re, at time, $t^* = 10$, for case 2 in Table 1 with Bo = 115 and Mo = $4.63 \cdot 10^{-3}$. Different computational resolutions according to the initial bubble diameter, D, are shown. The solid black lines illustrate the bubble's contour in the center cross-section with normal in the x-direction. The photograph of the laboratory experiment was reprinted from Reference [66] with the permission of Cambridge University Press.

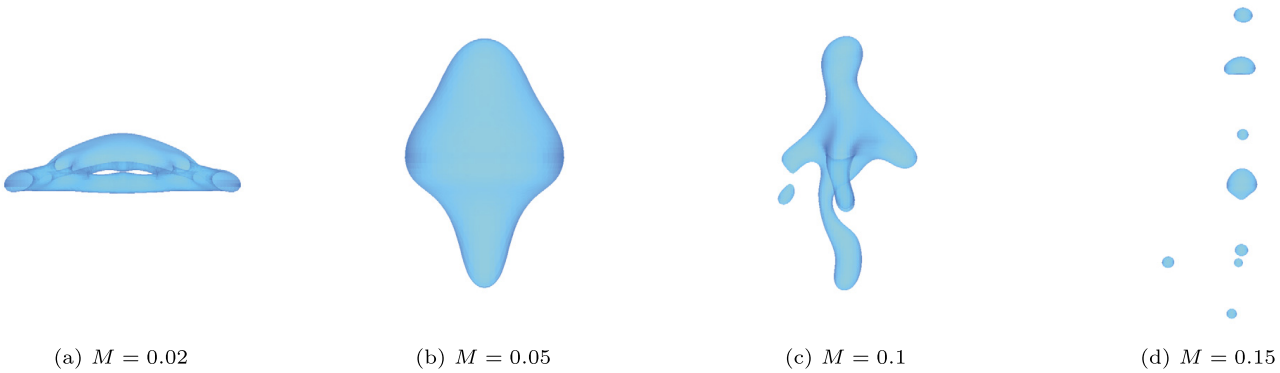


Fig. 9. Shape of the rising bubble as simulated with the PFLBM at time, $t^* = 12$, for case 2 in Table 1 with Bo = 115 and Mo = $4.63 \cdot 10^{-3}$. The computational resolution was set according to $D = 32$. Larger values for the mobility, M, lead to non-physical bubble shapes, i.e., break-up.

Results and discussion The simulated bubble shapes at $t^* = 10$ are presented in Fig. 8 and in the Appendix in Figs. 18 to 20. It can be seen that both models converged to the Reynolds numbers reported in the experiments in Reference [66]. The FSLBM simulated the rising bubble with reasonable accuracy for computational resolutions of $D \geq 16$. Although surface tension forces are not fully governing the rising bubble test cases, they still significantly determine the bubble shape and rise velocity here. In contrast to the capillary wave test case in Section 3.1.2, the FSLBM showed consistent convergence with increasing computational resolution. This emphasizes that the FSLBM can still be applicable in problems where surface tension is non-negligible. However, the FSLBM predicted the detachment of several satellite bubbles in cases 2 to 4 that can not be observed in the photographs of the experiments. Qualitatively similar bubble shapes were also predicted by the FSLBM with the LSQR curvature model, as shown in the Appendix in Figs. 21 to 24.

In contrast, for the PFLBM, it was not possible to obtain results for resolutions of $D < 32$. Furthermore, for case 2, neither the bubble shape nor the Reynolds number was predicted reasonably well, regardless of the resolution, as illustrated in Fig. 8. Moreover, when increasing the simulation run time, non-physical bubble shapes and collapse were also observed with the PFLBM. Fig. 9a shows this behavior for case 2 with $D = 32$ in which the skirted bubble film ruptures at $t^* > 10$. With increased computational resolution, this effect occurred at later t^*.

It was shown in the literature, that phase-field models are sensitive to the choice of the mobility parameter, M [67]. However, in general there appears to be no robust solution for how this parameter should be specified for arbitrary cases. In a study performed here, as depicted in Fig. 9, it was observed that larger values of M seem to boost such non-physical effects. On the other hand, with $M < 0.02$, instabilities were observed, as the relaxation time, τ_ϕ, in Equation (21) decreases and approaches its lower stability limits. These instabilities occurred even when using the weighted MRT scheme, which is generally known for good stability properties [32]. A rigorous study of this behavior is outside the scope of this work, but is proposed for future investigation.

186

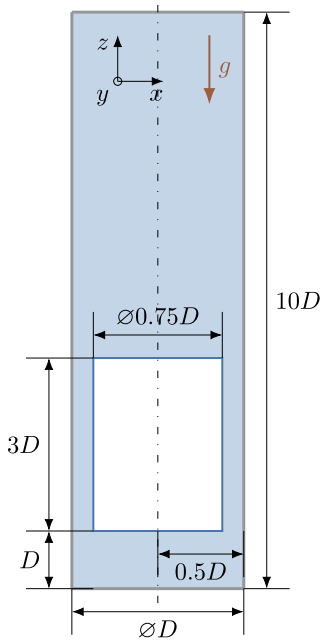


Fig. 10. Simulation setup of the three-dimensional Taylor bubble test case, with an initially cylindrical gas bubble in a cylindrical tube of diameter, D, and gravitational acceleration, g. No-slip boundary conditions are applied at the tube's walls and at the domain's top and bottom walls in z-direction.

The test cases used in this study have also been simulated in two dimensions by Kumar et al. [34], using the PFLBM. To check the implementations' validity, these two-dimensional simulations were also performed here, agreeing with Reference [34] and without becoming unstable or leading to implausible bubble shapes.

While the reason for these instabilities is not yet clear, it must be pointed out that the expected bubble shapes consist of only a thin film of gas. In the literature, similar circular destabilization of films with the PFLBM could be observed in other test cases, however, often of a thin liquid rather than gas film [37].

3.2.2. Taylor bubble

The fourth test case is a buoyancy driven confined flow, a large gas bubble rising through stagnant liquid in a cylindrical tube. During the bubble's rise, it takes an elongated shape with a rounded leading edge. Its length is several times the tube's diameter and it is commonly referred to as Taylor bubble [68,69].

Setup The simulation setup chosen here is similar to the one in Reference [22], conforming to the experiments in Reference [70]. As illustrated in Fig. 10, in a computational domain of size $1D \times 1D \times 10D$ (x-, y-, z-direction), the domain walls formed a cylindrical tube of diameter, D, pointing in z-direction. A gas bubble was initialized as cylinder with diameter, $0.75D$, and length, $3D$, oriented concentrically to the boundary tube. The gas bubble's bottom was located at D in positive z-direction. The rest of the domain was filled with a stagnant liquid. According to the gravitational acceleration, g, the liquid was initialized with hydrostatic pressure such that the reference pressure, $p_0 = \rho_0 c_s^2 = 1/3$, was set at $5D$ in z-direction. As in Section 3.2.1, the Morton number, Mo, Bond number, Bo, and the reference time, t^*, characterize the system. Here, the tube diameter, D, was used as characteristic length [70] in these non-dimensional numbers.

The experiments in Reference [70] were conducted with Bo= 100, Mo=0.015, and olive oil. Following Reference [22], it is assumed that the density and viscosity of the air injected into the oil was $\rho^{\mathrm{SI}} = 1.225\,\mathrm{kg/m^3}$ and $\mu^{\mathrm{SI}} = 1.983 \cdot 10^5\,\mathrm{kg/(m{\cdot}s)}$, respectively. Therefore, the density ratio, $\tilde{\rho} = 744$, and the dynamic viscosity ratio, $\tilde{\mu} = 4236$, were used. As in Section 3.2.1, for the FSLBM, the initial pressure of the bubble was set to the reference pressure, $p_0 = \rho_0 c_s^2 = 1/3$. The simulations were performed with computational resolutions according to the tube diameter, $D \in \{16, 32, 64, 128\}$. However, in the PFLBM, simulating a tube diameter of $D \leq 32$ was not possible, as the diffuse interface led to non-physical wall interactions with the interfacial region. Based on the investigations from Section 3.2.1, the mobility parameter was set to the lowest value at which the simulations at any tested resolution were stable, namely $M = 0.08$. The interface width was chosen as $\xi = 3$. For all simulations, the relaxation rate was set to $\omega = 1.8$ in the FSLBM, and $\omega_{\mathrm{H}} = 1.76$ in the heavier phase of the PFLBM's hydrodynamic LBM step.

Results and discussion Fig. 11 compares the simulated Taylor bubble's shape at different computational resolutions at time $t^* = 15$ with the experimental measurement [70]. To ease comparison, the axial location, $z^* = z/D$ and radial location, $r^* = r/(0.5D)$ are non-dimensionalized. Additionally, an axial shift is employed as to set $z^* = 0$ at $r^* = 0$ for the bubble's front and tail individually. Both models converged well, but showed minor deviations to the experimental data from Reference [70]. The shape of the front of the bubble was predicted with reasonable accuracy at all computational resolutions

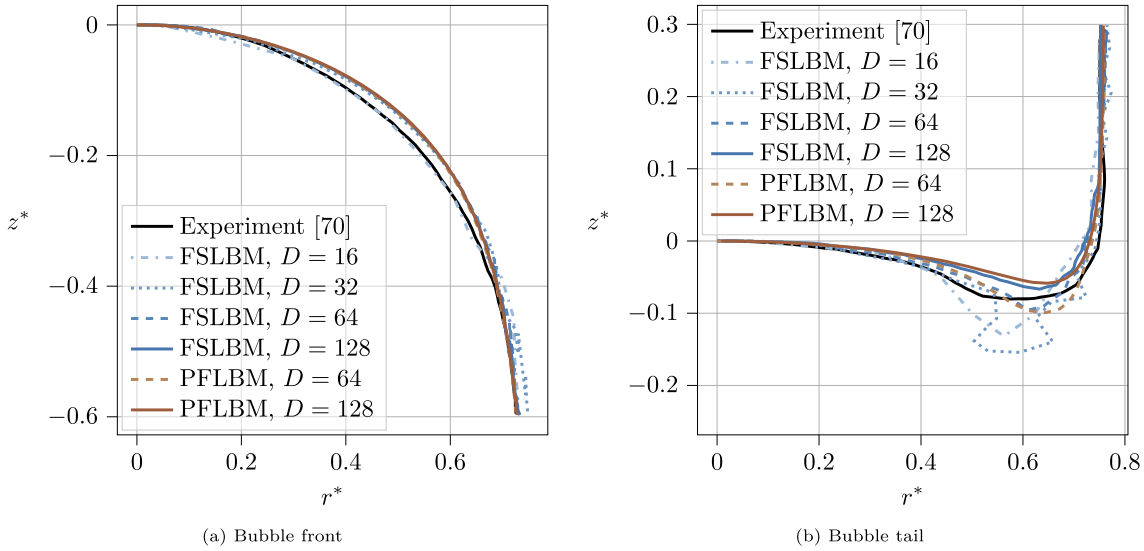


Fig. 11. Shape of the front and tail of the simulated Taylor bubble at different computational resolutions, specified by the tube diameter, D. The comparison with experimental data [70] is drawn in terms of the non-dimensionalized axial location, $z^* = z/D$, and radial location, $r^* = r/(0.5D)$ at time, $t^* = 15$.

Table 2
Reynolds number, Re, of the simulated Taylor bubble for different computational resolutions as specified by the tube diameter, D. The bubble's rise velocity, as used in Re, was computed from the Taylor bubble's locations in axial direction at time $t^* = 10$ and $t^* = 15$.

D	16	32	64	128
$\mathrm{Re}_{\mathrm{FSLBM}}$	22.15	24.12	25.35	25.89
$\mathrm{Re}_{\mathrm{PFLBM}}$	unstable	unstable	26.83	27.12
$\mathrm{Re}_{\mathrm{Experiment}}$ [70]		27		

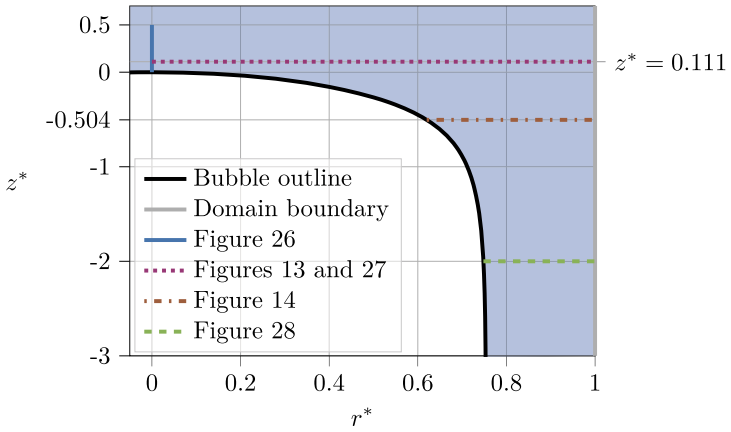


Fig. 12. Definition of the locations at the Taylor bubble's front, where the velocity profiles are evaluated in the subsequent figures. The monitored lines are expressed in terms of the non-dimensionalized axial location, $z^* = z/D$, and radial location, $r^* = r/(0.5D)$. The test case is radially symmetric such that the evaluation can be performed at an arbitrary cross-section.

tested. However, at the tail of the bubble, a resolution of $D \geq 64$ was required for the FSLBM to capture the interface contour moderately well. As also observed in Section 3.2.1, satellite bubbles separated from the main bubble in the case of the FSLBM, as shown in the Appendix in Fig. 25. In contrast to the observations for the rising bubble test, this effect vanished with increasing computational resolution.

In Table 2, the simulated Reynolds number, Re, as defined in Equation (44), is shown. The tube diameter, D, and the Taylor bubble's rise velocity, U, are used as characteristic quantities to determine Re. The rise velocity, U, was computed by the bubble's center of mass location in z-direction at time, $t^* = 10$, and $t^* = 15$. In comparison to the PFLBM, which agreed well with the experimental measurement [70], the FSLBM showed larger deviations. This was even more pronounced at lower computational resolutions, where it could capture the bubble's axial movement only moderately well.

188

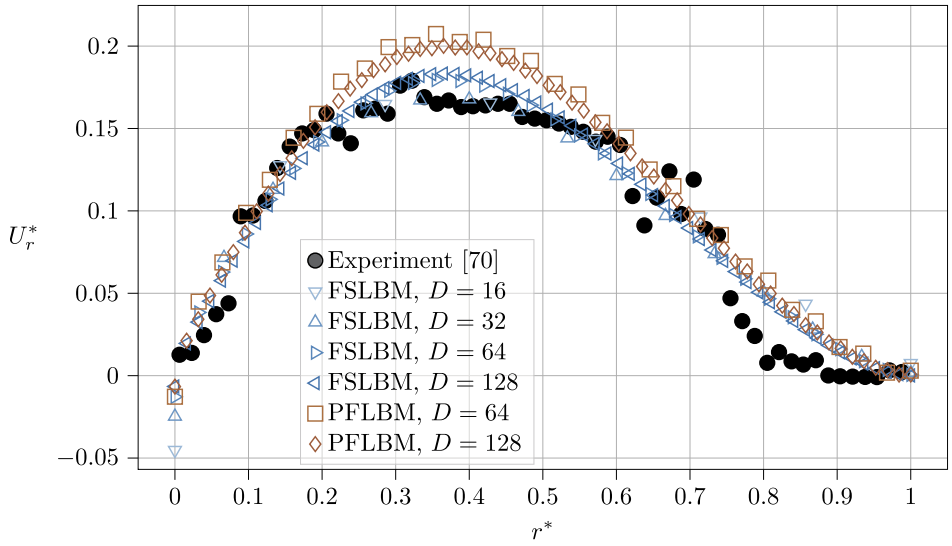


Fig. 13. Simulated non-dimensionalized radial velocity, U_r^*, along a radial line positioned at $0.111D$ in front of the Taylor bubble (see Fig. 12), with tube diameter, D. The comparison with experimental data [70] is drawn in terms of the non-dimensionalized radial location, $r^* = r/(0.5D)$, at time, $t^* = 15$.

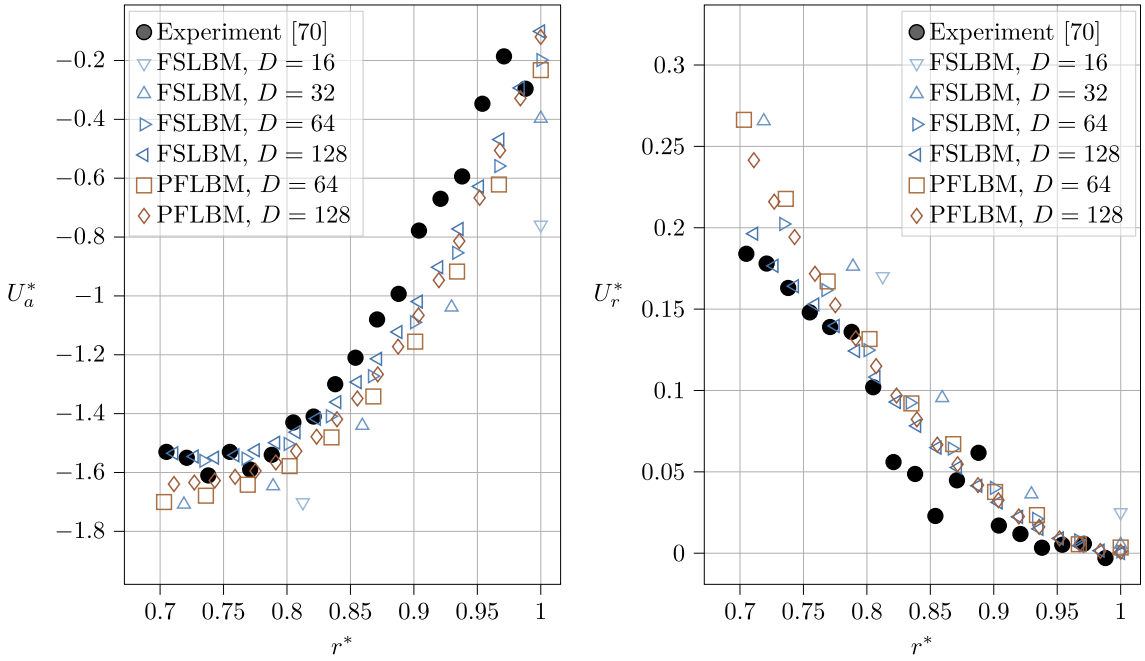


Fig. 14. Simulated non-dimensionalized axial velocity, U_a^*, and radial velocity, U_r^*, along a radial line positioned at $0.504D$ behind the Taylor bubble's front (see Fig. 12), with tube diameter D. The comparison with experimental data [70] is drawn in terms of the non-dimensionalized radial location, $r^* = r/(0.5D)$, at time, $t^* = 15$.

At the locations specified in Fig. 12, the flow field around the bubble was evaluated. The non-dimensionalized axial fluid velocity, $U_a^* = U_a/U$, along a central axial line of length $0.5D$ in front of the bubble is presented in the Appendix in Fig. 26. Both models converged and agreed well with the experimental data [70]. On the other hand, at a radial line situated at $0.111D$ in front of the bubble, the non-dimensionalized radial fluid velocity $U_r^* = U_r/U$ (see Fig. 13), and axial velocity (see Appendix, Fig. 27) showed larger deviations when using the PFLBM, favoring the FSLBM at higher computational resolution. A similar observation could be made at a radial line at $0.504D$ behind the bubble's front, as visualized in Fig. 14. Fig. 28 illustrates that at a radial line located $2D$ behind the front of the bubble, the predicted axial velocity by both models agreed reasonably well with the experimental data.

3.3. Dynamic coalescence – crown splash

Understanding the dynamics of splashing during liquid drop impacts has many implications, including aerosol production [71], erosion processes [72] and microplastic transfer in the environment [4]. In this study, two drop impact test cases are simulated for which photographs of the laboratory experiments are available in the literature [73]. Both test cases have

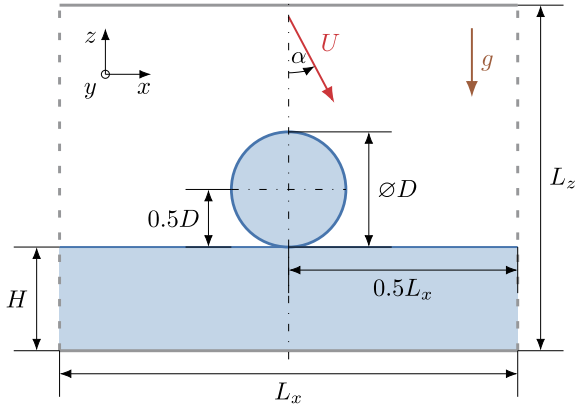


Fig. 15. Simulation setup of the drop impact test cases. In a domain of size $L_x \times L_y \times L_z$, a spherical drop of liquid with diameter, D, is initialized right above the surface of a liquid pool of height, H. While the gravitational acceleration, g, points in negative z-direction, the droplet is initialized with impact velocity, U, acting at an angle, α. The domain's side walls in x- and y-direction are periodic, whereas the domain's top and bottom walls in z-direction are set to no-slip.

already been simulated using the FSLBM with LSQR curvature computation model [4]. Due to the absence of quantitative experimental data, the comparisons with reference data can only be made qualitatively.

3.3.1. Vertical drop impact

In the fifth test case, a vertical drop impacting on a thin film of liquid was simulated and compared with experimental data [73].

Setup As shown in Fig. 15, a thin liquid film of height, $H = 0.5D$, was initialized in a computational domain of size, $L_x \times L_y \times L_z$ (x-, y-, z-direction) with $L_x = L_y = 10D$ and $L_z = 5D$. At the pool's surface, a spherical droplet with diameter, D, was initialized with an impact velocity, U, in the negative z-direction with $\alpha = 0°$, leading to a vertical impact. The domain's side walls in the x- and y-direction were periodic, whereas there were no-slip boundary conditions at the domain's top- and bottom walls. Conforming with the gravitational acceleration, g, hydrostatic pressure was initialized such that the reference density was $\rho_0 = 1$ at the surface of the pool.

The drop impact is described by the Weber number

$$\mathrm{We} = \frac{\rho U^2 D}{\sigma}, \tag{45}$$

which relates inertial and surface tension forces, and by the Ohnesorge number,

$$\mathrm{Oh} = \frac{\mu}{\sqrt{\sigma \rho D}}, \tag{46}$$

which is defined by the relation of viscous to inertial and surface tension forces. The drop diameter, D, the Bond number, Bo (see Equation (42)), and reference time, $t^* = t\,U/D$, close the definition of the system. As found in Reference [4], the simulation results must be offset by $t^* = 0.16$ to synchronize the first photograph of the laboratory experiment with the simulation setup chosen in this study.

In the experiments of Reference [73], a 70 % glycerol–water mixture at 23 °C was used with $\rho^{\mathrm{SI}} = 1200\,\mathrm{kg/m^3}$ and $\mu^{\mathrm{SI}} = 0.022\,\mathrm{kg/(m{\cdot}s)}$. The experiment obeyed the non-dimensional numbers, We=2010, and, Oh=0.0384. Assuming $g^{\mathrm{SI}} = 9.81\,\mathrm{m/s^2}$, the system is closed by Bo= 3.18. As in Section 3.3.2, the density ratio is set to $\tilde{\rho} = 1000$ and the dynamic viscosity ratio is set to $\tilde{\mu} = 100$.

The simulations were performed with computational resolutions according to $D \in \{20, 40, 80\}$. The FSLBM's relaxation rate was chosen at $\omega = 1.989$ and the PFLBM's hydrodynamic relaxation rate in the heavy phase was set to $\omega_{\mathrm{H}} = 1.988$. In agreement with the findings from Section 3.2, lower values for the PFLBM's interface width, ξ, and mobility, M, tended to give more physically realistic results, as shown in the Appendix in Fig. 29. Therefore, $\xi = 4$ and $M = 0.03$ were chosen as they are the lowest values that allowed stable simulations for all tested computational resolutions.

Results and discussion In Fig. 16, the crown formation at time, $t^* = 12$, is shown for both models at various computational resolutions. In the Appendix, Figs. 30 and 31 compare the simulated and experimental drop impact dynamically, i.e., with respect to time. While no scale bars for the photograph of the laboratory experiments are available, it can be noted that all simulations converged well with increasing resolution, and the dimensions of the simulated splash crowns agreed with each other. The measured simulated cavity depths and splash crowns' inner diameters are presented in the Appendix in Tables 3 and 4. The FSLBM captured the droplets ejected from the crown qualitatively well, even at low computational resolution. Similar results have also been obtained with the FSLBM and LSQR curvature computation [4]. In contrast, the PFLBM with the parameters chosen here, could not sufficiently predict these droplets.

190

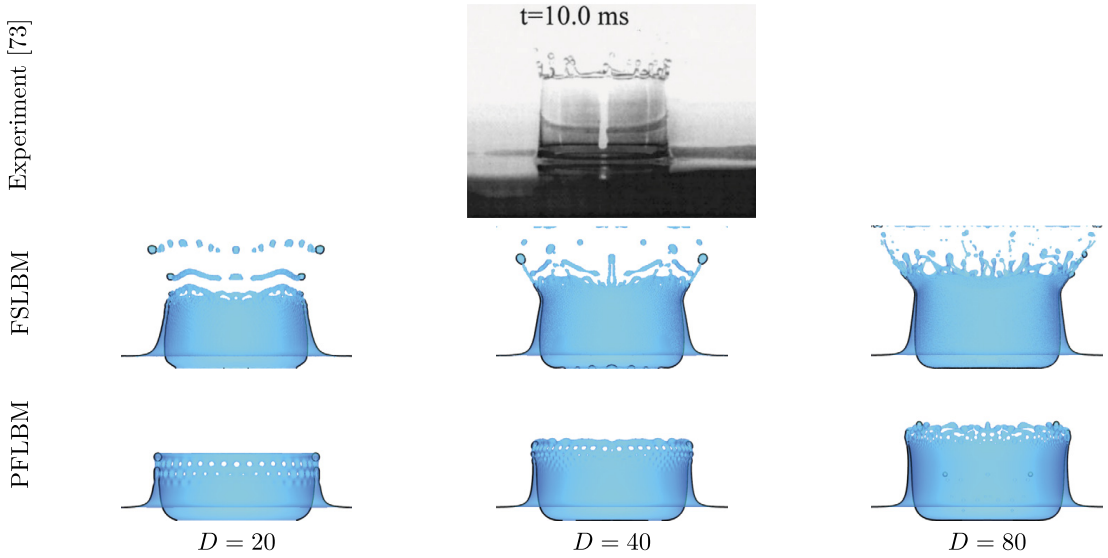


Fig. 16. Simulated vertical drop impact at time, $t^* = 12$, at different computational resolutions defined by the initial drop diameter, D. While the simulation results are true to scale, no scale bar is available for the photograph of the experiment [73]. Therefore, the splash crown's dimension can only be compared between simulations rather than with the experiment. The solid black line illustrates the crown's contour in the center cross-section with normal in the x-direction. The photograph of the laboratory experiment was reprinted from Reference [73] with the permission of AIP Publishing.

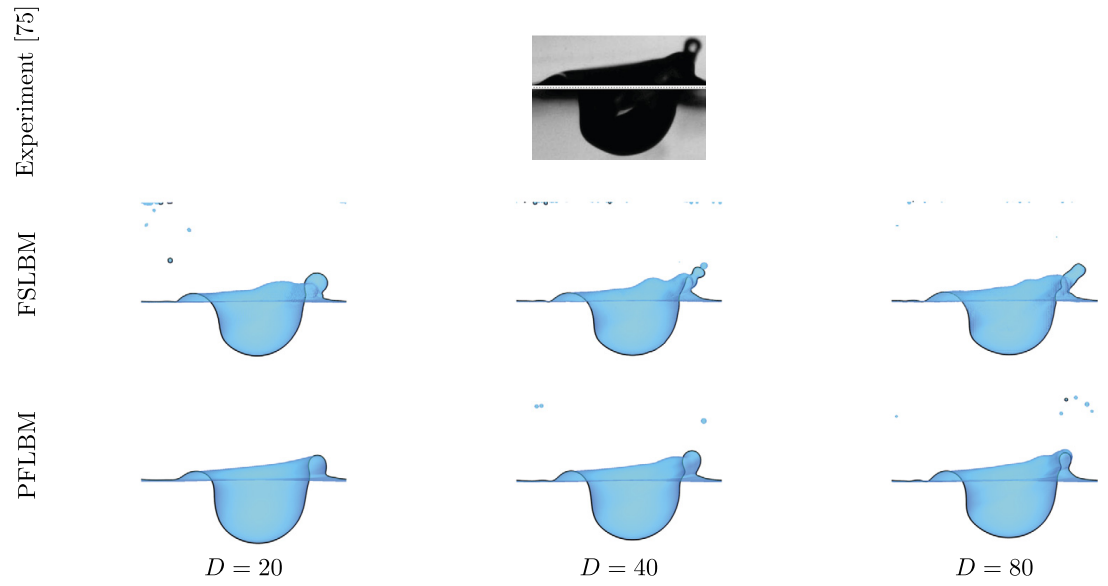


Fig. 17. Simulated oblique drop impact at time, $t^* = 18$, at different computational resolutions defined by the initial drop diameter, D. While the simulation results are true to scale, no scale bar is available for the photograph of the experiment [75]. Therefore, the splash crown's dimension can only be compared between simulations rather than with the experiment. The solid black line illustrates the crown's contour in the center cross-section with normal in the x-direction. The photograph of the laboratory experiment was reprinted from Reference [75] with the permission of the original authors.

It must be emphasized, that the PFLBM is sensitive to the choice of the interface width and mobility parameter (see Appendix, Fig. 29). That is, for consistency reasons, these values were chosen as to be stable with the lowest computational resolution, $D = 20$, and kept constant for higher resolutions. A rigorous study of the individual lower limits of these parameters at each resolution might improve the quality of the results.

3.3.2. Oblique drop impact

In the final test case, an oblique drop impact is simulated as in the experiments of References [74,75].

Setup The setup is similar to Section 3.3.1 and presented in Fig. 15. However, the computational domain is cubical with $L_x = L_y = L_z = 10D$, and the liquid pool is of height, $H = 5D$. The droplet's impact velocity, U, is oriented in an angle, $\alpha = 28.5°$, from negative z-direction.

The experimental investigations were performed with We$= 416.5$, $D^{\mathrm{SI}} = 1.15 \cdot 10^{-4}\,\mathrm{m}$, and liquid water with $\rho^{\mathrm{SI}} = 1000\,\mathrm{kg/m^3}$ and $\sigma^{\mathrm{SI}} = 0.072\,\mathrm{kg/(s^2)}$. Assuming $\mu^{\mathrm{SI}} = 10^{-3}\,\mathrm{kg/(m{\cdot}s)}$ for water at $20\,°\mathrm{C}$ and $g^{\mathrm{SI}} = 9.81\,\mathrm{m/s^2}$, the setup is

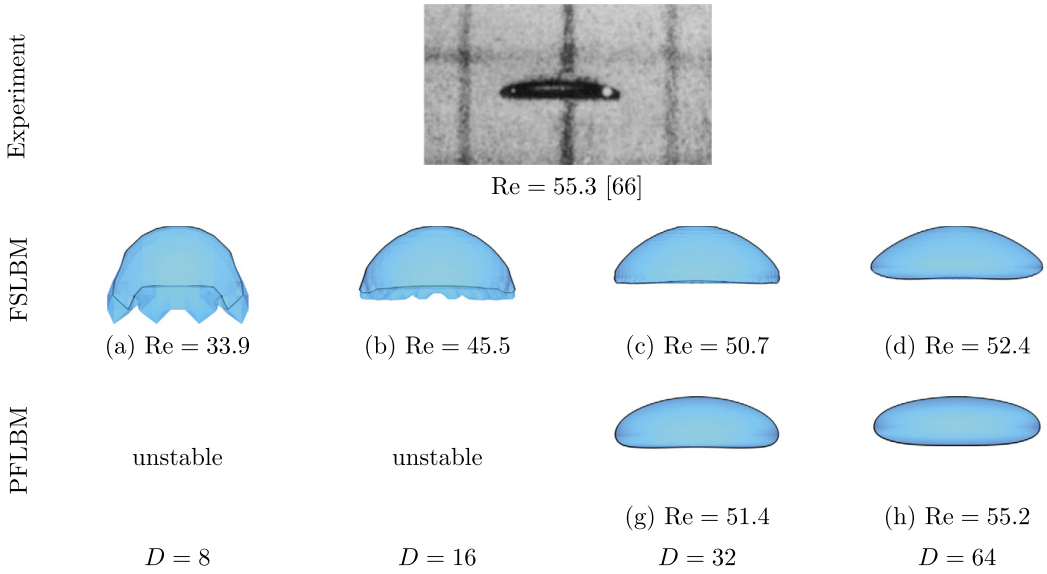


Fig. 18. Simulated bubble shape and Reynolds number, Re, at time, $t^* = 10$, for case 1 in Table 1 with Bo = 32.2 and Mo = $8.2 \cdot 10^{-4}$. Different computational resolutions according to the initial bubble diameter, D, are shown. The solid black lines illustrate the bubble's contour in the center cross-section with normal in the x-direction. The photograph of the laboratory experiment was reprinted from Reference [66] with the permission of Cambridge University Press.

defined by, Oh= 0.011, and, Bo= 0.0018. The density ratio, $\tilde{\rho} = 1000$, and dynamic viscosity ratio, $\tilde{\mu} = 100$, are chosen as to mimic an air–water system [75]. The computational resolution, relaxation rates, and hydrostatic pressure were set as for the vertical drop impact in Section 3.3.1. Here, the lowest interface width and mobility that allowed stable simulations in the PFLBM for all tested resolutions were, $\xi = 4$, and, $M = 0.09$, respectively.

Results and discussion In Fig. 17, the crown formation at time, $t = 18t^*$, is shown for both models at various computational resolutions and are compared with photographs of the laboratory experiments [75]. Additionally, Figs. 32 and 33 in the Appendix show the drop impact as simulated by the FSLBM and PFLBM over time. The FSLBM and PFLBM converged well and the dimensions of the simulated splash crowns agreed well with each other. As for the vertical impact, scale bars for the photograph of the laboratory experiments are missing and no quantitative comparison with reference data could be drawn. Nevertheless, the measured simulated cavity depths and splash crowns' inner diameters are presented in the Appendix in Tables 5 and 6. In contrast to the vertical impact, less droplets were ejected from the crown and the PFLBM agreed qualitatively well at high computational resolution. The FSLBM captured the shape of the drop cavity and splash crown qualitatively well, even with low computational resolution. Similar results have been obtained with the FSLBM and LSQR curvature computation [4].

4. Conclusion

This study has compared two different LBM approaches for simulating flows in which the dynamics of the lighter phase are assumed negligible. After an introduction of the numerical foundation of the FSLBM and PFLBM, both models were applied to a series of benchmark cases and their performance was discussed in terms of their numerical properties and implementation aspects.

The FSLBM ignores fluid flow in the secondary phase and requires much less memory, making it efficient and more applicable to limited-memory hardware. On the other hand, the PFLBM simulates flow in both phases but is well suited for massively parallel computing and can be implemented more easily in a flexible way using code generation technology. A very distinct difference between the two models is their sharp and diffuse interface representation in the FSLBM and PFLBM, respectively. Therefore, six numerical experiments were shown in which the models' accuracy is compared at different resolutions of the computational grid.

While the standing gravity wave was simulated more accurately by the FSLBM, a much higher resolution was required than in the PFLBM to capture the motion of the interface at low amplitude. However, it has to be remarked that this test setup represents a limit case of the FSLBM. In consistency with the analytical solution, the interface motion is limited to only a few LBM cells, even for highly resolved grids.

In the capillary wave test case, the FSLBM diverged with increasing resolution due to deficiencies in all tested approaches for the computation of infinitesimal interface curvature. In contrast, the PFLBM could simulate the capillary wave with reasonable accuracy.

The third and fourth test case featured buoyancy driven flows. That is, an unconfined single rising gas bubble in liquid in four different characteristic parameter sets, and a confined Taylor bubble traversing a cylindrical tube, were simulated.

192

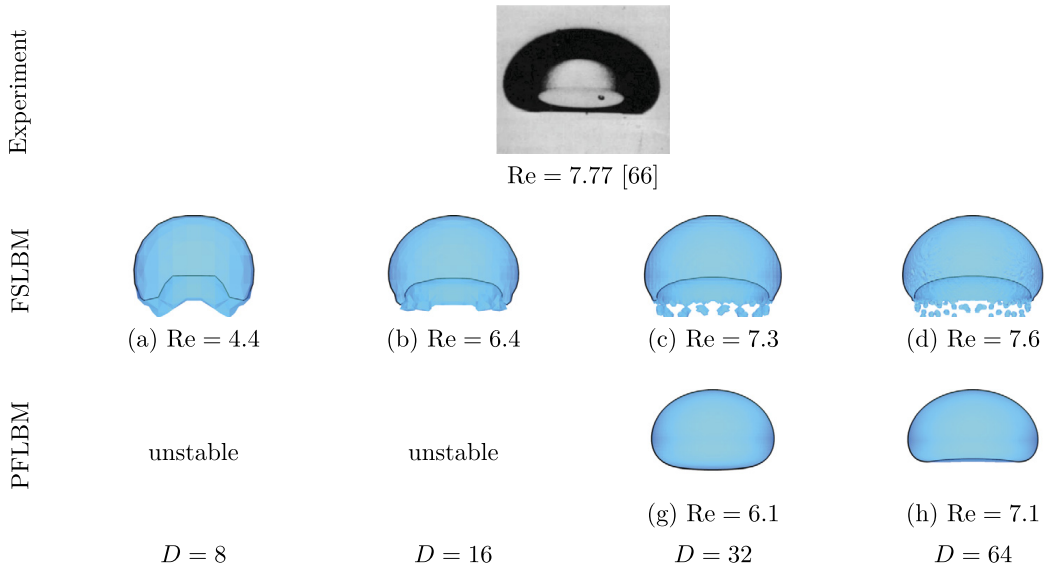


Fig. 19. Simulated bubble shape and Reynolds number, Re, at time, $t^* = 10$, for case 3 in Table 1 with Bo = 243 and Mo = 266. Different computational resolutions according to the initial bubble diameter, D, are shown. The solid black lines illustrate the bubble's contour in the center cross-section with normal in the x-direction. The photograph of the laboratory experiment was reprinted from Reference [66] with the permission of Cambridge University Press.

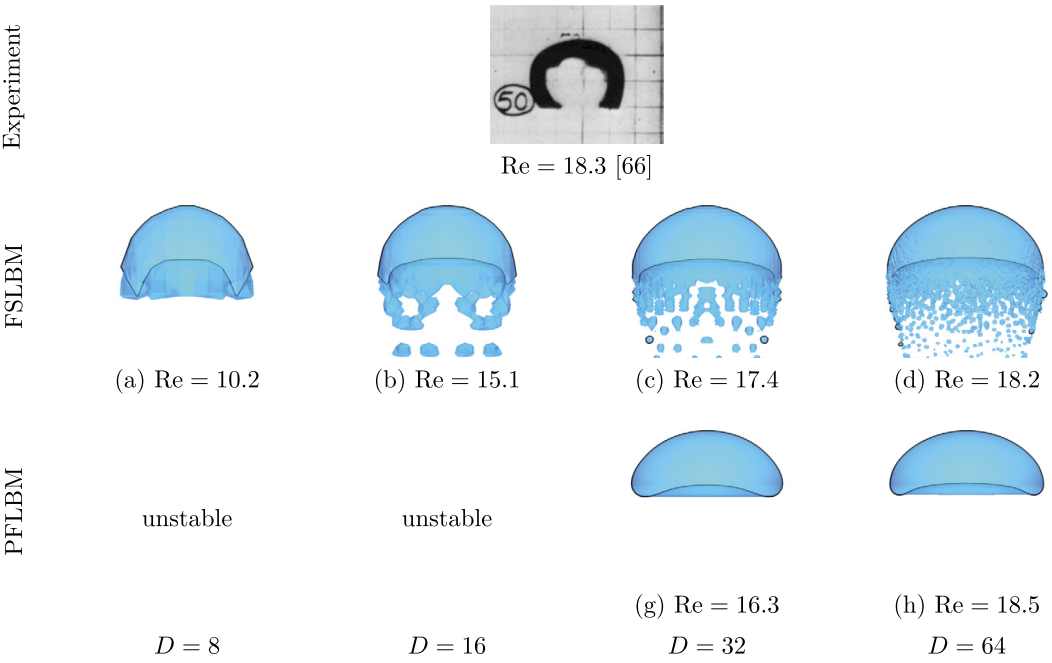


Fig. 20. Simulated bubble shape and Reynolds number, Re, at time, $t^* = 10$, for case 4 in Table 1 with Bo = 339 and Mo = 43.1. Different computational resolutions according to the initial bubble diameter, D, are shown. The solid black lines illustrate the bubble's contour in the center cross-section with normal in the x-direction. The photograph of the laboratory experiment was reprinted from Reference [66] with the permission of Cambridge University Press.

The FSLBM was able to capture the bubble shape and Reynolds number with reasonable accuracy, even with moderate resolution. On the other hand, with the parameters used in this study, the PFLBM required higher resolutions for simulations to be stable. While it predicted bubble shape and accuracy well in the initial phase of the single gas bubble rise, the bubbles tended to evolve into non-physical shapes leading to eventually collapse of the simulation. This observation was made even with the highest computational resolution used in this study. A sensitivity of the chosen mobility on the phase-field was observed. However, the mobility can only be chosen in a certain range to obtain stable simulations. In this range no generally suitable values could be found.

In the fifth and sixth test case, the models' ability to capture dynamic coalescence was validated. To do this, a vertical and oblique drop impact into a pool of liquid were simulated. The FSLBM predicted the shape of the splash crown reasonably well, even with low computational resolution. With sufficiently high computational resolution, the PFLBM was also able to simulate the oblique drop impact with satisfying accuracy. However, for the vertical drop impact, only the FSLBM was able

193

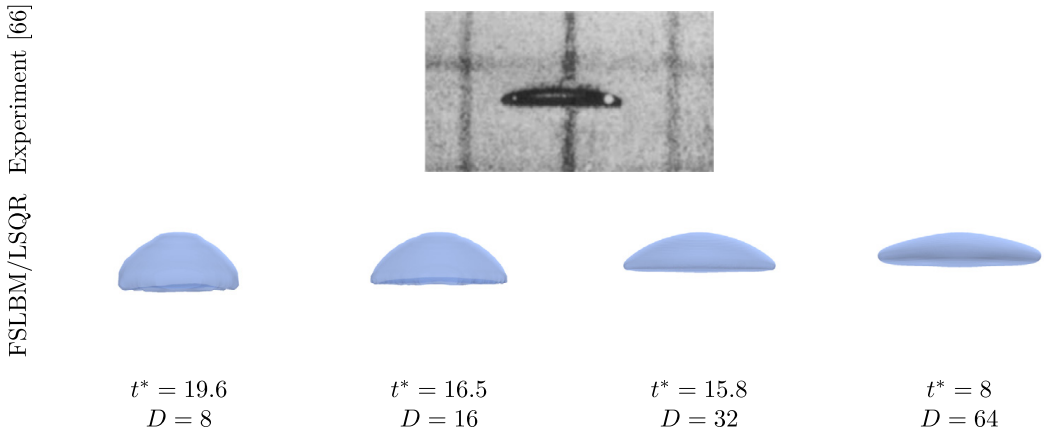


Fig. 21. Simulated bubble shape for case 1 in Table 1 with Bo = 32.2 and Mo = $8.2 \cdot 10^{-4}$. Different computational resolutions according to the initial bubble diameter, D, are shown. The simulations were performed with the FSLBM with LSQR curvature computation model implemented in FluidX3D [4,40]. The photograph of the laboratory experiment was reprinted from Reference [66] with the permission of Cambridge University Press.

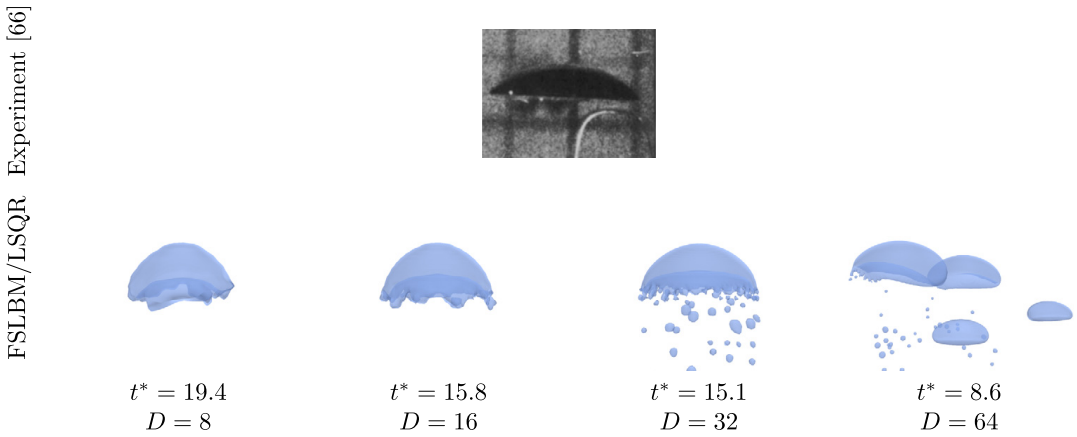


Fig. 22. Simulated bubble shape for case 2 in Table 1 with Bo = 115 and Mo = $4.63 \cdot 10^{-3}$. Different computational resolutions according to the initial bubble diameter, D, are shown. The simulations were performed with the FSLBM with LSQR curvature computation model implemented in FluidX3D [4,40]. In contrast to the experiment, the bubble broke apart into several smaller bubbles in the simulation at a resolution of $D = 64$. The photograph of the laboratory experiment was reprinted from Reference [66] with the permission of Cambridge University Press.

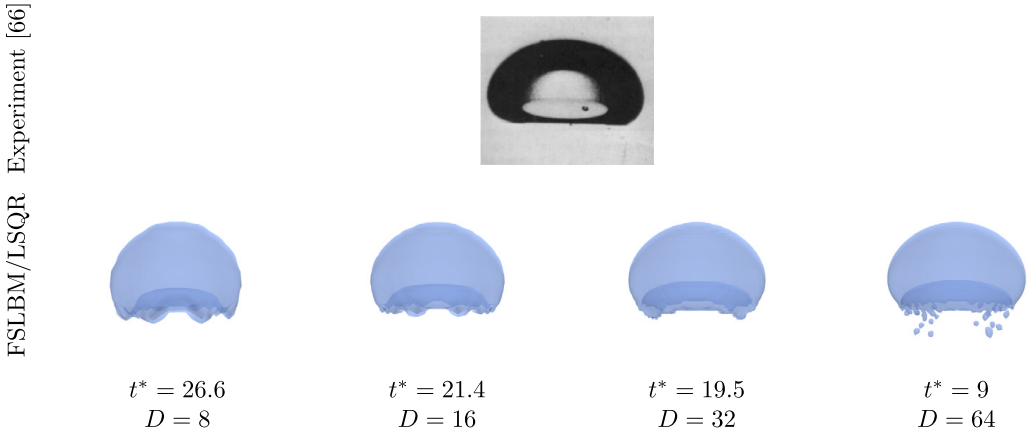


Fig. 23. Simulated bubble shape for case 3 in Table 1 with Bo = 243 and Mo = 266. Different computational resolutions according to the initial bubble diameter, D, are shown. The simulations were performed with the FSLBM with LSQR curvature computation model implemented in FluidX3D [4,40]. The photograph of the laboratory experiment was reprinted from Reference [66] with the permission of Cambridge University Press.

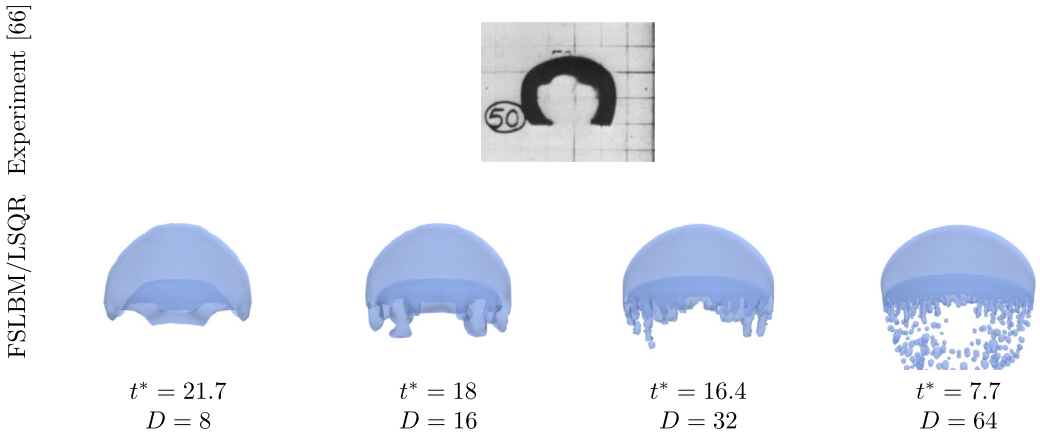


Fig. 24. Simulated bubble shape for case 4 in Table 1 with Bo = 339 and Mo = 43.1. The simulations were performed with the FSLBM with LSQR curvature computation model implemented in FluidX3D [4,40]. Different computational resolutions according to the initial bubble diameter, D, are shown. The photograph of the laboratory experiment was reprinted from Reference [66] with the permission of Cambridge University Press.

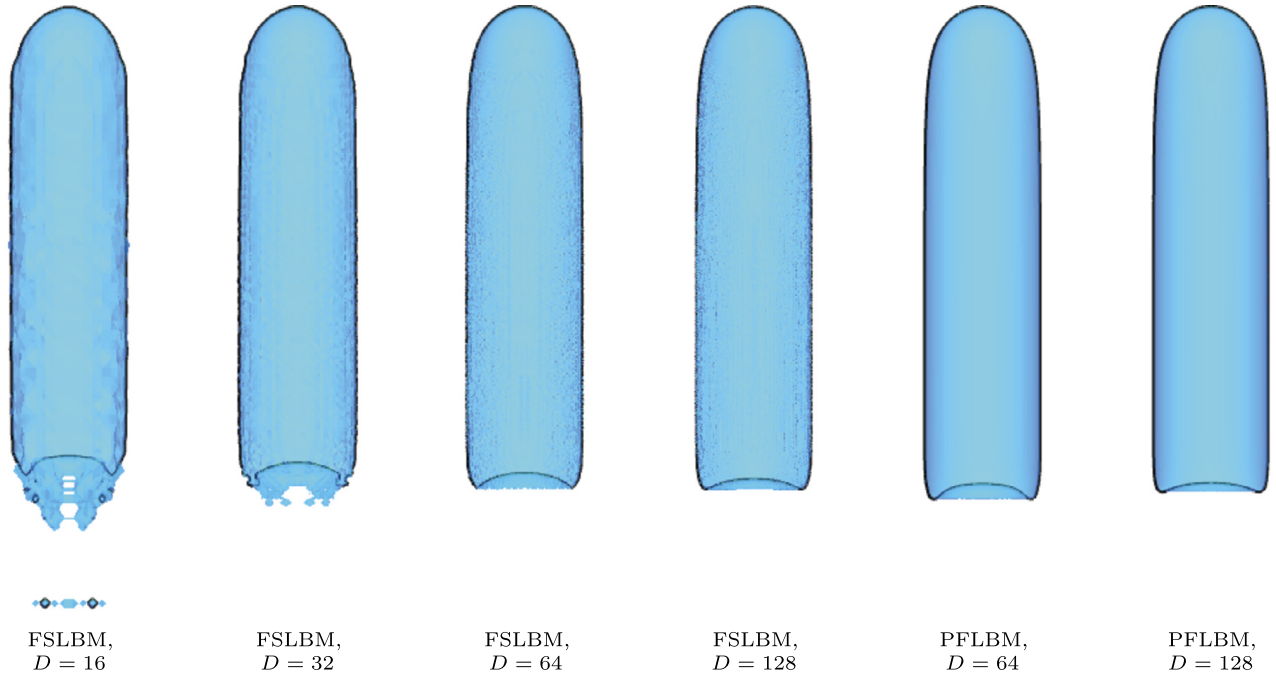


Fig. 25. Shape of the Taylor bubble at time, $t^* = 15$, as simulated with the FSLBM and PFLBM at different computational resolutions, defined by tube diameter, D. The solid black line illustrates the bubble's contour in the center cross-section with normal in the x-direction.

to capture the droplets ejected from the crown formation sufficiently well. As for the rising bubble, the PFLBM was observed to be sensitive to the choice of the mobility and interface width, but no generally applicable choice could be identified.

The investigation of the optimal choice of mobility and interface width in the PFLBM remains future work. Extending the implementations of both models with adaptive refinement of the computational grid is expected to significantly enhance the issues observed, and the efficiency of the implementations. Additionally, the applicability of the FSLBM to code generation should be explored leading to a flexible and portable code basis.

CRediT authorship contribution statement

Christoph Schwarzmeier: Conceptualization, Data curation, Funding acquisition, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Markus Holzer:** Investigation, Methodology, Software, Validation, Writing – original draft. **Travis Mitchell:** Investigation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Moritz Lehmann:** Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Fabian Häusl:** Software, Writing – review & editing. **Ulrich Rüde:** Funding acquisition, Resources, Supervision, Writing – review & editing.

195

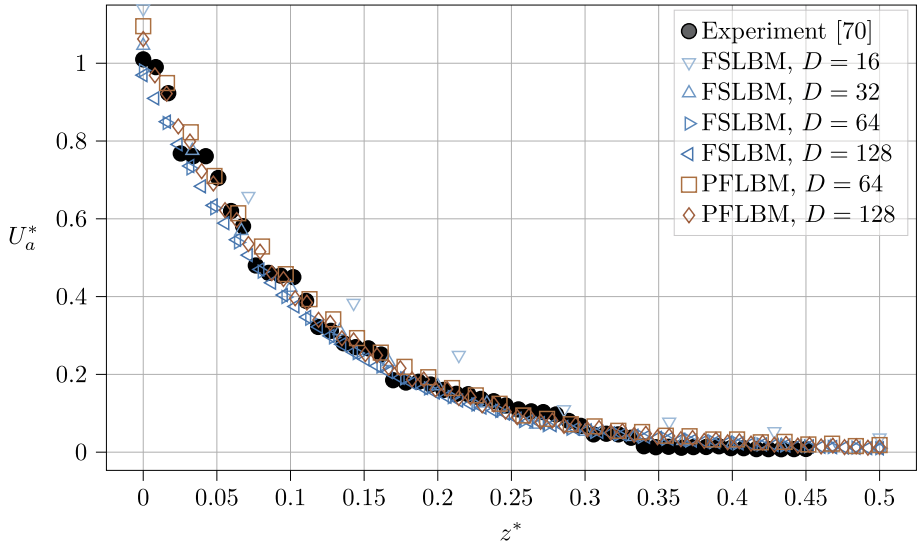


Fig. 26. Simulated non-dimensionalized axial velocity, U_a^*, along an axial line of length $0.5D$ in front of the Taylor bubble (see Fig. 12). The line is located in the center of the boundary tube with diameter, D. The comparison with experimental data [70] is drawn in terms of the non-dimensionalized axial location, $z^* = z/D$, at time, $t^* = 15$.

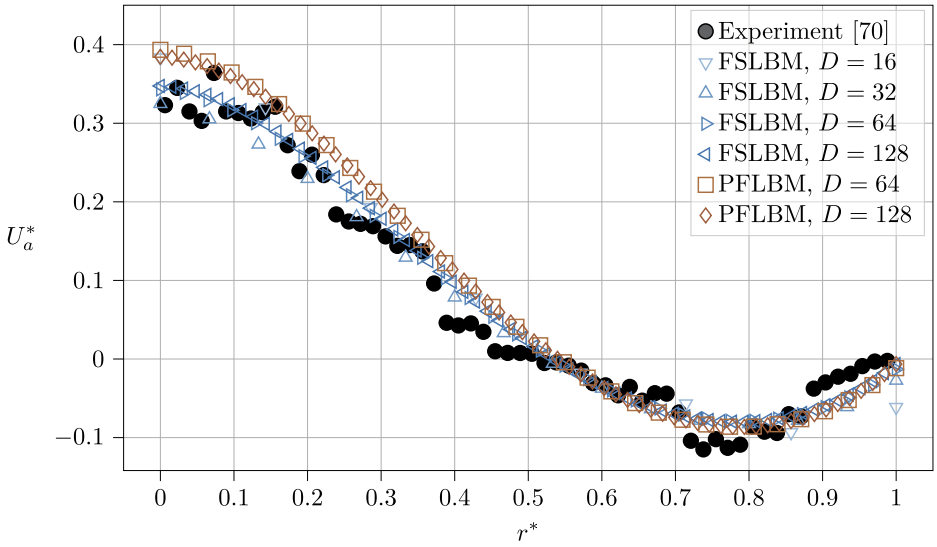


Fig. 27. Simulated non-dimensionalized axial velocity, U_a^*, along a radial line positioned at $0.111D$ in front of the Taylor bubble (see Fig. 12), with tube diameter, D. The comparison with experimental data [70] is drawn in terms of the non-dimensionalized radial location, $r^* = r/(0.5D)$, at time, $t^* = 15$.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The software used for the simulations in this study (waLBerla and lbmpy) is open source. The source code of the simulation setups and a brief description are provided as part of the supplementary material.

Acknowledgements

The authors C. Schwarzmeier and U. Rüde are grateful to the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for funding project 408062554.
This work was supported by the SCALABLE project. This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No. 956000. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Germany, the Czech Republic.

196

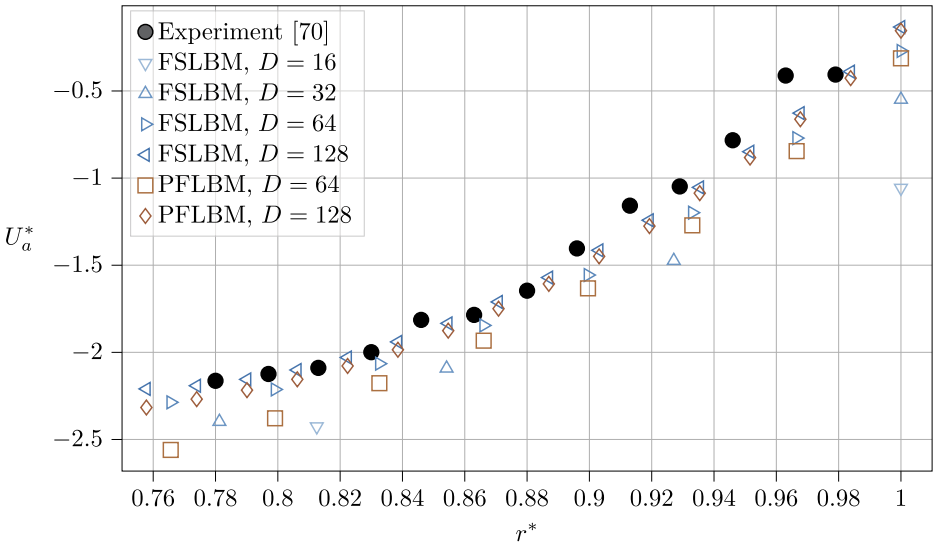


Fig. 28. Simulated non-dimensionalized axial velocity, U_a^*, along a radial line positioned at $2D$ behind the Taylor bubble's front (see Fig. 12), with tube diameter, D. The comparison with experimental data [70] is drawn in terms of the non-dimensionalized radial location, $r^* = r/(0.5D)$, at time, $t^* = 15$.

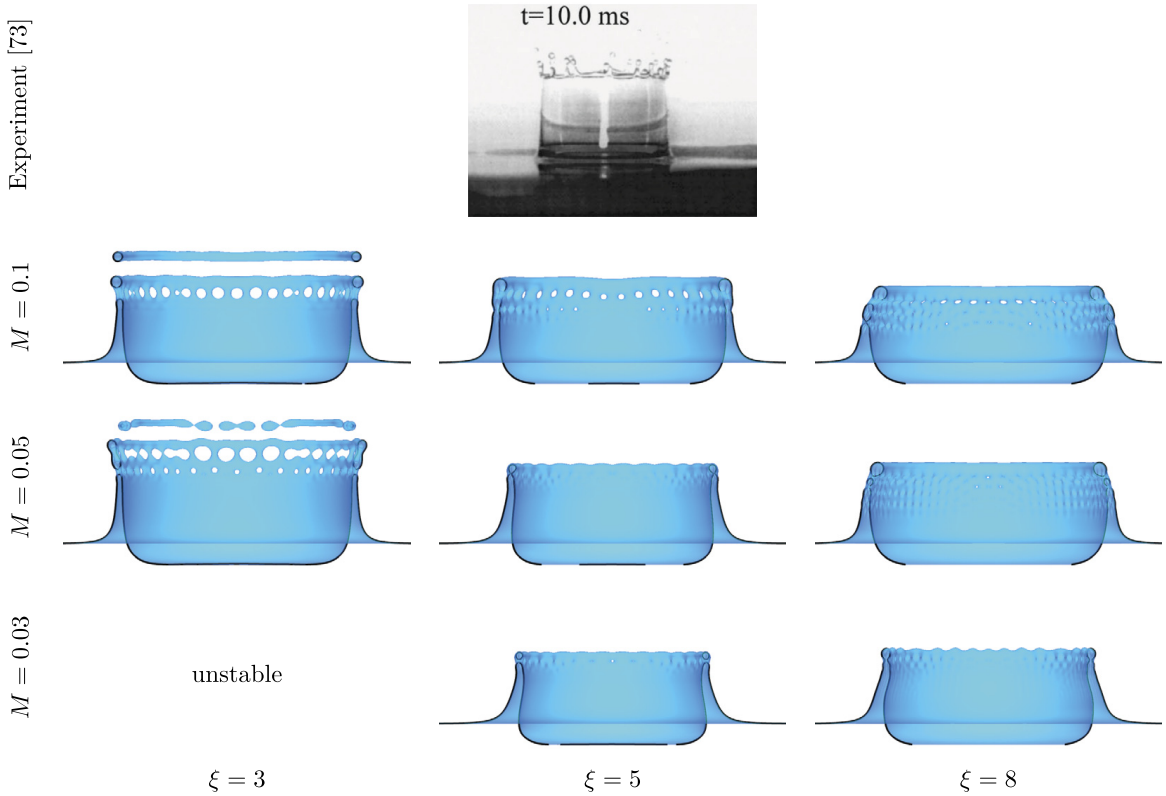


Fig. 29. Vertical drop impact at reference time $t^* = 12$, as simulated with the PFLBM with initial drop diameter, $D = 40$. The influence of the mobility, M, and interface width, ξ, are shown. While the simulation results are true to scale, no scale bar is available for the photograph of the experiment [73]. Therefore, the splash crown's dimension can only be compared between simulations rather than with the experiment. The solid black line illustrates the crown's contour in the center cross-section with normal in the x-direction. The photograph of the laboratory experiment was reprinted from Reference [73] with the permission of AIP Publishing.

Author T. Mitchell acknowledges that this work was supported by resources provided by the Pawsey Supercomputing Centre with funding from the Australian Government and the Government of Western Australia.
Author M. Lehmann acknowledges funding by the DFG – project number 391977956 – SFB 1357 and support from the ENB Biological Physics.
The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer SuperMUC at Leibniz Supercomputing Centre (www.lrz.de).
The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Per-

197

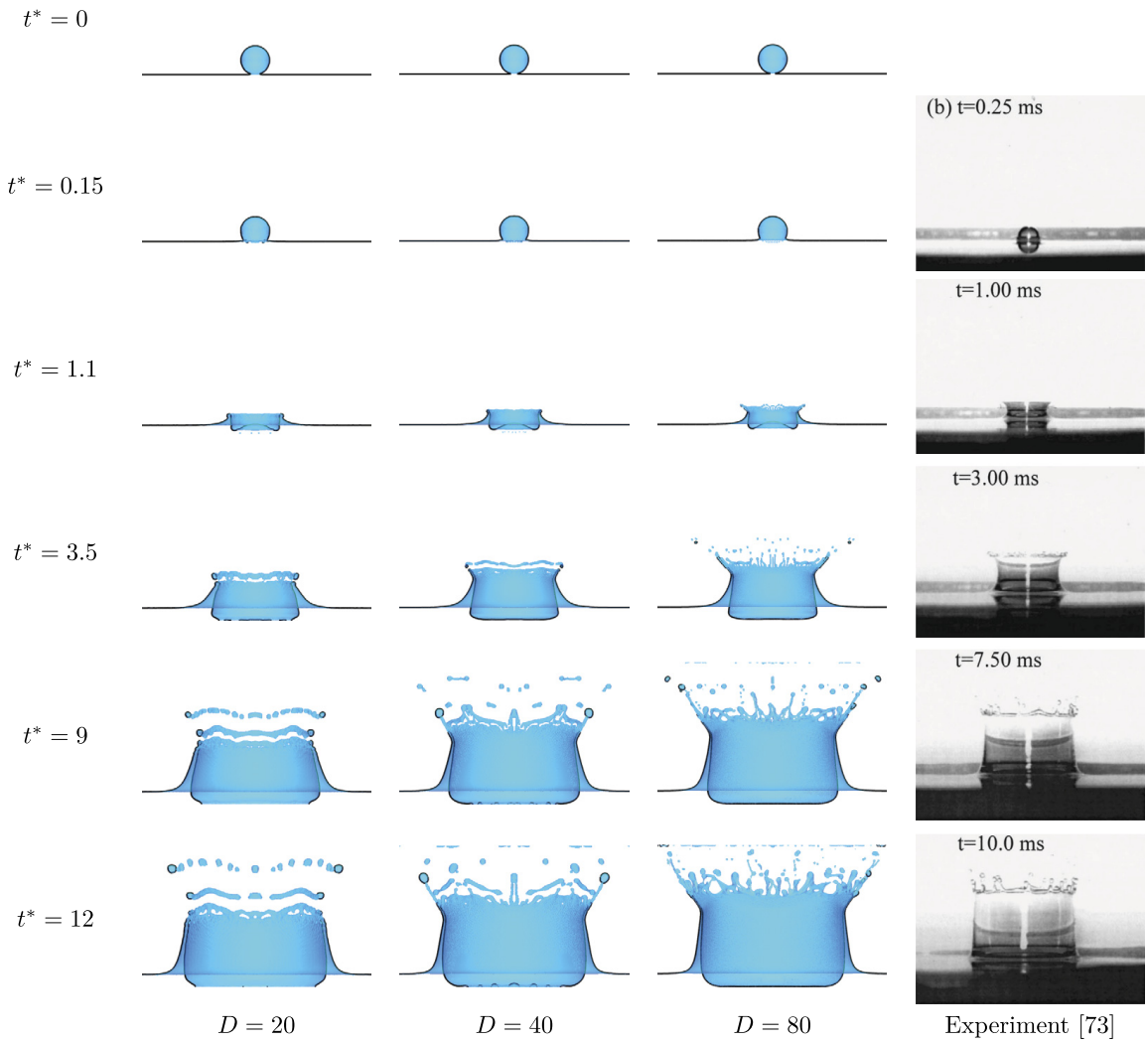


Fig. 30. Vertical drop impact over non-dimensionalized time, t^*, as simulated with the FSLBM. The computational resolution is defined by the initial drop diameter, D. While the simulation results are true to scale, no scale bar is available for the photographs of the experiment [73]. Therefore, the splash crown's dimension can only be compared between simulations rather than with the experiment. The solid black line illustrates the crown's contour in the center cross-section with normal in the x-direction. The photographs of the laboratory experiment were reprinted from Reference [73] with the permission of AIP Publishing.

formance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). The hardware is funded by the German Research Foundation (DFG).

Appendix A

The appendix presents additional simulation results that were not shown in the main part of the manuscript for reasons of brevity.

A.1. Rising bubble

Figs. 18 to 20 extend Section 3.2.1 with simulation results for the rising bubble test cases 1, 3, and 4 from Table 1.

Additionally, simulation results for the FSLBM with the LSQR curvature computation model as described in Section 3.1.2, are presented in Figs. 21 to 24. The simulations were conducted with parameters as in Table 1 using the software FluidX3D [4,40]. The results agree reasonably well with those of the FSLBM with FDM curvature computation model as presented in Figs. 8 and 18 to 20, and therefore also with the experimental results. However, as shown in Fig. 22, with the LSQR curvature model, the bubble broke apart into several smaller bubbles at resolution $D = 64$ in case 2 with $\mathrm{Bo} = 115$ and $\mathrm{Mo} = 4.63 \cdot 10^{-3}$.

198

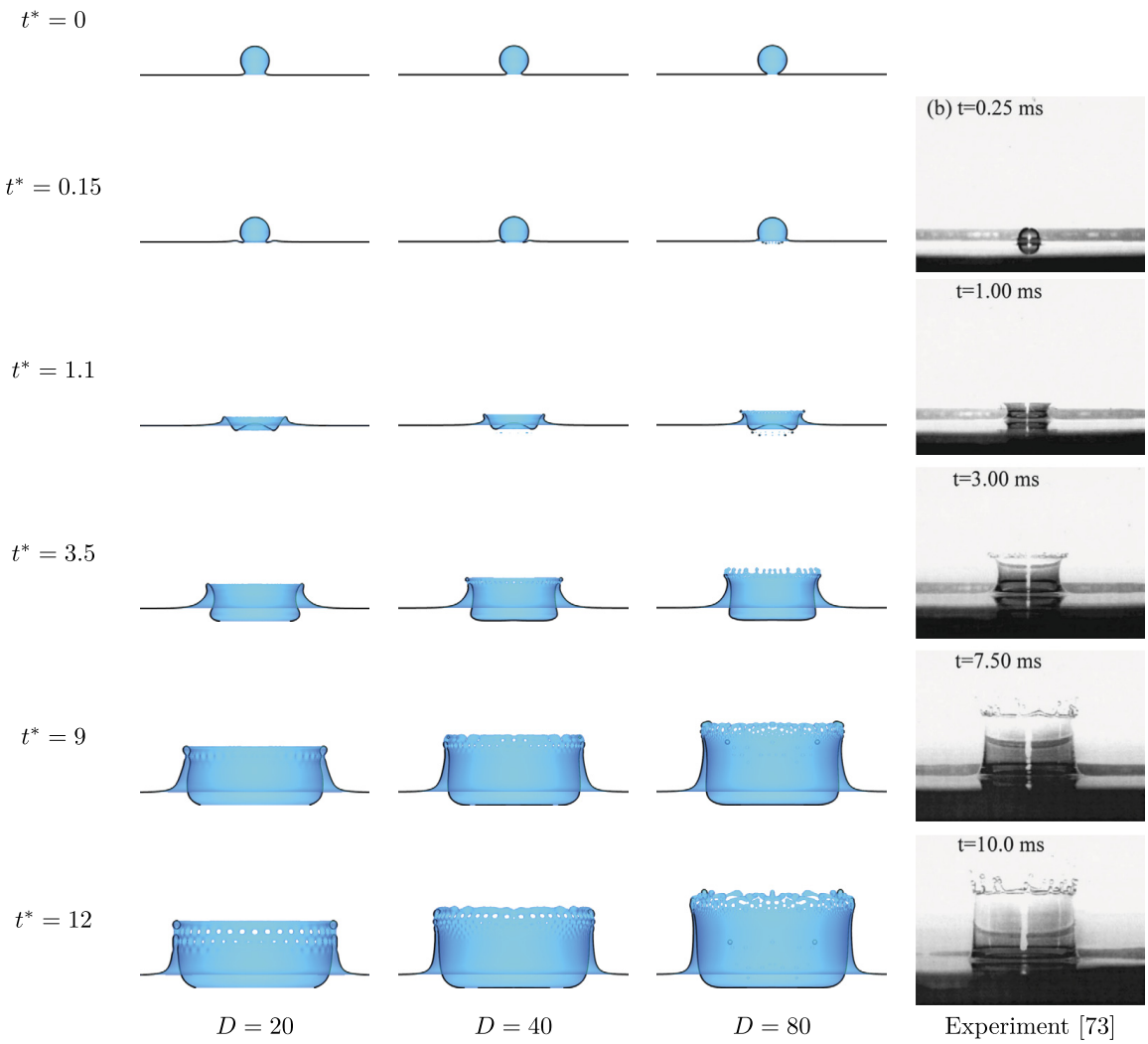


Fig. 31. Vertical drop impact over non-dimensionalized time, t^*, as simulated with the PFLBM. The computational resolution is defined by the initial drop diameter, D. While the simulation results are true to scale, no scale bar is available for the photographs of the experiment [73]. Therefore, the splash crown's dimension can only be compared between simulations rather than with the experiment. The solid black line illustrates the crown's contour in the center cross-section with normal in the x-direction. The photographs of the laboratory experiment were reprinted from Reference [73] with the permission of AIP Publishing.

Table 3
Simulated non-dimensionalized cavity depth, $h^*_{ca}(t^*) = h_{ca}(t^*)/D$, of the vertical drop impact. The cavity depth, $h_{ca}(t^*)$, is the maximum distance of the cavity bottom to the initial position of the liquid surface at time, $t^* = 0$, measured in the center cross-section with normal in the x-direction. The results are presented for different dimensionless times, t^*, and computational resolutions as defined by the initial drop diameter, D.

	t^*	1.1	3.5	9	12
FSLBM, $D = 20$	h^*_{cav}	−0.18	−0.42	−0.42	−0.42
FSLBM, $D = 40$		−0.17	−0.43	−0.46	−0.46
FSLBM, $D = 80$		−0.15	−0.43	−0.45	−0.45
PFLBM, $D = 20$		−0.18	−0.45	−0.45	−0.45
PFLBM, $D = 40$		−0.14	−0.44	−0.48	−0.48
PFLBM, $D = 80$		−0.38	−0.43	−0.47	−0.47

A.2. Taylor bubble

Figs. 25 to 28 extend Section 3.2.2 with the shape of the Taylor bubble and additional evaluations of the flow field in the surrounding fluid according to Fig. 12.

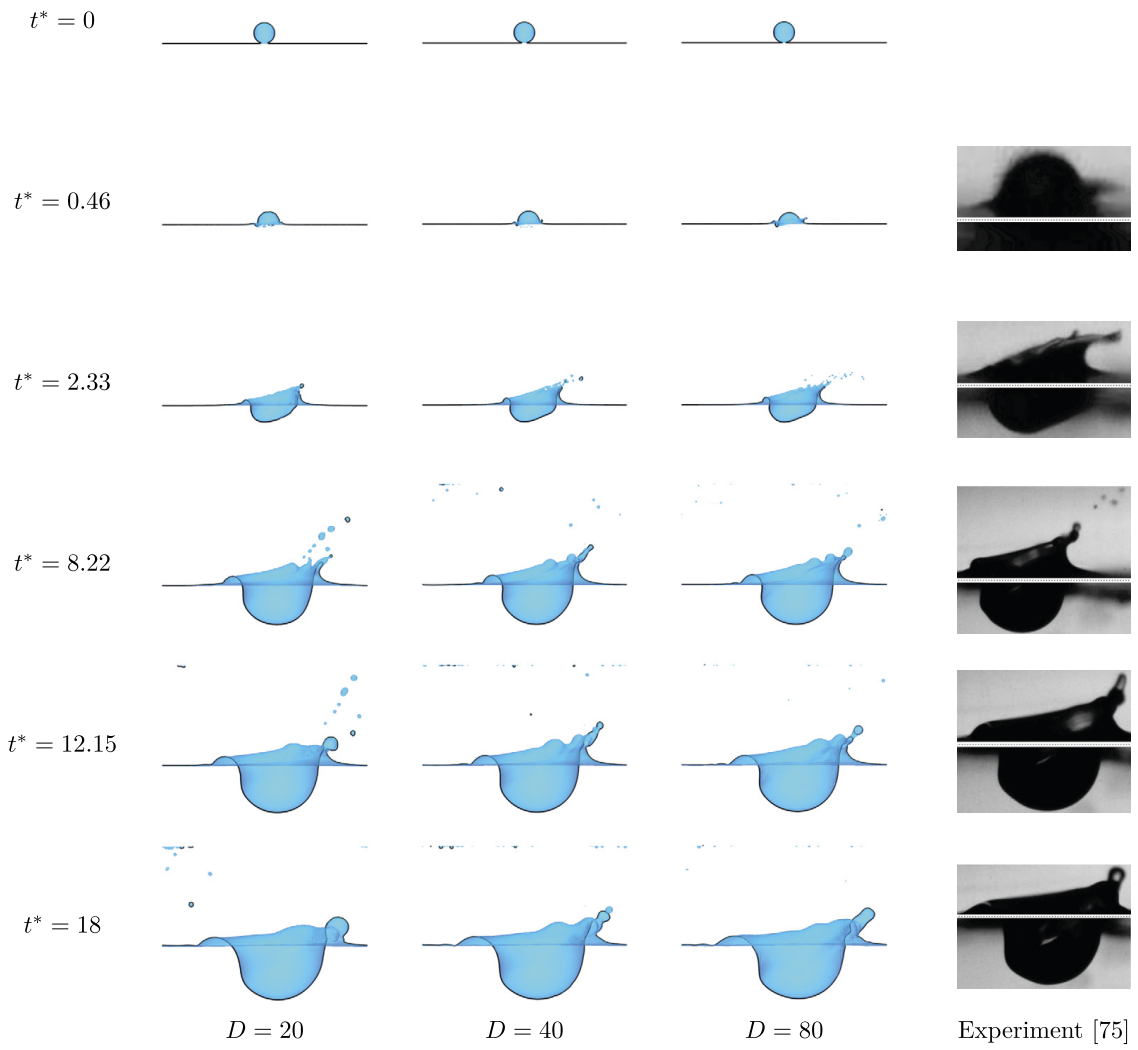


Fig. 32. Oblique drop impact over non-dimensionalized time, t^*, as simulated with the FSLBM. The computational resolution is defined by the initial drop diameter, D. While the simulation results are true to scale, no scale bar is available for the photographs of the experiment [75]. Therefore, the splash crown's dimension can only be compared between simulations rather than with the experiment. The solid black line illustrates the crown's contour in the center cross-section with normal in the x-direction. The photographs of the laboratory experiment were reprinted from Reference [75] with the permission of the original authors.

Table 4
Simulated non-dimensionalized splash crown diameter, $d^*_{cr}(t^*) = d_{cr}(t^*)/D$, of the vertical drop impact. The splash crown diameter, $d_{cr}(t^*)$, is the crown's inner diameter at the position of the initial liquid surface at time, $t^* = 0$, measured in a center cross-section with normal in the x-direction. The results are presented for different dimensionless times, t^*, and computational resolutions as defined by the initial drop diameter, D.

	t^*	1.1	3.5	9	12
FSLBM, $D = 20$	d^*_{cr}	1.66	2.99	4.50	4.93
FSLBM, $D = 40$		1.76	3.03	4.57	4.93
FSLBM, $D = 80$		1.73	3.05	4.55	4.92
PFLBM, $D = 20$		1.66	2.99	4.71	5.17
PFLBM, $D = 40$		1.62	2.94	4.58	5.02
PFLBM, $D = 80$		1.72	2.98	4.60	5.04

A.3. Vertical drop impact

Extending Section 3.3.1, Fig. 29 illustrates the PFLBM's sensitivity to the mobility parameter, M, and interface width, ξ, in the vertical drop impact test case. Figs. 30 and 31 qualitatively compare the simulated vertical drop impact with experimental data at different points in time. Tables 3 and 4 present the temporal evolution of the quantified simulated cavity depth and inner crown diameter.

200

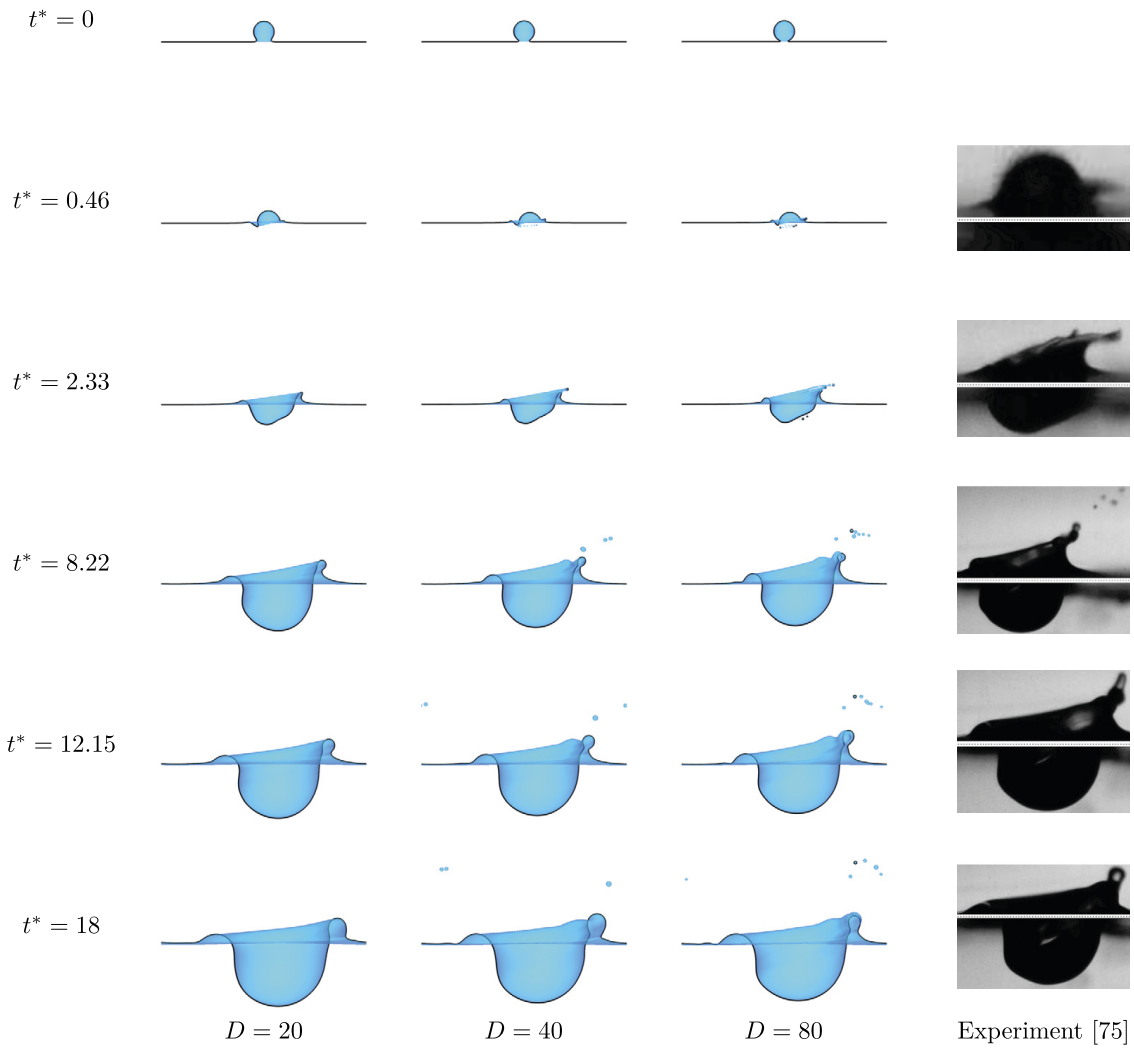


Fig. 33. Oblique drop impact over non-dimensionalized time, t^*, as simulated with the PFLBM. The computational resolution is defined by the initial drop diameter, D. While the simulation results are true to scale, no scale bar is available for the photographs of the experiment [75]. Therefore, the splash crown's dimension can only be compared between simulations rather than with the experiment. The solid black line illustrates the crown's contour in the center cross-section with normal in the x-direction. The photographs of the laboratory experiment were reprinted from Reference [75] with the permission of the original authors.

Table 5
Simulated non-dimensionalized cavity depth, $h^*_{ca}(t^*) = h_{ca}(t^*)/D$, of the oblique drop impact. The cavity depth, $h_{ca}(t^*)$, is the maximum distance of the cavity bottom to the initial position of the liquid surface at time, $t^* = 0$, measured in the center cross-section with normal in the x-direction. The results are presented for different dimensionless times, t^*, and computational resolutions as defined by the initial drop diameter, D.

	t^*	2.33	8.22	12.15	18
FSLBM, $D = 20$	h^*_{ca}	−0.80	−1.85	−2.19	−2.5
FSLBM, $D = 40$		−0.81	−1.86	−2.2	−2.49
FSLBM, $D = 80$		−0.84	−1.87	−2.13	−2.48
PFLBM, $D = 20$		−0.96	−2.18	−2.5	−2.84
PFLBM, $D = 40$		−0.92	−2.03	−2.37	−2.68
PFLBM, $D = 80$		−0.88	−1.95	−2.27	−2.58

A.4. Oblique drop impact

Figs. 32 and 33 extend Section 3.3.2 and qualitatively compare the simulated oblique drop impact with experimental data at different points in time. Tables 5 and 6 present the temporal evolution of the quantified simulated cavity depth and inner crown diameter.

201

Table 6
Simulated non-dimensionalized splash crown diameter, $d^*_{cr}(t^*) = d_{cr}(t^*)/D$, of the oblique drop impact. The splash crown diameter, $d_{cr}(t^*)$, is the crown's inner diameter at the position of the initial liquid surface at time, $t^* = 0$, measured in a center cross-section with normal in the x-direction. The results are presented for different dimensionless times, t^*, and computational resolutions as defined by the initial drop diameter, D.

	t^*	2.33	8.22	12.15	18
FSLBM, $D = 20$	d^*_{cr}	2.23	3.39	3.87	4.43
FSLBM, $D = 40$		2.23	3.44	3.9	4.40
FSLBM, $D = 80$		2.25	3.44	3.88	4.46
PFLBM, $D = 20$		2.20	3.4	3.88	4.6
PFLBM, $D = 40$		2.14	3.38	3.92	4.45
PFLBM, $D = 80$		2.19	3.45	3.95	4.49

Appendix B. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jcp.2022.111753.
The following supplementary material is available as part of the online article:

- An archive of the C++ source code of the FSLBM and PFLBM as part of the software framework waLBerla [39], version https://i10git.cs.fau.de/walberla/walberla/-/tree/01a28162ae1aacf7b96152c9f886ce54cc7f53ff.
  The simulation setups are located in the directories `apps/showcasesFreeSurface` and `apps/showcases/PhaseFieldAllenCahn/CPU`.
- An archive of the Python source code used for the PFLBM gravity and capillary wave test cases. These test cases are provided as Jupyter Notebooks as part of the code generation framework *lbmpy* [41], version https://pypi.org/project/lbmpy/1.0.1/. The notebooks are located in the directory `lbmpy_tests/full_scenarios/phasefield_allen_cahn`.

References

[1] H. Leuner, C. Gerstenberg, K. Lechner, C. McHardy, C. Rauh, J.-U. Repke, Overcoming unwanted foam in industrial processes of the chemical and food industry – an ongoing survey, Chem. Eng. Res. Des. 163 (2020), https://doi.org/10.1016/j.cherd.2020.09.006.
[2] J. Thünnesen, B. Gatternig, A. Delgado, Ultrasonic effects on foam formation of fruit juices during bottling, Engineering 2 (3) (2021), https://doi.org/10.3390/eng2030023.
[3] B. Wu, M. Firouzi, T. Mitchell, T.E. Rufford, C. Leonardi, B. Towler, A critical review of flow maps for gas-liquid flows in vertical pipes and annuli, Chem. Eng. J. 326 (2017) 350–377, https://doi.org/10.1016/j.cej.2017.05.135.
[4] M. Lehmann, L.M. Oehlschlägel, F.P. Häusl, A. Held, S. Gekle, Ejection of marine microplastics by raindrops: a computational and experimental study, Microplast. Nanoplast. 1 (1) (2021), https://doi.org/10.1186/s43591-021-00018-8.
[5] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annu. Rev. Fluid Mech. 31 (1) (1999), https://doi.org/10.1146/annurev.fluid.31.1.567.
[6] S. Bogner, J. Harting, U. Rüde, Direct simulation of liquid–gas–solid flow with a free surface lattice Boltzmann method, Int. J. Comput. Fluid Dyn. 31 (10) (2017), https://doi.org/10.1080/10618562.2018.1424836.
[7] S. Donath, K. Mecke, S. Rabha, V. Buwa, U. Rüde, Verification of surface tension in the parallel free surface lattice Boltzmann method in waLBerla, Comput. Fluids 45 (1) (2011), https://doi.org/10.1016/j.compfluid.2010.12.027.
[8] C. Körner, M. Thies, T. Hofmann, N. Thürey, U. Rüde, Lattice Boltzmann model for free surface flow for modeling foaming, J. Stat. Phys. 121 (1) (2005), https://doi.org/10.1007/s10955-005-8879-8.
[9] M. Wöhrwag, C. Semprebon, A. Mazloomi Moqaddam, I. Karlin, H. Kusumaatmaja, Ternary free-energy entropic lattice Boltzmann model with a high density ratio, Phys. Rev. Lett. 120 (23) (2018), https://doi.org/10.1103/PhysRevLett.120.234501.
[10] X. Shan, H. Chen, Simulation of nonideal gases and liquid-gas phase transitions by the lattice Boltzmann equation, Phys. Rev. E 49 (4) (1994), https://doi.org/10.1103/PhysRevE.49.2941.
[11] P.-H. Chiu, Y.-T. Lin, A conservative phase field method for solving incompressible two-phase flows, J. Comput. Phys. 230 (1) (2011), https://doi.org/10.1016/j.jcp.2010.09.021.
[12] C. Rettinger, U. Rüde, An efficient four-way coupled lattice Boltzmann – discrete element method for fully resolved simulations of particle-laden flows, J. Comput. Phys. 453 (2022), https://doi.org/10.1016/j.jcp.2022.110942.
[13] M. Kuron, G. Rempfer, F. Schornbaum, M. Bauer, C. Godenschwager, C. Holm, J. de Graaf, Moving charged particles in lattice Boltzmann-based electrokinetics, J. Chem. Phys. 145 (21) (2016), https://doi.org/10.1063/1.4968596.
[14] T.R. Mitchell, M. Majidi, M.H. Rahimian, C.R. Leonardi, Computational modeling of three-dimensional thermocapillary flow of recalcitrant bubbles using a coupled lattice Boltzmann-finite difference method, Phys. Fluids 33 (3) (2021), https://doi.org/10.1063/5.0038171.
[15] J. Becker, M. Junk, D. Kehrwald, G. Thömmes, Z. Yang, A combined lattice BGK/level set method for immiscible two-phase flows, Comput. Math. Appl. 58 (5) (2009), https://doi.org/10.1016/j.camwa.2009.02.005.
[16] P. Lallemand, L.-S. Luo, Y. Peng, A lattice Boltzmann front-tracking method for interface dynamics with surface tension in two dimensions, J. Comput. Phys. 226 (2) (2007), https://doi.org/10.1016/j.jcp.2007.05.021.
[17] A.K. Gunstensen, D.H. Rothman, S. Zaleski, G. Zanetti, Lattice Boltzmann model of immiscible fluids, Phys. Rev. A 43 (8) (1991), https://doi.org/10.1103/PhysRevA.43.4320.
[18] M.R. Swift, W.R. Osborn, J.M. Yeomans, Lattice Boltzmann simulation of nonideal fluids, Phys. Rev. Lett. 75 (5) (1995), https://doi.org/10.1103/PhysRevLett.75.830.
[19] T. Inamuro, T. Ogata, S. Tajima, N. Konishi, A lattice Boltzmann method for incompressible two-phase flows with large density differences, J. Comput. Phys. 198 (2) (2004), https://doi.org/10.1016/j.jcp.2004.01.019.
[20] H.W. Zheng, C. Shu, Y.T. Chew, Lattice Boltzmann interface capturing method for incompressible flows, Phys. Rev. E 72 (5) (2005), https://doi.org/10.1103/PhysRevE.72.056705.

[21] A. Fakhari, T. Mitchell, C. Leonardi, D. Bolster, Improved locality of the phase-field lattice-Boltzmann model for immiscible fluids at high density ratios, Phys. Rev. E 96 (5) (2017), https://doi.org/10.1103/physreve.96.053301.
[22] T. Mitchell, C. Leonardi, A. Fakhari, Development of a three-dimensional phase-field lattice Boltzmann method for the study of immiscible fluids at high density ratios, Int. J. Multiph. Flow 107 (2018) 1–15, https://doi.org/10.1016/j.ijmultiphaseflow.2018.05.004.
[23] C. Hirt, B. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1) (1981), https://doi.org/10.1016/0021-9991(81)90145-5.
[24] C. Janßen, M. Krafczyk, Free surface flow simulations on GPGPUs using the LBM, Comput. Math. Appl. 61 (12) (2011), https://doi.org/10.1016/j.camwa.2011.03.016.
[25] Y. Sun, C. Beckermann, Sharp interface tracking using the phase-field equation, J. Comput. Phys. 220 (2) (2007) 626–653, https://doi.org/10.1016/j.jcp.2006.05.025.
[26] X. He, X. Shan, G.D. Doolen, Discrete Boltzmann equation model for nonideal gases, Phys. Rev. E 57 (1) (1998), https://doi.org/10.1103/PhysRevE.57.R13.
[27] K.N. Premnath, J. Abraham, Lattice Boltzmann model for axisymmetric multiphase flows, Phys. Rev. E 71 (5) (2005), https://doi.org/10.1103/PhysRevE.71.056706.
[28] A. Fakhari, M.H. Rahimian, Simulation of falling droplet by the lattice Boltzmann method, Commun. Nonlinear Sci. Numer. Simul. 14 (7) (2009), https://doi.org/10.1016/j.cnsns.2008.10.017.
[29] G. Gruszczyński, T. Mitchell, C. Leonardi, Ł. Łaniewski-Wołłk, T. Barber, A cascaded phase-field lattice Boltzmann model for the simulation of incompressible, immiscible fluids with high density contrast, Comput. Math. Appl. 79 (4) (2020), https://doi.org/10.1016/j.camwa.2019.08.018.
[30] T. Mitchell, C. Leonardi, Development of closure relations for the motion of Taylor bubbles in vertical and inclined annular pipes using high-fidelity numerical modeling, Phys. Fluids 32 (6) (2020), https://doi.org/10.1063/5.0011456.
[31] T. Mitchell, C. Leonardi, On the rise characteristics of Taylor bubbles in annular piping, Int. J. Multiph. Flow 130 (2020), https://doi.org/10.1016/j.ijmultiphaseflow.2020.103376.
[32] T. Mitchell, M. Holzer, C. Schwarzmeier, M. Bauer, U. Rüde, C. Leonardi, Stability assessment of the phase-field lattice Boltzmann model and its application to Taylor bubbles in annular piping geometries, Phys. Fluids 33 (2021) 083325, https://doi.org/10.1063/5.0061694.
[33] M. Holzer, M. Bauer, H. Köstler, U. Rüde, Highly efficient lattice Boltzmann multiphase simulations of immiscible fluids at high-density ratios on CPUs and GPUs through code generation, Int. J. High Perform. Comput. Appl. 35 (4) (2021), https://doi.org/10.1177/10943420211016525.
[34] E. Dinesh Kumar, S.A. Sannasiraj, V. Sundar, Phase field lattice Boltzmann model for air-water two phase flows, Phys. Fluids 31 (7) (2019), https://doi.org/10.1063/1.5100215.
[35] T. Mitchell, B. Hill, M. Firouzi, C. Leonardi, Development and evaluation of multiphase closure models used in the simulation of unconventional wellbore dynamics, in: SPE/AAPG/SEG Asia Pacific Unconventional Resources Technology Conference, OnePetro, 2019.
[36] A. Fakhari, M. Geier, T. Lee, A mass-conserving lattice Boltzmann method with dynamic grid refinement for immiscible two-phase flows, J. Comput. Phys. 315 (2016) 434–457, https://doi.org/10.1016/j.jcp.2016.03.058.
[37] A. Fakhari, D. Bolster, L.-S. Luo, A weighted multiple-relaxation-time lattice Boltzmann method for multiphase flows and its application to partial coalescence cascades, J. Comput. Phys. 341 (2017), https://doi.org/10.1016/j.jcp.2017.03.062.
[38] N. Thürey, Physically based Animation of Free Surface Flows with the Lattice Boltzmann Method, Ph.D. thesis, Universität Erlangen-Nürnberg, 2007, https://www10.cs.fau.de/publications/dissertations/Diss_2007-Thuerey.pdf.
[39] M. Bauer, S. Eibl, C. Godenschwager, N. Kohl, M. Kuron, C. Rettinger, F. Schornbaum, C. Schwarzmeier, D. Thönnes, H. Köstler, U. Rüde, walberla: a block-structured high-performance framework for multiphysics simulations, Comput. Math. Appl. 81 (2021) 478–501, https://doi.org/10.1016/j.camwa.2020.01.007.
[40] M. Lehmann, S. Gekle, Analytic solution to the piecewise linear interface construction problem and its application in curvature calculation for volume-of-fluid simulation codes, Computation 10 (2) (2022), https://doi.org/10.3390/computation10020021.
[41] M. Bauer, H. Köstler, U. Rüde, lbmpy: automatic code generation for efficient parallel lattice Boltzmann methods, J. Comput. Sci. 49 (2021) 101269, https://doi.org/10.1016/j.jocs.2020.101269.
[42] Ł. Łaniewski-Wołłk, J. Rokicki, Adjoint lattice Boltzmann for topology optimization on multi-GPU architecture, Comput. Math. Appl. 71 (3) (2016), https://doi.org/10.1016/j.camwa.2015.12.043.
[43] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E.M. Viggen, The Lattice Boltzmann Method, Springer International Publishing, 2017.
[44] D. d'Humières, Generalized lattice-Boltzmann equations, in: Rarefied Gas Dynamics: Theory and Simulations, Progress in Astronautics and Aeronautics, American Institute of Aeronautics and Astronautics, 1994.
[45] M. Bauer, G. Silva, U. Rüde, Truncation errors of the D3Q19 lattice model for the lattice Boltzmann method, J. Comput. Phys. 405 (2020), https://doi.org/10.1016/j.jcp.2019.109111.
[46] X. He, L. Luo, Lattice Boltzmann model for the incompressible Navier–Stokes equation, J. Stat. Phys. 88 (1997) 927–944, https://doi.org/10.1023/B:JOSS.0000015179.12689.e4.
[47] M. Geier, M. Schönherr, A. Pasquali, M. Krafczyk, The cumulant lattice Boltzmann equation in three dimensions: theory and validation, Comput. Math. Appl. 70 (4) (2015) 507–547, https://doi.org/10.1016/j.camwa.2015.05.001.
[48] M. Lehmann, M.J. Krause, G. Amati, M. Sega, J. Harting, S. Gekle, On the accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit and novel 16-bit number formats, arXiv preprint, arXiv:2112.08926, https://arxiv.org/abs/2112.08926, 2021.
[49] T. Pohl, High Performance Simulation of Free Surface Flows Using the Lattice Boltzmann Method, Ph.D. thesis, Universität Erlangen-Nürnberg, 2008, https://www10.cs.fau.de/publications/dissertations/Diss_2008-Pohl.pdf.
[50] S. Bogner, U. Rüde, J. Harting, Curvature estimation from a volume-of-fluid indicator function for the simulation of surface tension and wetting with a free-surface lattice Boltzmann method, Phys. Rev. E 93 (4) (2016), https://doi.org/10.1103/PhysRevE.93.043302.
[51] M. Lehmann, High Performance Free Surface LBM on GPUs, Masterthesis, Universität Bayreuth, 2019, https://doi.org/10.15495/EPub_UBT_00005400.
[52] S. Donath, Wetting Models for a Parallel High-Performance Free Surface Lattice Boltzmann Method, Ph.D. thesis, Universität Erlangen-Nürnberg, 2011, https://www10.cs.fau.de/publications/dissertations/Diss_2011-Donath.pdf.
[53] T. Pavlidis, Algorithms for Graphics and Image Processing, Springer Berlin Heidelberg, Berlin, Heidelberg, 1982.
[54] F. Häusl, Soft Objects in Newtonian and Non-Newtonian Fluids – a Computational Study of Bubbles and Capsules in Flow, Masterthesis, Universität, Bayreuth, 2021.
[55] Z. Guo, C. Zheng, B. Shi, Discrete lattice effects on the forcing term in the lattice Boltzmann method, Phys. Rev. E 65 (4) (2002), https://doi.org/10.1103/PhysRevE.65.046308.
[56] S. Donath, C. Feichtinger, T. Pohl, J. Götz, U. Rüde, Localized parallel algorithm for bubble coalescence in free surface lattice-Boltzmann method, in: H. Sips, D. Epema, H.-X. Lin (Eds.), Euro-Par 2009 Parallel Processing, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009.
[57] H. Ding, P.D.M. Spelt, Wetting condition in diffuse interface simulations of contact line motion, Phys. Rev. E 75 (4) (2007), https://doi.org/10.1103/PhysRevE.75.046708.
[58] A. Fakhari, D. Bolster, Diffuse interface modeling of three-phase contact line dynamics on curved boundaries: a lattice Boltzmann model for large density and viscosity ratios, J. Comput. Phys. 334 (2017) 620–638, https://doi.org/10.1016/j.jcp.2017.01.025.
[59] M.W. Dingemans, Water Wave Propagation over Uneven Bottoms: Part 1, Advanced Series on Ocean Engineering, vol. 13, World Scientific Publishing Company, 1997.

[60] H. Lamb, Hydrodynamics, sixth edition, Cambridge University Press, 1975.
[61] A. Prosperetti, Motion of two superposed viscous fluids, Phys. Fluids 24 (7) (1981), https://doi.org/10.1063/1.863522.
[62] F. Denner, G. Paré, S. Zaleski, Dispersion and viscous attenuation of capillary waves with finite amplitude, Eur. Phys. J. Spec. Top. 226 (6) (2017), https://doi.org/10.1140/epjst/e2016-60199-2.
[63] G. Taubin, Estimating the tensor of curvature of a surface from a polyhedral approximation, in: Proceedings of IEEE International Conference on Computer Vision, IEEE Comput. Soc. Press, Cambridge, MA, USA, 1995.
[64] K. Mecke, K. Falk, M. Rauscher, Dynamics of nanoscopic capillary waves, in: G. Radons, B. Rumpf, H.G. Schuster (Eds.), Nonlinear Dynamics of Nanosystems, Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany, 2010.
[65] J.W. Bullard, E.J. Garboczi, W.C. Carter, E.R. Fuller, Numerical methods for computing interfacial mean curvature, Comput. Mater. Sci. 4 (2) (1995), https://doi.org/10.1016/0927-0256(95)00014-H.
[66] D. Bhaga, M.E. Weber, Bubbles in viscous liquids: shapes, wakes and velocities, J. Fluid Mech. 105 (1981), https://doi.org/10.1017/S002211208100311X.
[67] F. Ren, B. Song, M.C. Sukop, Terminal shape and velocity of a rising bubble by phase-field-based incompressible lattice Boltzmann model, Adv. Water Resour. 97 (2016), https://doi.org/10.1016/j.advwatres.2016.08.012.
[68] R.M. Davies, G. Taylor, The mechanics of large bubbles rising through extended liquids and through liquids in tubes, Proc. R. Soc. Lond. Ser. A, Math. Phys. Sci. 200 (1062) (1950) 375–390, https://doi.org/10.1098/rspa.1950.0023.
[69] F.P. Bretherton, The motion of long bubbles in tubes, J. Fluid Mech. 10 (02) (1961) 166, https://doi.org/10.1017/s0022112061000160.
[70] J. Bugg, G. Saad, The velocity field around a Taylor bubble rising in a stagnant viscous fluid: numerical and experimental results, Int. J. Multiph. Flow 28 (5) (2002), https://doi.org/10.1016/S0301-9322(02)00002-2.
[71] Y.S. Joung, C.R. Buie, Aerosol generation by raindrop impact on soil, Nat. Commun. 6 (1) (2015) 1–9.
[72] J. Lu, F. Zheng, G. Li, F. Bian, J. An, The effects of raindrop impact and runoff detachment on hillslope soil erosion and soil aggregate loss in the mollisol region of northeast China, Soil Tillage Res. 161 (2016) 79–85.
[73] A.-B. Wang, C.-C. Chen, Splashing impact of a single drop onto very thin liquid films, Phys. Fluids 12 (9) (2000), https://doi.org/10.1063/1.1287511.
[74] M.V. Gielen, P. Sleutel, J. Benschop, M. Riepen, V. Voronina, C.W. Visser, D. Lohse, J.H. Snoeijer, M. Versluis, H. Gelderblom, Oblique drop impact onto a deep liquid pool, Phys. Rev. Fluids 2 (8) (2017), https://doi.org/10.1103/PhysRevFluids.2.083602.
[75] S.A. Reijers, B. Liu, D. Lohse, H. Gelderblom, Oblique droplet impact onto a deep liquid pool, arXiv:1903.08978 [physics], 2019, arXiv:1903.08978, https://arxiv.org/abs/1903.08978.

8.7 Publication 7

Water-air transfer rates of microplastic particles through bubble bursting as a function of particle size

by

Lisa Marie Oehlschlägel, Sebastian Schmid, Moritz Lehmann, Stephan Gekle, and Andreas Held

Manuscript submitted for peer-review in *Microplastics and Nanoplastics*. (2022)

Water-air transfer rates of microplastic particles through bubble bursting as a function of particle size

Lisa Marie Oehlschlägel[1,*], Sebastian Schmid[1], Moritz Lehmann[2], Stephan Gekle[2], Andreas Held[1]

[1]Environmental Chemistry and Air Research, Technische Universität Berlin, Berlin, Germany

[2]Biofluid Simulation and Modeling, University of Bayreuth, Bayreuth, Germany

*Corresponding author: Lisa Marie Oehlschlägel, Tel. +49 030 314 79608, email address: oehlschlaegel@tu-berlin.de, postal address: Technische Universität Berlin, Fachgebiet Umweltchemie und Luftreinhaltung, KF3, Straße des 17. Juni 135, 10623 Berlin, GERMANY

Abstract

Microplastic (MP) particles can be ejected into the air by jet drops when gas bubbles burst at water surfaces. For a qualitative and quantitative understanding of this transport mechanism from the hydrosphere to the atmosphere, we studied the transfer of MP due to bubble bursting at the air-water interface in laboratory experiments. Gas bubbles were produced with filtered air that was pushed through a stainless-steel frit at two different volume flow rates in a glass flask filled with polystyrene (PS) particles of six different diameters (0.35 µm, 0.5 µm, 0.75 µm, 1 µm, 1.5 µm, 2 µm) suspended in deionized water. Airborne PS particle concentrations were measured by an optical particle counter. Additionally, size and volume of the bursting bubbles and the resulting jet droplets were analyzed with a camera. Depending on the volume flow rates, bubble bursting rates from 688 s^{-1} to 1176 s^{-1} and mean diameters of the bursting bubbles from 0.76 mm to 0.81 mm were observed. The mean diameters of the top jet drops were estimated to be between 0.10 mm and 0.11 mm. The measured number of jet droplets ranged from 2092 s^{-1} to 2391 s^{-1}. For particle diameters from 0.35 µm – 2.0 µm, the airborne MP particle concentrations ranged from 4.2 l^{-1} to 348 l^{-1}. We determined size-dependent transfer factors for the water-air transfer and found a maximum for 1 µm particles at the lowest volume flow rate. For MP particles up to 1 µm diameter, the particle concentration in the jet droplets was enhanced compared to the bulk water concentration, indicating an enrichment of MP particles at the water-air-interface of bubbles.

Keywords: Airborne microplastic; Atmospheric microplastic; Jet drops; Particle ejection; Laboratory experiments

1. Introduction

Approximately 10^{18} – 10^{20} bubbles burst every second over the oceans (Ghabache and Séon 2016). Whitecaps and breaking waves are the main sources of gas bubbles in the oceans, with bubble sizes ranging from smaller than 0.1 mm to approximately 10 mm (Blanchard 1989). Bubble bursting generates sea spray aerosols (e.g. de Leeuw et al. 2011), and inorganic salts, organic matter as well as bacteria and viruses are efficiently transferred from the water into the air by this mechanism (e.g. Bigg and Leck 2008; Aller et al. 2005; Quinn et al. 1975). Since microplastic (MP) particles are abundant in ocean water (e.g. Chae et al. 2015), on shorelines (Browne et al. 2011) and in ocean sediment (Zhang et al. 2016), it can be expected that bubble bursting is an efficient process for MP transfer from oceans to the atmosphere (e.g. Allen et al. 2020; Liu et al. 2019; Trainic et al. 2020). Currently, this process is poorly quantified and recent studies such as Masry et al. (2021) point out that MP transfer rates must be quantified in order to evaluate the relevance of bubble bursting as a source for atmospheric MP. Atmospheric MP particles are considered a potential risk to human health (Prata 2018; Chen et al. 2020), and may also be a vector for toxic substances added or attached to MPs (Gallo et al. 2018, Cormier et al. 2021).

At the water surface, bubbles burst and film droplets develop from the surface water film when the bubble exceeds a diameter of approximately 2.4 mm (Spiel 1998). For bubbles with diameters less than 3 mm, the film simply rolls up (Spiel 1997). In addition, one or multiple jet drops are formed due to the collapse of the bubble cavity (Spiel 1998). The top jet drop is the largest and fastest drop (Ghabache und Séon 2016), and may be ejected up to a height of 20 cm for a ~2 mm bubble in salt water (Blanchard 1989).

It has been postulated that the composition of film drops is influenced by the sea surface microlayer, while jet drops represent the bulk water composition (e. g. Wang et al., 2017). The sea surface microlayer is known to be enriched in hydrophobic compounds such as organic matter (e. g. Wurl et al. 2011; Engel et al. 2017), and also MP particles (e. g. Song et al. 2014; Chae et al. 2015; Anderson et al. 2018). Anderson et al. (2018) report an enrichment of approximately an order of magnitude of MP particles in the sea surface microlayer of two estuaries compared to bulk water. Chae et al. (2015) observed higher MP particle concentrations in the sea surface microlayer compared to bulk surface water off the Western Korean coast, and Song et al. (2014) found a strongly enhanced MP concentration in the sea surface microlayer compared to conventional Manta trawl sampling off the Southern Korean coast.

Particles transported into the air via bubble bursting can be picked up by wind and transported to more distant areas (Allen et al. 2020). Various studies have detected atmospheric MP particles in remote areas (e.g. Allen et al. 2019; Brahney et al. 2020), and in recent years, further studies have identified MP in the marine boundary layer (Allen et al. 2020; Enyoh et al. 2019; Huang et al. 2020; Liu et al. 2019; Trainic et al. 2020; Zhang et al. 2020). Liu et al. (2019) sampled suspended atmospheric MP during a cruise in the west Pacific Ocean and detected 0 to 1.37 particles per cubic meter of air. Allen et al. (2020) sampled air in the marine boundary layer on the Atlantic coast in France. The average concentration of MP particles found in onshore winds was 2.9 m^{-3}, and in offshore winds 9.6 m^{-3}. Trainic et al. (2020) sampled aerosols in the North Atlantic Ocean and found atmospheric MP particles with estimated atmospheric residence times from 5 minutes to 2 days in 20 % of their samples.

In laboratory experiments, Quinn et al. (1975) studied the water-air transfer of 0.48 µm and 0.79 µm polystyrene particles in jet drops produced by bubble bursting. In their experiments, they found that

the particle concentration in the jet drop was basically independent of the particle concentrations in the bulk water but increased with the bubble age. Similarly, Sakai et al. (1988) found an enrichment of 0.5 µm latex particles in the surface layer of rising bubbles, and in jet drops. Recently, Masry et al. (2021) showed that 0.35 µm polystyrene particles are transferred from water to air by bubble bursting. They also used 0.6 µm and 1 µm particles but could not clearly confirm water-air transfer of these larger particles, and they conclude that MP particle transfer rates could bring answers to the significance of bubble bursting as a source of atmospheric MP particles.

We hypothesize that bubble bursting is an efficient water-air transfer process of MP particles. The goal of this study is to quantify the water-air transfer of MP particles as a function of particle size by the bubble bursting process. To this end, we present results of laboratory experiments with well-defined bubble populations and pre-defined concentrations of polystyrene (PS) particles with diameters between 0.35 µm and 2 µm suspended in water. Specifically, we quantify size-dependent transfer rates of MP particles, and analyze bubble and jet drop size distributions in order to contribute to estimating an order of magnitude of the global emission of atmospheric MP particles from oceans by bubble bursting.

2. Methods

2.1 General setup of bubble bursting experiments

We set up experiments to generate small gas bubbles in water with varying concentrations of MP particles, and to measure the number concentration of particles transferred from the water into the air due to the bubble bursting process (Figure 1).

A glass flask with a total volume of 2.45 l was filled with 1 l of deionized water (Seradest S750, conductivity $\kappa < 0.1$ µS cm^{-1}) at room temperature, resulting in a total water column of 8 cm. Deionized water was used to reduce the production of polydisperse airborne salt particles from drying droplets. The water volume was mixed with 0.1 to 5 ml of aqueous suspensions of monodisperse spherical polystyrene (PS) particles (Polybead Microspheres, Polysciences, Hirschberg an der Bergstraße, Germany) containing 2.5 % solids and a small amount of proprietary surfactant. The MP particle suspensions were filled into small glass vials, and defined volumes were taken with a microliter pipette. The pipette volume was discharged into the glass flask, and the water was stirred for uniform particle distribution. For each experiment, the total number of MP particles in 1 l of water N_w is the product of the particle concentration in suspension c_{susp} and the volume V_{susp} of the MP suspension (Table 1). The MP particle density of 1.05 g cm^{-3} is slightly larger than the density of water. Six different particle diameters in the diameter range from 0.35 µm to 2 µm (0.35 µm, 0.5 µm, 0.75 µm, 1 µm, 1.5 µm, 2 µm) were used. The particles contain a slight anionic surface charge from sulfate ester groups (Polysciences, 2022).

Before and after each experimental run, the glass flasks and all materials that came in contact with the water and the suspended MP particles were cleaned by thoroughly rinsing with deionized water and detergent, and again with deionized water. At the beginning of the experiment, the headspace with a volume of 1.45 l was free of particles. In order to produce gas bubbles, filtered (NY Simplepure syringe filter; 0.45 µm, Membrane Solutions LLC., USA) room air was pushed through a stainless-steel frit (IDEX

Health & Science, Oak Harbor, WA, USA) with a pore size of 20 µm with a peristaltic pump (IPS-8, Ismatec SA, Glattbrugg-Zürich, Switzerland) at two different pumping rates. The volume flow rates (VFR) corresponding to the two pumping rates measured with a bubble flowmeter (mini-Buck-Calibrator M-5, A.P. Buck Inc., Orlando, Florida, USA) were 182 $mm^3\ s^{-1}$ [50 RPM] and 353 $mm^3\ s^{-1}$ [90 RPM]. Depending on the pumping rate, this produced a population of small air bubbles. The mean bubble diameter ranged from 0.76 mm to 0.81 mm with maximum diameters up to 2.72 mm, thus producing mostly jet drops after bubble bursting. With a volume flow rate of 1.2 l min^{-1}, air was sampled from the headspace of the glass flask through a dryer to an optical particle counter. Droplets were dried with a Nafion diffusion dryer (Perma Pure MD-110-24S-4, Lakewood, NJ, USA) and introduced into an optical particle counter (OPC, Mini Laser Aerosol Spectrometer 11-R, Grimm Aerosoltechnik, Ainring, Germany) to measure the particle number size distribution in 31 size channels between 0.25 and 32 µm particle diameter with a sampling interval of 6 s. A HEPA filter was added to the glass flask to compensate for the difference of the OPC sample flow rate and the volume flow rate for generating gas bubbles.

In each experimental run, two experiments with different MP particle diameters were run in parallel. Two glass flasks were filled with deionized water taken out of a glass tank that was bottled before all experiments to guarantee the same water quality for all experiments. Suspensions of MP particles with different diameters were added to each flask according to Table 1. Both glass flasks were connected to the diffusion dryer and the OPC through a pinch valve that switched the sample flow between the two flasks every 15 minutes. In order to remove all particles from the headspace in the flasks, the OPC sampled air alternating every 15 min between the two glass flasks until the average total particle concentration was less than 1 l^{-1}. Then, the peristaltic pump was turned on for one hour with a VFR of 353 $mm^3\ s^{-1}$ before data was collected for evaluation. After the one-hour lead time, the experiments were carried out for 300 minutes with VFR 182 $mm^3\ s^{-1}$ and 353 $mm^3\ s^{-1}$, respectively. In total, this resulted in 10 x 15 min sample intervals per glass flask and VFR. For each particle diameter, two experiments with different particle concentrations in water were carried out.

2.2 Characterization of air bubbles and jet droplets

To analyze the size of bursting bubbles and of the resulting jet droplets, a standard camera (Panasonic DC-FZ82) with a resolution of 1280 pixel x 720 pixel and a frame rate of 100 frames per second was used in two different arrangements with illumination by high-power LEDs as shown schematically in the supplementary material (S1).

Perpendicular recording of the water surface allowed capturing the bursting bubbles (Fig. S1a). To block interference with lower ascending bubbles in the tank, the illumination was restricted to a thin layer on the surface by using LEDs in combination with slit blinds. Size was calibrated by using a benchmark at the surface level. The resulting video material was then analyzed using computer vision to detect and track individual bubbles. Thresholding the image based on brightness values resulted in isolated bubble contours. However, those contours can either belong to single bubbles or clustered bubbles and require segmentation. The segmentation was done following an algorithm proposed by Bettaieb et. al (2020). In contrast to the proposed algorithm, start and end points of individual segments were identified by convexity defects of the bubble shapes. As individual segments can belong to the same bubble, they were matched and recombined. This involved fitting circles to each segment. The fitted circles were then compared to every other segment in the corresponding cluster. Identical

geometry and center coordinates (within a tolerance of $\pm$25 % of bubble radius) indicated contour segments belonging to a single bubble. Estimated center coordinates and radii were matched between previous and current frames to track bubble movement. Matching was done by using a simple linear movement estimation from the last two frames of a bubble. The estimated position was then matched with the currently detected bubbles by shortest distance but limited by the maximum distance a bubble could reasonably travel within a frame to avoid wrong matching. Using this method allowed following individual bubbles until no more match was detected, indicating that the bubble of interest burst or merged. To check whether the lost bubble did in fact burst and not merge into a bubble nearby, all surrounding bubbles were checked for an increase in size. If so, the lost bubble was marked as merged and did not contribute to further analysis. According to Kočárková et. al (2013), the shape of small bubbles can be approximated by a sphere for Bond numbers up to 0.25, which corresponds to bubble diameters up to approximately 2.7 mm for air bubbles in water at room temperature.

For the characterization of jet droplets, the camera was directed at the setup from the front, and illumination with high-power LEDs from the side (Fig. S1b). Overlapping and merging of droplets was neglected. However, the droplet movement did produce motion blur in the image material. To estimate the droplet size, a rotated rectangle was fitted around the blurred droplets. Estimating that the deformation only occurred in the direction of movement, the shorter side of the rotated rectangle indicated the actual size of the droplet. Matching between previous and current frames was carried out as described above for bursting bubbles.

The image recognition of bursting bubbles and jet droplets in this study was technically limited. First, the detection of larger bubbles was limited under poor lighting conditions, which may have led to bubbles with diameters over 1 mm being detected as several smaller bubbles. This was caused by irregular illumination from point light sources on the edges of the rectangular basin. Second, the resolution of the video material was not sufficient to accurately detect and quantify the sizes of bubbles and droplets with diameters below 0.4 mm. The minimum resolved bubble diameter was 0.32 mm, and the minimum resolved droplet diameter was 0.15 mm. In general, the number of bubbles and droplets in the smallest size classes may be underestimated.

2.3 Data analysis and calculations

Airborne MP particle concentrations were calculated from the observed OPC particle size distributions. For each of the six particle diameters used in this study, the corresponding size channels of the OPC were evaluated individually. In each 15 min sampling interval, the first 5 min after switching the valve were discarded. The remaining data were averaged separately for each experiment and each VFR.

The expected airborne MP concentration were calculated assuming that particle input into the flask headspace is only from drying droplets generated by bubble bursting, and particles are removed from the flask headspace only with the sampling flow of the OPC. In this evaluation, there is no differentiation between film drops, the top jet drop and following secondary jet drops.

The input of airborne particles into the flask headspace, dN_{in}/dt [s^{-1}], is estimated by Eq. 1,

$$\frac{dN_{in}}{dt} = \frac{N_w}{V_w} \cdot Q_d \cdot f_{w-a} \quad [1],$$

with N_w, number of MP particles in the water volume V_w [mm^3], Q_d, rate of droplet volume generated per second [mm^3 s^{-1}], and $f_{w\text{-}a}$ a dimensionless transfer factor that takes into account the effective water – air transfer of MP particles.

The number of particles removed from the flask headspace per unit time, dN_{out}/dt, is estimated by Eq. 2,

$$\frac{dN_{out}}{dt} = \frac{N_a}{V_a} \cdot Q_a \quad [2],$$

with N_a, number of MP particles in the headspace volume V_a [mm^3], Q_a, air sampling flow rate [mm^3 s^{-1}].

For steady state conditions, the expected temporal evolution of the airborne MP concentration can be calculated by equating the particle input and output fluxes of Eqs. 1 and 2, and solving for N_a/V_a:

$$\frac{N_a}{V_a} = \frac{N_w}{V_w} \cdot \frac{Q_d}{Q_a} \cdot f_{w-a} \quad [3].$$

In this study, the water volume is V_w = 10^6 mm^3 (= 1 l), the headspace volume is V_a = 1.45 x 10^6 mm^3 (= 1.45 l), the sampling flow rate of the OPC is Q_a = 2 x 10^4 mm^3 s^{-1} (= 1.2 l min^{-1}), the number of MP particles in the water volume N_w is taken from Table 1, and the droplet volume generated per second, Q_d, is estimated from the observed bubble and droplet size distributions (section 3.1). Note that the transfer factor $f_{w\text{-}a}$ is unknown; it is determined by equating observed and expected airborne MP concentration acc. Eq. 3, N_a/V_a, and solving for $f_{w\text{-}a}$,

$$f_{w-a} = \frac{N_a}{V_a} \cdot \frac{V_w}{N_w} \cdot \frac{Q_a}{Q_d} \quad [4].$$

When switching the sampling flow between two glass flasks in 15 min intervals, the change in airborne MP particle number in one particular glass flask will be $N_{in} - N_{out}$ for the 15 min intervals when the sampling flow is taken from this glass flask, and N_{in} for the 15 min intervals when the sampling flow is taken from the other glass flask.

In order to estimate droplet size distributions from observed size distributions of bursting bubbles, the universal scaling law for the top jet drop diameter, D_{jet}, as a function of the bursting bubble diameter, D_{bub}, as given by Gañán-Calvo (2017) has been applied. The ambient droplet volume flux of ocean sea spray has been calculated using the parameterization by Andreas (1989) following environmental observations by Monahan et al. (1986). Details of the calculation method can be found in the supplementary material.

3. Results and discussion

3.1 Bursting bubbles and jet droplets - number concentrations and size distributions

Under two different conditions to generate bubbles, the number of bursting bubbles increased with increasing volume flow rates from 688 bursts per second at the low VFR (182 mm³ s^{-1}) to 1176 s^{-1} at the high VFR (353 mm³ s^{-1}) (Table 2). Thus, the rate of bursting bubbles at the low VFR was about 59 % of the high VFR.

While the number of bursting bubbles changed, the bubble size distributions were similar for the different VFRs. The mean diameter of the bursting bubbles ranged from 0.76 mm to 0.81 mm, with minimum diameters from 0.32 mm to 0.48 mm and maximum diameters from 2.40 mm to 2.72 mm

(Table 2). The fraction of bubbles with diameters larger than 2.4 mm was negligible, and therefore mostly jet drop production is expected in these experimental conditions. Overall, the modal diameter of the bursting bubbles was 0.6 mm. Figure S2 shows average bubble size distributions derived experimentally at two different VFRs.

In stationary behavior, the total volume of the observed bursting bubbles should correspond to the VFRs. As shown in Table 2, estimated burst volumes per second exceeded corresponding VFRs, indicating an overestimation of bubble number or size, especially with the lower VFR. Uncertainties might be introduced by the limited image resolution, which did not allow for precise segmentation of coalesced bubbles. Additionally, non-optimal lighting could lead to large bubbles being detected as several small bubbles. At higher VFR, the burst volume is only slightly overestimated, possibly due to the smaller relative influence of larger bubbles and coalescence with higher bubble counts.

With varying burst rates of bubbles and similar bubble size distributions, an increasing droplet number and similar droplet diameters generated by bubble bursting for the two different VFRs is expected. Table 2 shows the experimentally determined number of droplets generated per second, the mean and maximum droplet diameter, the average number of droplets generated per bubble and the droplet volume rate for low and high VFRs.

The droplet rate, i. e. the number of droplets generated per second, increases from 2092 s^{-1} to 2391 s^{-1}. Comparing with the burst rate, this yields on average 3.0 droplets per bursting bubble at VFR 182 mm^3 s^{-1} and 2.0 droplets per bubble at VFR 353 mm^3 s^{-1} (Table 2). Figure S3 shows average droplet size distributions for the two different VFRs. There is a slight increase in mean droplet diameter for the higher VFR from 0.42 mm to 0.48 mm, and the maximum droplet diameter increases from 2.46 mm to 4.15 mm. Consistent with the shift of the droplet size distribution to larger diameters, the mean volume of droplets generated per second increases from 159 mm^3 s^{-1} at low VFR to 340 mm^3 s^{-1} at high VFR.

The minimum droplet diameter that could be resolved with the experimental setup was 0.15 mm, and it was observed in both VFR setups. Taking into account Eqs. S4 and S5, the top jet droplet diameter can be expected to be smaller than 0.15 mm for bubble diameters smaller than 0.55 mm. Fig. S2 indicates that a considerable fraction of the bubble size distributions was smaller than 0.55 mm. The bubble size distributions can be used together with Eqs. S4 and S5 to calculate an independent estimate for the diameters of the top jet droplets resulting from the bursting bubbles. This calculation yields a mean diameter of the top jet droplet of 0.11 mm for VFR 182 mm^3 s^{-1}, and 0.10 mm for 353 mm^3 s^{-1}, respectively.

3.2 Airborne MP concentrations

In order to identify water-air transfer of MP particles due to bubble bursting, size distributions of airborne particles in the head space of the glass flask were measured and analyzed for concentration increases in the size range of interest. For example, Figure 2 shows average particle size distributions of two experiments with 0.35 µm and 1.0 µm MP particles. Significantly enhanced number concentrations can be seen in the diameter ranges around 0.35 µm and 1 µm, respectively.

The temporal evolution of the particle concentrations in the size channels corresponding to 0.35 µm and 1.0 µm diameter MP particles as well as the expected airborne MP concentration acc. Eqs. 1 – 3 are shown in Figure 3. The first 15 min interval shown both in Fig. 3a and b is when the OPC samples

air from the glass flask with 0.35 µm MP particles. When the valve switches to the other glass flask, indicated by a vertical red line, the observed particle concentration drops to much lower background concentrations in Fig. 3a, while the concentration of 1.0 µm MP particles shown in Fig. 3b increases. In this example, the average concentration of the 0.35 µm particles was 165 l^{-1} with a background concentration of 65 l^{-1}, and the average concentration of the 1.0 µm particles was 348 l^{-1} with a background concentration of less than 3 l^{-1}. While sampling from the other flask, particles produced by bubble bursting accumulate in the flask of interest, and when the valve switches back to this flask, a higher concentration is expected shown in Fig. 3 as green lines calculated acc. to Eq. 3. Within the uncertainties due to the limited counting statistics of the OPC, the expected decrease of the particle concentration while sampling from the flask is also visible in the observed concentrations.

Enhanced particle concentrations in the corresponding size ranges were detected in all experiments. Figure 4 shows average size distributions for five experiments with MP particle diameters from 0.35 µm to 1.5 µm, and one experiment with pure water without addition of MP suspensions. Local maxima of the particle size distributions can be clearly seen for the 0.35 µm, 0.5 µm, 0.75 µm and 1.0 µm MP particles. For larger MP particles, the concentration is still enhanced in the corresponding diameter range, but cannot be clearly distinguished from background measurements without MP particles. The size distributions in Figure 4 indicate that the bubble bursting process also produces particles in size ranges not corresponding with the added MP particle diameters. While larger particle diameters are expected due to MP particle agglomerates, particles in the smaller diameter ranges can be attributed to minuscule impurities of the water and the MP suspensions. Depending on the experiment, an increased background particle concentration was measured in the diameter range from 0.25 µm to 0.5 µm. In order to take into account this particle background, all experiments were run with two glass flasks in parallel with MP particles of two different diameters, and the particle concentration measured in the diameter range of interest in the flask without MP particles of this diameter was considered as a background concentration from the particle concentration.

The left part of Table 3 shows observed airborne particle concentrations of experiments with six different particle diameters and two different VFRs. As expected, the observed particle concentrations are higher in the experiments with higher VFR, and thus, higher burst rates and droplet production rates. The observed particle concentrations of 1.5 µm and 2 µm particles are in the range of background concentrations and should be treated with caution. Low concentrations of airborne MP particles of 1.5 µm and 2 µm diameter are expected due to the low concentration of these MP particles in the water volume (cf. Table 1). For experiments with MP particle diameters of 1 µm and less, the number concentration in water is close to 10^{11} l^{-1}, and changes in airborne MP particle number concentration indicate size-dependent effects in the water-air transfer.

3.3 Size-dependent MP particle transfer rates

Size-dependent water-air transfer factors of MP particles, $f_{w\text{-}a}$, are shown in Figure 5a and in Table 3. Transfer factors $f_{w\text{-}a}$ were calculated using Eq. 4, with the observed airborne MP concentrations given in Table 3, the water volume V_w = 1 l, the number of MP particles in the water volume N_w given in Table 1, and the sampling flow rate of the OPC, Q_a = 20 ml s^{-1}. The droplet volume generated per second, Q_d, can be estimated from the observed droplet size distributions (Fig. S3), or from the droplet size distribution calculated based on the observed bubble size distributions using Eqs. S4 and S5. The total droplet volume generated per second is given in Table 2. However, most large droplets are expected

to return to the water volume by gravitational settling before fully evaporating. Vertical air motion in the headspace of the glass flask is expected to be negligible, and the droplet settling time can be estimated by dividing the headspace height of 10 cm by the size-dependent gravitational settling velocity calculated acc. Eq. S6. For droplets larger than approximately 7 µm, the droplet settling time becomes smaller than the average residence time in the headspace, i.e. the headspace volume V_a = 1.45 l divided by the volume flow rate of the OPC, Q_a = 20 ml s^{-1}. Therefore, the total volume of droplets smaller than 7 µm was estimated by extrapolating the calculated droplet size distributions using a Weibull fit (Figure 5b), yielding $Q_{d,low}$ = 1.0 x 10^{-5} mm^3 s^{-1} and $Q_{d,high}$ = 2.3 x 10^{-4} mm^3 s^{-1} for the two different VFRs (indicated by the vertical broken line at 7 µm in Fig. 5b).

For each individual particle diameter, the experiments with the low VFR = 182 mm^3 s^{-1} (red squares in Fig. 5a) yield larger transfer factor than with the high VFR = 353 mm^3 s^{-1} (blue diamonds in Fig. 5a). This may be explained by a larger number of droplets at higher VFR that may collide with each other, leading to a shift of the droplet size distribution to larger sizes, and thus, to a larger fraction of large droplets that cannot efficiently transfer particles from water to air. Interestingly, for particle diameters up to 1.0 µm, transfer factors larger than 1 were observed. This means that the MP particle concentration in the droplet was enhanced compared to the bulk water concentration. These findings are consistent with earlier studies by Blanchard and Syzdek (1970) and Bezdek and Carlucci (1972), who found enhanced bacterial concentrations in jet drops due to enrichment of suspended bacteria at the water-gas interface of rising bubbles. Similarly, enrichment of suspended MP particles at the interface of rising bubbles was expected in the present experiment. It must be noted that the observed enrichment is likely an underestimation due to additional loss processes of airborne droplets and MP particles. For example, a fraction of 4.3 % to 5.1 % of the jet drops ejected by bubble bursting reached a height of 10 cm or more in the experiments at VFR = 182 mm^3 s^{-1} to VFR = 353 mm^3 s^{-1}, and thus, were lost to the glass flask wall. Clearly, the transfer factors shown in Figure 5a increase with increasing particle diameter from 0.35 µm to 1 µm but then decrease for larger particle diameters. This observation was made for both VFRs, and it suggests a size-dependent transfer efficiency of MP particles by bubble bursting. Under the experimental conditions of this study, MP particles with a diameter of 1.0 µm were transferred most efficiently from water to air by bubble bursting. The presented findings indicate that particle enrichment at the interface of the rising bubble, bubble size distributions and the resulting droplet size distributions, as well as droplet and particle loss processes in the glass flask contribute to changes of the observed airborne MP particle concentrations. All these factors are expected to be dependent on particle size to varying degrees, and it is not possible to fully disentangle these size-dependencies within this study. It must be noted that the transfer efficiency in natural waters, as mentioned before, is certainly affected by additional factors such as water salinity or surface tension, and the presented results should not be applied directly to natural systems. Nevertheless, we conclude that particle size must be taken into account in future parameterizations of the water-air transfer of MP particles.

Recently, Masry et al. (2021) demonstrated in their lab experiments that bubble bursting can transfer MP particles from water to air. Their experimental setup was similar to ours in principle, however, with larger water and air volumes, uncharacterized bubble and droplet size spectra but probably a much larger droplet volume compared to this study. In their study, they clearly demonstrate the water-air transfer of 0.35 µm polystyrene particles but do not distinguish 0.6 and 1.0 µm particles from background concentrations. Given the fact that the water-air transfer factor was highest for 1.0 µm polystyrene particles in the present study, it can be speculated that the water concentration of the

larger particles was too low, similar to the experiments using 1.5 and 2.0 µm particles in this study. Also, it must be noted that the observed decrease in transfer factors with increasing volume flow rates for droplet generation suggests less effective water-air transfer of particles in their experiments in general.

4. Environmental implications

Even though the presented size-dependent water-air transfer factors of MP particles are associated with large uncertainties, this study corroborates the hypothesis that oceans (and other water bodies) can act as sources of airborne MP particles. Recently, Lehmann et al. (2021) investigated the ejection of marine MP particles by impacting rain droplets, and estimated an upper boundary on the order of 10^{14} MP particles ejected per year globally. Due to the large uncertainties associated with MP particle concentrations in marine waters, we do not attempt to extrapolate our findings to the global scale but rather compare the relative efficiency of the water-air-transfer processes due to impacting rain droplets and due to bubble bursting. To do this, we follow the assumptions made in Lehmann et al. (2021) and only consider jet droplets with diameters less than 200 µm which are likely to be picked up by wind under the typical conditions at the ocean surface. The ocean surface water concentration of MP particles assumed in this calculation is 2.9 MP particles per liter according to Choy et al. (2019). It must be noted that MP concentrations in the oceans are expected to greatly vary in time and space, and MP particles with diameters < 2 µm used in this study are certainly underrepresented in measurements of sea surface MP concentrations due to limitations in sampling and analytical techniques. Since most film droplets are smaller than 1 µm in radius (e.g. de Leeuw et al. 2011), it can be expected that supermicron MP particles are transferred from water to air mostly by jet droplets. In order to estimate ambient droplet volume fluxes of ocean sea spray, we use Eqs. S7 – S10 to calculate a jet droplet size distribution at formation according to Andreas (1989) and Monahan et al. (1986). With an average wind speed of 8 m s^{-1}, we calculate a total volume flux of droplets smaller than 200 µm diameter of 1.5 x 10^{-8} l m^{-2} s^{-1}. With the total surface area of the global oceans of approximately 3.61 x 10^{14} m^2, the total annual droplet volume flux is 1.7 x 10^{14} l.

Assuming the particle concentration in jet droplets is the same as in the bulk water at the ocean surface, this results in an estimated 5 x 10^{14} particles per year globally that are ejected in droplets small enough to potentially enter atmospheric uptake by wind. Taking into account the enrichment of MP particles in jet droplets, with an MP concentration in jet droplets enhanced by a factor ranging from 3 (as observed in this study) to 200 (as observed e.g. by Sakai et al., 1988), the global emission estimate of MP particles from oceans by bubble bursting ranges from 10^{15} to 10^{17} particles per year. This is 10 to 1000 times larger than the estimated 10^{14} MP particles per year by impacting raindrops. There are various uncertainties affecting these estimates including (1) uncertainties of the size distribution of the jet droplets, (2) the rate of bursting bubbles on global oceans, and (3) MP concentrations in the ocean surface water. Nevertheless, this estimate indicates that bubble bursting is an effective water-air transfer process for MP particles in the environment.

5. Conclusions

The results of this study demonstrate that submicron MP particles are readily transferred from water to air by bubble bursting. This supports the hypothesis that oceans (and other water bodies) can act as sources of atmospheric MP particles. Compared with the ejection of MP particles by impacting rain droplets, bubble bursting is estimated to be more efficient by a factor of 10 to 1000. The experimentally determined water-air transfer factors are dependent on particle size and the droplet volume flow rate, with a maximum for 1 µm particles at low volume flow rate. For particle diameters up to 1.0 µm, MP particle concentrations are found to be larger in the droplet than in the bulk water. The enhanced concentration of MP particles in droplets compared to the bulk water concentration can be explained by enrichment of MP particles at the water-gas interface of rising bubbles, as previously discussed by Quinn et al. (1975) and Sakai et al. (1988). Hence, bubble bursting may not only be a highly efficient transfer mechanism for MP particles from water to air but also, rising gas bubbles may be an important vertical transport process of MP particles in the water column, potentially contributing to the enrichment of MP particles in the sea surface microlayer. While the water-air transfer of MP particles with diameters larger than 1.0 µm could not be quantified, it should be noted that this is very likely due to experimental limitations such as low particle concentrations, relatively high background concentrations, a low sampling flow rate, and particle losses in the experimental setup. Therefore, the presented experiments do not allow conclusions regarding the water-air transfer of MP particles larger than 1.0 µm. In future work, it is also important to investigate the effects of surfactants and water salinity on the bubble bursting process and the water-air transfer of MP particles, which is expected to play a role in natural waters (Masry et al. 2021).

Authors' contributions

LMO did the conceptualization, methodology, investigation and writing. SS was responsible for methodology, investigation and writing. ML made the formal analysis and was writing. SG contributed the funding acquisition and did review & editing. AH was responsible for project administration, funding acquisition, conceptualization, writing and the resources. LMO, SS and AH contributed to the visualisation. SS, ML, SG and AH made the review.

Funding

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 391977956 – SFB1357.

Availability of data and materials

The datasets used and/or analysed during the current study are available from the corresponding author on reasonable request.

Competing interests

The authors declare that they have no competing interests.

References

Allen, S., Allen, D., Phoenix, V. R., Le Roux, G., Durántez Jiménez, P., Simonneau, A., Binet, S., Galop, D., 2019. Atmospheric transport and deposition of microplastics in a remote mountain catchment. Nat. Geosci. 12, 339–344. DOI: 10.1038/s41561-019-0335-5

Allen, S., Allen, D., Moss, K., Le Roux, G., Phoenix, V. R., Sonke, J. E., 2020. Examination of the ocean as a source for atmospheric microplastics. PLoS One 15 (5): e0232746. DOI: 10.1371/journal.pone.0232746.

Aller, J. Y., Kuznetsova, M. R., Jahns, C. J., Kemp, P. F., 2005. The sea surface microlayer as a source of viral and bacterial enrichment in marine aerosols. J. Aerosol Sci. 36, 801–812

Anderson, Z.T., Cundy, A.B., Croudace, I.W., Warwick, P.E., Celis-Hernandez, O., Stead, J.L., 2018. A rapid method for assessing the accumulation of microplastics in the sea surface microlayer (SML) of estuarine systems. Sci. Rep. 8, 9428.

Andreas, E.L., 1989. Thermal and size evolution of sea spray droplets. CRREL report 89-11, US Army Corps of Engineers Cold Regions Research & Engineering Laboratory, Hanover, NH.

Bettaieb, A., Filali, A., Filali, T., Aissia, H.B., 2020. An efficient algorithm for overlapping bubbles segmentation. Comput. Opt. 44, 363–374. DOI: 10.18287/2412-6179-CO-605.

Bezdek H. F. and Carlucci A. F., 1972. Surface concentration of marine bacteria. Limnol. Oceanogr. 17, 566-569.

Bigg, E.K., Leck, C., 2008. The composition of fragments of bubbles bursting at the ocean surface. J. Geophys. Res. 113, D11209.

Blanchard D. C. and Syzdek L., 1970. Mechanism for the water-to-air transfer and concentration of bacteria. Science 170, 626-628.

Blanchard, D. C., 1989. The Ejection of Drops from the Sea and Their Enrichment with Bacteria and Other Materials: A Review. Estuaries 12 (3), 127–137.

Brahney, J., Hallerud, M., Heim, E., Hahnenberger, M., Sukumaran, S., 2020. Plastic rain in protected areas of the United States. Science 368 (6496), 1257–1260.

Browne, M. A., Crump, P., Niven, S. J., Teuten, E., Tonkin, A., Galloway, T., Thompson, R., 2011. Accumulation of microplastic on shorelines woldwide: sources and sinks. Environ. Sci. Technol. 45, 9175–9179. DOI: 10.1021/es201811s.

Chae, D.-H., Kim, I.-S., Kim, S.-K., Song, Y.K., Shim, W.J., 2015. Abundance and distribution characteristics of microplastics in surface seawaters of the Incheon/Kyeonggi coastal region. Arch. Environ. Contam. Toxicol. 69, 269-278.

Chen, L., Li, L., Li, Z., Zhang, K., 2017. Submillimeter-sized bubble entrapment and high-speed jet during droplet impact on solid surfaces. Langmuir 33. DOI: 10.1021/acs.langmuir.7b01506.

Chen, G., Feng, Q., Wang, J., 2020. Mini-review of microplastics in the atmosphere and their risks to humans. Sci. Total Environ. 703, DOI: 10.1016/j.scitotenv.2019.135504

Choy, C.A., Robison, B.H., Gagne, T.O., Erwin, B, Firl, E., Halden, R.U., Hamilton, J.A., Katija, K., Lisin, S.E., Rolsky, C., Van Houtan, K.S, 2019. The vertical distribution and biological transport of marine microplastics across the epipelagic and mesopelagic water column. Sci. Rep. 9, 7843. DOI: 10.1038/s41598-019-44117-2

Cormier, B., Gambardella, C., Tato, T., Perdriat, Q., Costa, E., Veclin, C., Le Bihanic, F., Grassl, B., Dubocq, F., Kärrman, A., Van Arkel, K., Lemoine, S., Lagarde, F., Morin, B., Garaventa, F., Faimali, M., Cousin, X., Bégout, M.-L., Beiras, R., Cachot, J., 2021. Chemicals sorbed to environmental microplastics are toxic to early life stages of aquatic organisms. Ecotoxicology and Environmental Safety, Volume 208, 111665, DOI: 10.1016/j.ecoenv.2020.111665.

de Leeuw, G., Andreas, E. L., Anguelova, M. D., Fairall, C. W., Lewis, E. R., O'Dowd, C., Schulz, E., Schwartz, S. E., 2011. Production flux of sea spray aerosol. Rev. Geophys. 49 (RG2001). DOI: 10.1029/2010RG000349.

Engel, A., Bange, H.W., Cunliffe, M., Burrows, S.M., Friedrichs, G., Galgani, L., Herrmann, H., Hertkorn, N., Johnson, M., Liss, P.S., Quinn, P.K., Schartau, M., Soloviev, A., Stolle, C., Upstill-Goddard, R.C., van Pinxteren, M., Zäncker, B., 2017. The ocean's vital skin: toward an integrated understanding of the sea surface microlayer. Front. Mar. Sci. 4, 165.

Enyoh, C.E., Verla, A.W., Verla, E.N., Ibe, F.C., Amaobi, C.E., 2019. Airborne microplastics: a review study on method for analysis, occurrence, movement and risks. Environ. Monit. Assess. 191, 290–293. DOI: 10.1016/j.marpolbul.2016.01.006.

Gallo, F., Fossi, C., Weber, R., Santillo, D., Sousa, J., Ingram, I., Nadal, A., Romano, D., 2018. Marine litter plastics and microplastics and their toxic chemicals components: the need for urgent preventive measures. Environ. Sci. Eur. 30, 13. DOI: 10.1186/s12302-018-0139-z

Gañán-Calvo, A.M., 2017. Revision of bubble bursting: Universal scaling laws of top jet drop size and speed. Physical Review Letters 119, 204502.

Ghabache, E., Séon, T., 2016. Size of the top jet drop produced by bubble bursting. Phys. Rev. Fluids 1. DOI: 10.1103/PhysRevFluids.1.051901.

Huang, Y., Qing, X., Wang, W., Han, G., Wang, J., 2020. Mini-review on current studies of airborne microplastics: Analytical methods, occurrence, sources, fate and potential risk to human beings. TrAC Trends Analyt. Chem. 125, 115821.

Kočárková, H., Rouyer, F., Pigeonneau, F., 2013. Film drainage of viscous liquid on top of bare bubble: Influence of the Bond number. Phys. Fluids 25. DOI: 10.1063/1.4792310.

Lehmann, M., Oehlschlägel, L.M., Häusl, F.P., Held, A., Gekle, S., 2021. Ejection of marine microplastics by raindrops: a computational and experimental study. Microplastics and Nanoplastics 1, 18. DOI: 10.1186/s43591-021-00018-8

Liu, K., Wu, T., Wang, X., Song, Z., Zong, C., Wei, N., Li, D., 2019. Consistent Transport of Terrestrial Microplastics to the Ocean through Atmosphere. Environ. Sci. Technol. 53, 10612–10619. DOI: 10.1021/acs.est.9b03427.

Masry, M., Rossignol, S., Temime Roussel, B., Bourgogne, D., Bussière, P.-O., R'mili, B., Wong-Wah-Chung, P., 2021. Experimental evidence of plastic particles transfer at the water-air interface through bubble bursting. Environ. Pollut. 280, 116949. DOI: 10.1016/j.envpol.2021.116949.

Monahan, E.C., Spiel D.E., Davidson, K.L., 1986. A model of marine aerosol generation via whitecaps and wave disruption. Monahan, E.C., Mac Niocaill, G. (Eds.) Oceanic whitecaps and their role in air–sea exchange processes. Springer, Dordrecht, 167-174. DOI: 10.1007/978-94-009-4668-2

Polysciences (2022) https://www.polysciences.com/german/polybead-microspheres-035956m. Last access on 20 June 2022

Prata, J.C., 2018. Airborne microplastics: consequences to human health? Environ. Pollut. 234, 115e126. DOI: 10.1016/j.envpol.2017.11.043.

Quinn, J.A., Steinbrook, R.A., Anderson, J.L., 1975. Breaking bubbles and the water-to-air transport of particulate matter. Chem. Eng. Sci. 30, 1177-1184.

Sakai, M., Tanaka, A., Egawa, H. & Sugihara, G., 1988. Enrichment of suspended particles in top jet drops from bursting bubbles J. Coll. Interface Sci. 125, 428-436.

Song, Y.K., Hong, S.H., Jang, M., Kang, J.-H., Kwon, O.Y., Han, G.M., Shim, W.J., 2014. Large accumulation of micro-sized synthetic polymer particles in the sea surface microlayer. Environ. Sci. Technol. 48, 9014-9021.

Spiel, D. E., 1995. On the births of jet drops from bubbles bursting on water surfaces. J. Geophys. Res. 100, 4995–5006, DOI: 10.1029/94JC03055.

Spiel, D. E., 1997. A hypothesis concerning the peak in film drop production as a function of bubble size. J. Geophys. Res. 102, 1153–1161.

Spiel, D. E., 1998. On the births of film drops from bubbles bursting on seawater surfaces. J. Geophys. Res. 103, 24907-24918.

Trainic, M., Flores, J. M., Pinkas, I., Pedrotti, M. L., Lombard, F., Bourdin, G., Gorsky, G., Boss, E., Rudich, Y., Vardi, A., Koren, I., 2020. Airborne microplastic particles detected in the remote marine atmosphere. Commun. Earth Environ. 1:64. DOI: 10.1038/s43247-020-00061-y.

Wang, X., Deane, G.B., Moore, K.A., Ryder, O.S., Stokes, M.D., Beall, C.M., Collins, D.B., Santander, M.V., Burrows, S.M., Sultana, C.M., Prather, K.A., 2017. The role of jet and film drops in controlling the mixing state of submicron sea spray aerosol particles. Proc. Natl. Acad. Sci. 114, 6978-6983.

Wurl, O., Wurl, E., Miller, L., Johnson, K., Vagle, S., 2011. Formation and global distribution of sea-surface microlayers. Biogeosciences 8, 121-135.

Zhang, K., Su, J., Xiong, X., Wu, X., Wu, C., Liu, J., 2016. Microplastic pollution of lakeshore sediments from remote lakes in Tibet plateau, China. Environ. Pollut. 219, 450–455. DOI: 10.1016/j.envpol.2016.05.048.

Zhang, Y., Kang, S., Allen, S., Allen, D., Gao, T., Sillanpää, M., 2020. Atmospheric microplastics: A review on current status and perspectives. Earth Sci. Rev. 203, 103118. DOI: 10.1016/j.earscirev.2020.103118.

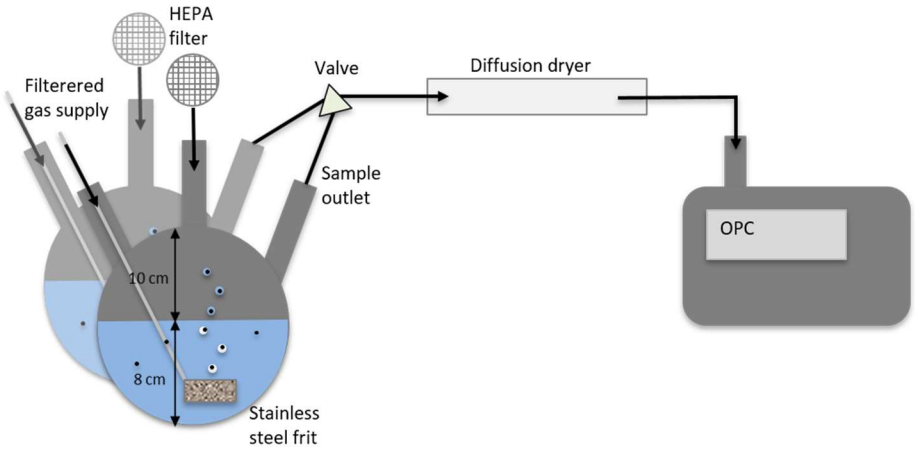


Figure 1: Schematic representation of the experimental setup including two glass flasks with filtered gas supply and a stainless-steel frit to produce bubbles (white) in water with MP particles (black), and droplets (blue) in the headspace. Sample air is transferred through a diffusion dryer to an optical particle counter (OPC). The valve is switching between the two flasks every 15 minutes.

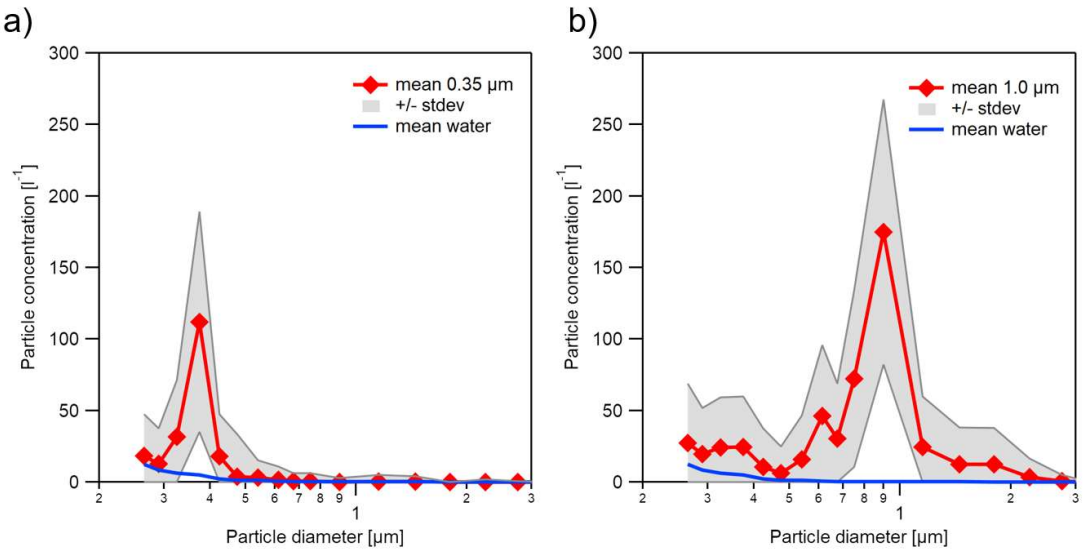


Figure 2: Particle size distribution for experiments with (a) 0.35 µm diameter MP particles and (b) 1.0 µm diameter MP particles, VFR = 353 mm^3 s^{-1}.

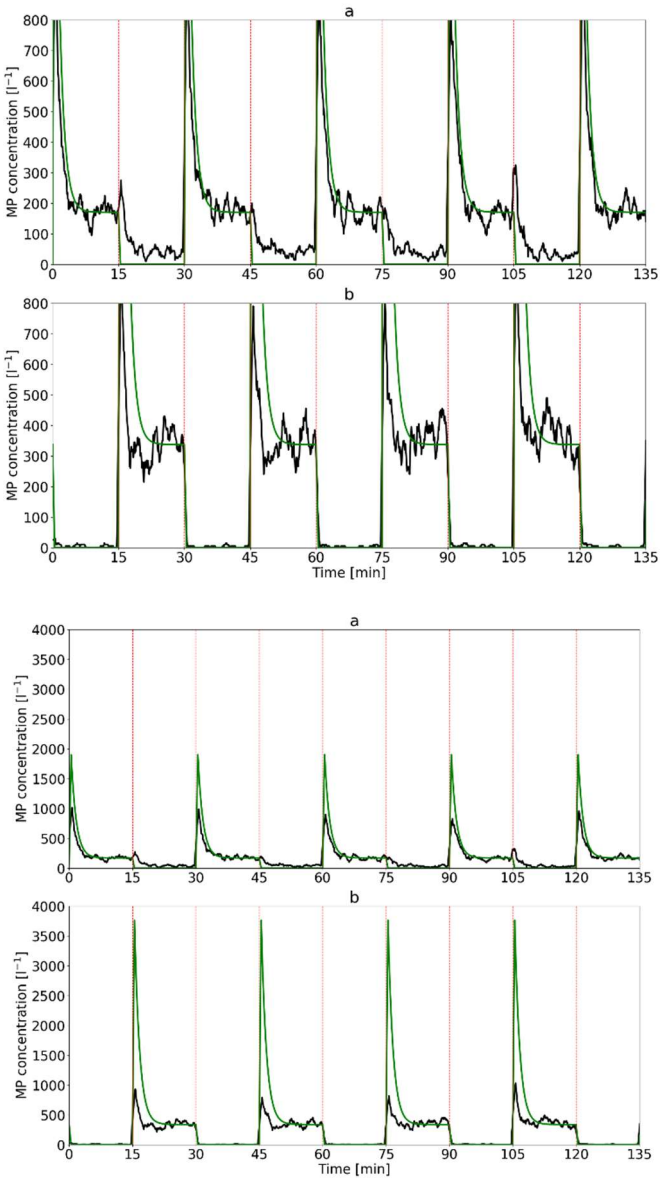


Figure 3: Temporal evolution of the observed airborne concentration (black line) corresponding to (a) 0.35 µm and (b) 1.0 µm diameter MP particles, and the expected airborne MP concentration acc. Eqs. 1 and 2 (green line). Vertical dashed red lines indicate valve switching between two glass flasks. The two experiments were carried out in parallel.

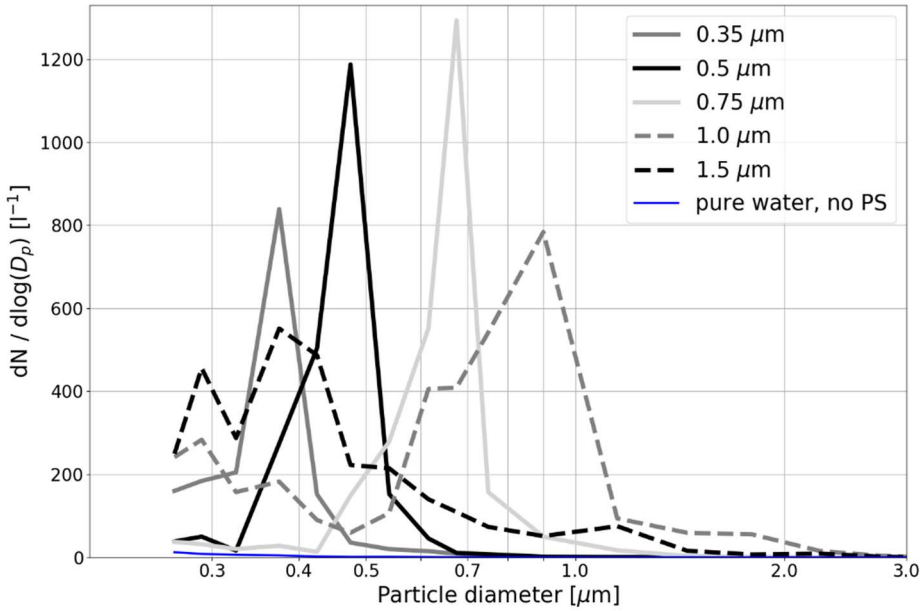


Figure 4: Particle size distributions with MP particle diameters from 0.35 µm to 1.0 µm (black and grey) and pure water particle concentration (blue), VFR = 353 mm^3 s^{-1}.

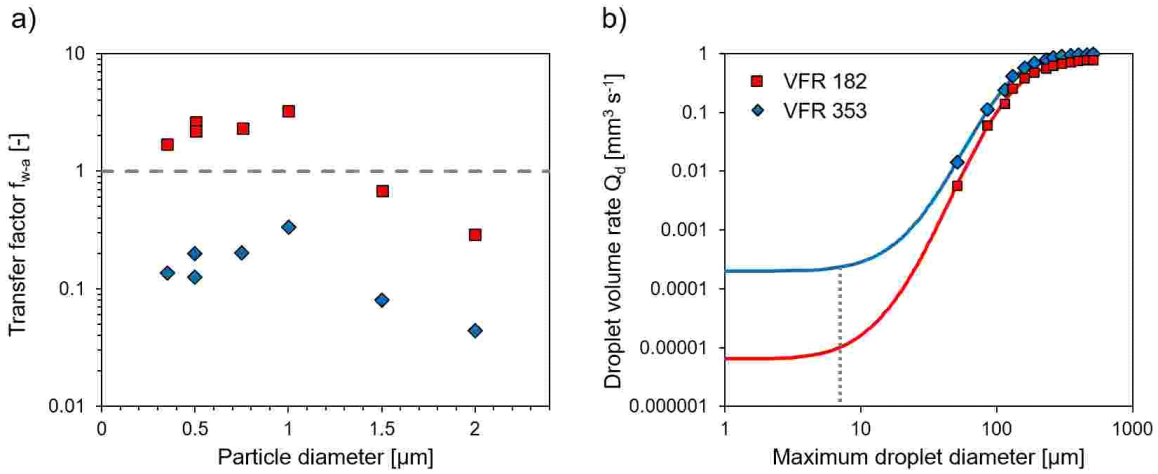


Figure 5: a) Water-air transfer factors f_{w-a} as a function of particle diameter in the range from 0.35 µm to 2.0 µm, and b) droplet volume Q_d generated per second by all droplets with the given maximum droplet diameter. Results for low volume flow rates are shown in red, for high volume flow rates in blue.

Table 1: Particle size, concentration of particles in suspension c_{susp}, supplied volume of suspension V_{susp} and number of particles in water N_w.

Particle diameter D_p [µm]	***Particle concentration in suspension c_{susp} [ml^{-1}]***	***Suspension volume V_{susp} [ml]***	***Number of particles in water N_w***
0.35 ± 0.035	1.06 x 10^{12}	0.1	1.06 x 10^{11}
0.5 ± 0.05	3.64 x 10^{11}	0.3	1.09 x 10^{11}
		0.9	3.28 x 10^{11}
0.75 ± 0.075	1.08 x 10^{11}	1	1.08 x 10^{11}
1.0 ± 0.1	4.55 x 10^{10}	2	9.10 x 10^{10}
1.5 ± 0.15	1.35 x 10^{10}	2	2.70 x 10^{10}
2.0 ± 0.2	5.68 x 10^{9}	5	2.84 x 10^{10}

Table 2: Overview of experimentally determined bubble burst rates, minimum, mean and maximum bubble diameters, volume of bursting bubbles and droplet production rates, mean and maximum droplet diameters, average number of droplets per bursting bubble, and volume of droplets generated per second for low and high VFR.

	low	high
VFR *[mm³ s^{-1}]*	182	353
Burst rate [s^{-1}]	688	1176
$D_{bub, min}$ [mm]	0.48	0.32
$D_{bub, mean}$ [mm]	0.81	0.76
$D_{bub, max}$ [mm]	2.40	2.56
Burst volume rate [mm³ s^{-1}]	265	362
Droplet rate [s^{-1}]	2092	2391
$D_{drop, mean}$ [mm]	0.42	0.48
$D_{drop, max}$ [mm]	2.46	4.15
Droplet volume rate Q_d [mm³ s^{-1}]	159	340
Droplets per bubble [-]	3.0	2.0

Table 3: Observed MP particle concentrations in air for six different particle diameters D_p at VFR = 182 mm³ s^{-1} and VFR = 353 mm³ s^{-1}. Results for 0.5 µm particle diameter are for experiment with total number of 1.09 x 10^{11} particles in water volume.

Particle diameter [µm]	MP concentration [l^{-1}]		Transfer factor $f_{w\text{-}a}$	
	VFR 182 mm^3 s^{-1}	VFR 353 mm^3 s^{-1}	VFR 182 mm^3 s^{-1}	VFR 353 mm^3 s^{-1}
0.35	88.6	165.2	1.67	0.14
0.5	143.3	248.8	2.63	0.20
0.75	126.0	252.4	2.33	0.20
1	147.2	348.3	3.24	0.33
1.5	9.2	24.6	0.68	0.08
2	4.2	14.3	0.29	0.04

8.8 Publication 8

Combined scientific CFD simulation and interactive raytracing with OpenCL

by

Moritz Lehmann

International Workshop on OpenCL. (2022), pp. 1–2
http://dx.doi.org/10.1145/3529538.3529542
https://youtu.be/9q3lXsrVF30

Combined scientific CFD simulation and interactive raytracing with OpenCL

Moritz Lehmann
moritz.lehmann@uni-bayreuth.de
University of Bayreuth
Germany

ABSTRACT

One of the main uses for OpenCL is (scientific) compute applications where graphical rendering is done externally, after the simulation has finished. However separating simulation and rendering has many disadvantages, especially the extreme slowdown caused by copying simulation data from device to host, and needing to store raw data on the hard drive, taking up hundreds of gigabyte, just to visualize preliminary results. A much faster approach is to implement both simulation and rendering in OpenCL. The rendering kernels have direct read-only access to the raw simulation data that resides in ultra-fast GPU memory. This eliminates all PCIe data transfer but camera parameters and finished frames, allowing for interactive visualization of simulation results in real time while the simulation is running. This is an invaluable tool for rapid prototyping. Although OpenCL does not have existing functionality for graphical rendering, being a general compute language, it allows for implementing an entire graphics engine, such that no data has to be moved to the CPU during rendering. On top, specific low-level optimizations make this OpenCL graphics engine outperform any existing rendering solution for this scenario, enabling drawing billions of lines per second and fluid raytracing in real time on even non-RTX GPUs. This combination of simulation and rendering in OpenCL is demonstrated with the software FluidX3D [3] - a lattice Boltzmann method (LBM) fluid dynamics solver.

The first part will briefly introduce the numerical method for simulating fluid flow in a physically accurate manner. After introducing the LBM, the optimizations to make it run at peak efficiency are discussed: Being a memory-bound algorithm, coalesced memory access is key. This is achieved through array-of-structures data layout as well as the one-step-pull scheme, a certain variant of the LBM streaming step. One-step-pull leverages the fact that the misaligned read penalty is much smaller than the misaligned write penalty on almost all GPUs. Roofline analysis shows that with these optimizations, the LBM runs at 100% efficiency on the fastest data-center and gaming GPUs [5]. To simulate free surface flows, the LBM is extended with the Volume-of-Fluid (VoF) model. An efficient algorithm has been designed to vastly accelerate the challenging surface tension computation [4]. This extremely efficient VoF-LBM GPU implementation allows covering new grounds in science: FluidX3D has been used to simulate more than 1600 raindrop impacts to statistically evaluate how microplastics transition from the ocean surface into the atmosphere when the spray droplets are generated during drop impact [6]. At the same power consumption, with existing CPU-parallelized codes, compute time would have been several years, whilst with FluidX3D it was about a week.

The second part will focus on real time rendering with OpenCL, especially raytracing. Rasterization on the GPU is parallelized not over pixels but lines/triangles instead, making runtime mostly independent of screen resolution and lightning fast. Each line/triangle is transformed with the camera parameters from 3D to 2D screen coordinates and then rasterized onto the frame (integer array) with Bresenham algorithm [2] and z-buffer. The raytracing graphics are based on a combination of fast ray-grid traversal and marching-cubes, leveraging that the computational grid from the LBM already is an ideal acceleration structure for raytracing. The idea of raytracing is simple: Through each pixel on the screen, shoot a reverse light ray out of the camera and see where it intersects with a surface in the scene. Then (recursively) calculate reflected/refracted rays and mix the colors. If a ray doesn't intersect with anything, its color is determined by the skybox image via UV mapping and bilinear pixel interpolation. With mesh surfaces consisting of many triangles, computation time quickly becomes a problem, as for each ray all triangles have to be tested for intersection. To overcome this, an acceleration structure is required. While computer games often use a bounding volume hierarchy, the LBM already provides an ideal alternative acceleration structure: the simulation grid. The corresponding algorithm is called ray-grid traversal: When a ray shoots through the 3D grid, intersections with the surface only have to be checked for at each traversed grid cell rather than the entire grid. In each traversed grid cell, the 0-5 surface triangles are generated on-the-fly with the marching-cubes algorithm and ray-triangle intersections are checked with the Möller-Trumbore algorithm. If an intersection has been found, only afterwards the normals are calculated on the 8 grid points spanning the cell, and are trilinearly interpolated to the intersection coordinates. The so interpolated surface normal makes the raytraced surface appear perfectly smooth. On the GPU, the ray(s) for each pixel on screen are computed in parallel, vastly speeding up rendering. It is of key importance how to align the OpenCL workgroups on the 2D array of screen pixels: best performance is achieved for 8x8 pixel tiles; this is about 50% faster than 64x1 tiles, because with small, square-ish tiles, all rays of the workgroup are more likely to traverse the same grid cells, greatly improving memory broadcasting. In ray-grid traversal, 8 isovalues spanning a cell have to be loaded from GPU memory for each traversed cell. Once the triangle intersection has been found, the gradient on each of the 8 cell isovalues is calculated with central differences. Instead of loading an additional 6 isovalues

IWOCL'22, May 10–12, 2022, Bristol, United Kingdom, United Kingdom

ACM ISBN 978-1-4503-9658-5/22/05.
https://doi.org/10.1145/3529538.3529542

for each of the 8 grid points, their isovalues are reused such that only 24 additional isovalues are loaded. For marching-cubes, the algorithm by Paul Bourke [1] is implemented in OpenCL. With 16-/8-bit integers, bit-packing and symmetry, the tables are reduced to 1/8 of their original size and stored in constant memory space. For computing the cube index, branching is eliminated by bit operations. The Möller-Trumbore algorithm [7] is implemented in an entirely branchless manner.

This raytracing implementation is fast enough to run in real time for even the largest lattice dimensions that fit into the memory of a GPU. Finally, the combined VoF-LBM simulation and raytracing implementation is demonstrated on the most realistic simulation of an impacting raindrop ever done [8].

CCS CONCEPTS

• **Computing methodologies** → **Parallel programming languages**.

KEYWORDS

lattice Boltzmann method, Volume-of-Fluid, raytracing, rasterization, graphics, OpenCL, GPU, CFD, fluid, marching-cubes, ray-grid traversal, raindrop

ACM Reference Format:
Moritz Lehmann. 2022. Combined scientific CFD simulation and interactive raytracing with OpenCL. In *International Workshop on OpenCL (IWOCL'22), May 10–12, 2022, Bristol, United Kingdom, United Kingdom.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3529538.3529542

REFERENCES

[1] [n.d.]. Polygonising a scalar field. http://www.paulbourke.net/geometry/polygonise/
[2] J. E. Bresenham. 1965. Algorithm for computer control of a digital plotter. *IBM Systems Journal* 4, 1 (1965), 25–30. https://doi.org/10.1147/sj.41.0025
[3] Moritz Lehmann. 2021. High Performance Free Surface LBM on GPUs. https://epub.uni-bayreuth.de/5400/
[4] Moritz Lehmann and Stephan Gekle. 2022. Analytic Solution to the Piecewise Linear Interface Construction Problem and Its Application in Curvature Calculation for Volume-of-Fluid Simulation Codes. *Computation* 10, 2 (2022). https://doi.org/10.3390/computation10020021
[5] Moritz Lehmann, Mathias J. Krause, Giorgio Amati, Marcello Sega, Jens Harting, and Stephan Gekle. 2021. On the accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit and novel 16-bit number formats. https://doi.org/10.48550/ARXIV.2112.08926
[6] Moritz Lehmann, Lisa Marie Oehlschlägel, Fabian P. Häusl, Andreas Held, and Stephan Gekle. 2021. Ejection of marine microplastics by Raindrops: A computational and experimental study - microplastics and Nanoplastics. https://doi.org/10.1186/s43591-021-00018-8
[7] Tomas Möller and Ben Trumbore. 1997. Fast, Minimum Storage Ray-Triangle Intersection. *Journal of Graphics Tools* 2, 1 (1997), 21–28. https://doi.org/10.1080/10867651.1997.10487468 arXiv:https://doi.org/10.1080/10867651.1997.10487468
[8] ProjectPhysX. [n.d.]. *FluidX3D - Defeating the Lattice Boltzmann Method (Esoteric Twist + FP32/FP16 on GPU).* YouTube. https://youtu.be/1g_zFsvScME

(Eidesstattliche) Versicherungen und Erklärungen

(§ 9 Satz 2 Nr. 3 PromO BayNAT)

Hiermit versichere ich eidesstattlich, dass ich die Arbeit selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel benutzt habe (vgl. Art. 64 Abs. 1 Satz 6 BayHSchG).

(§ 9 Satz 2 Nr. 3 PromO BayNAT)

Hiermit erkläre ich, dass ich die Dissertation nicht bereits zur Erlangung eines akademischen Grades eingereicht habe und dass ich nicht bereits diese oder eine gleichartige Doktorprüfung endgültig nicht bestanden habe.

(§ 9 Satz 2 Nr. 4 PromO BayNAT)

Hiermit erkläre ich, dass ich Hilfe von gewerblichen Promotionsberatern bzw. -vermittlern oder ähnlichen Dienstleistern weder bisher in Anspruch genommen habe noch künftig in Anspruch nehmen werde.

(§ 9 Satz 2 Nr. 7 PromO BayNAT)

Hiermit erkläre ich mein Einverständnis, dass die elektronische Fassung meiner Dissertation unter Wahrung meiner Urheberrechte und des Datenschutzes einer gesonderten Überprüfung unterzogen werden kann.

(§ 9 Satz 2 Nr. 8 PromO BayNAT)

Hiermit erkläre ich mein Einverständnis, dass bei Verdacht wissenschaftlichen Fehlverhaltens Ermittlungen durch universitätsinterne Organe der wissenschaftlichen Selbstkontrolle stattfinden können.

Bayreuth, den _____ _____

Moritz Lehmann