Report: Fluid Simulation in Computer Graphics WS2020

Weakly Compressible Smoothed Particle Hydrodynamics and Position Based Fluids

Berat Ertural, Dennis Ledwon
RWTH Aachen, Visual Computing Institute
Supervisor: Jose Antonio Fernandez Fernandez

> Reading maketh a full man; conference a ready man; and writing an exact man.
>
> *Sir Francis Bacon*

1 Introduction

Fluids play a vital role in our lifes. Understanding the motion and behaviour of fluids is not only crucial to many fields of engineering, but it also aids us in creating beatiful virtual worlds, such as used in games and cinema. The simulation of fluids is a vast and rapidly growing field. During the course of this practical, we studied various methods, worked on simulations of our own and analysed different aspects a fluid solver.

The motion of fluids is governed by the famous Navier-Stokes equation, as given below in its Lagrangian form;

$$\dot{v}_i = \frac{\partial v_i}{\partial t} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2 v_i + \frac{F_i^{ext}}{m_i}. \quad (1)$$

The Lagrangian form denotes the sampling points as dynamic particles moving with the flow of the fluid. The above PDE describes the acceleration $\dot{v}_i$ of such a particle i as depending on the internal pressure gradient ∇p_i and the veloctity divergence $\nabla^2 v_i$ of the fluid. The velocity divergence and friction coefficient ν represent the viscosity of the fluid due to internal friction. The pressure gradient drives the acceleration due to internal pressure differences, to achieve incompressibility. In theory the density of the fluid is said to be constant, $\frac{\delta\rho}{\delta t} = 0$. In practice, absolute incompressibility is not achievable due to numerical errors. Sometimes small deviations from the rest density of the fluid is desirable. This will be discussed in the next chapters. [**?**, **?**]

As with many PDE describing natural phenomena, the Navier-Stokes equation has no analytical solution. If it had, this report would end here with a reference to the proof of someone who is now one million dollars richer. Nonetheless, numerical approximations to fluid dynamics exist and during the course of the practical lecture we were tasked to implement two such solvers, namely Weakly Compressible Smoothed Particle Hydrodynamics (WCSPH) [**?**] and Position based Fluids (PBF) [**?**]. In the next section we will give a short overview of the used methods and implementation of the application. We will then present various experiments and discuss our observations and the results.

2 Methods

In the following a short overview of the methodological background and implementation will be given. In addition, certain aspects of the implementation that were done differently than from the given assignment will be presented. For a thorough discussion of the SPH and PBF framework we refer the reader to the review of SPH fluids by Ihmsen et al. [**?**] and to the original paper about PBF by Macklin et al. [**?**], as well as the works of Akinci et al. [**?**, **?**].

2.1 Weakly Compressible SPH

Smoothed Particle Hydrodynamics (SPH) is a class of particle based methods for fluid simulation with a long record of scientific publications. The core idea of SPH is to approximate any attribute A_i of a point at an arbitrary position $\vec{x}_i$

by weighting the attributes of the immediate surrounding particles $j \in N$;

$$A_i = \sum_{j \in N} \frac{m_j}{\rho_j} A_j \cdot W(||\vec{x}_i - \vec{x}_j||/h). \quad (2)$$

Particle mass m_j and density p_j is used as a normalization factor. The weighting is performed by the so called kernel function $W : \mathbb{R} \to \mathbb{R}$. It depends on the ratio of the distance between two particles and the smoothing length h, which is defined to be a multiple of the particle diameter; $h = \mu \cdot 2r$. We set $\mu = 1.1$. during the remainder of this report. The kernel is then given as $W(q) = \frac{1}{h^3} f(q)$. The function for the kernel used throughout the application is the following cubic spline [**?**],

$$f(q) = \frac{3}{2\pi} \begin{cases} \frac{2}{3} - q^2 + \frac{1}{2}q^3 & 0 \leq q < 1 \\ \frac{1}{6}(2-q)^3 & 1 \leq q < 2 \\ 0 & 2 \leq q \end{cases}. \quad (3)$$

It should be noted, that the effective radius of the kernel function, and therefore the number of neighbourhood particles taken into account, depends on the smoothing length. Weakly Compressible SPH first computes the particle density ρ_i using (2) and then applies this to a so called Equation of State (EOS) to derive the resulting particle pressures p_i. In the application the following EOS was used to derive the pressures;

$$p_i = max(0, B(\rho_i - \rho_0)), \quad (4)$$

where B is a stiffness constant. Equation (4) is similar to hookes law, as the pressure and therefore the resulting force depends linearly on the displacement. Therefore WCSPH allows for a small degree of incompressibility of the fluid. We will later discuss the implications of this. Using the formulations above we can derive expressions for the gradient and divergence field of an attribute and therefore compute the pressure and viscosity term of the Navier-Stokes equation.

2.2 Position Based Fluids

Position Based Fluids (PBF) is a relatively new method based on the authors previous work on Position Based Dynamics (PBD) [**?**]. Within this framework, a system is modelled using constraint functions, which describe the theoretical optimum or the equilibrium state. A solver will try to move the system towards the equilibrium by iteratively moving the positions of the sample points, such that the constraint violation is minimized. For fluidsimulations, PBF tries to achieve incompressibility by specifying the constraint (for a particle i);

$$C_i(\hat{x}) = \frac{\rho_i(\hat{x})}{\rho_0} - 1, \quad (5)$$

where $\hat{x}$ is the collection of all particles involved in the computation of ρ_i. Therefore the solver will try to find a displacement $\Delta\hat{x}$, such that $C_i(\hat{x} + \Delta\hat{x}) = 0$. A sensible choice is to move the particles a fraction λ along the constraint gradient ∇C proportional to inverse inertia M^{-1};

$$\Delta\hat{x} = M^{-1}\nabla C\lambda. \quad (6)$$

An approximate solution to the linear equation above is derived using a method similar to a Jacobi iteration. The velocities are then derived implicitly from the differences of the positions between frames. In the application PBF is used to replace the computation of the pressure term in the Navier-Stokes equation. To avoid attractive forces, which may lead to particle clumping and unwanted behaviour, the constraint from (5) is changed to an inequality constraint, $C_i > 0$.

2.3 Neighborhood Search

Most of the time the bottleneck of a particle based fluidsimulation is the neighborhood search. An efficient algorithm to find the particles in direct proximity is key to a stable runtime performance. The application uses the CompactNSearch algorithm by Bender et al. [**?**], which implements the parallel, hash-based neighborhood search by Ihmsen et al. [**?**]. To further enhance runtime performance, we perform a Z-index sort of particle positions after a fixed number of iterations. Sorting the particles depending on their spatial properties will decrease the number of cache misses and therefore speed up the access time significantly.

2.4 Time integration

The system is advanced in time using the semi-implicit euler method, which is also known as the symplectic euler, as it conserves the energy over time. Furthermore, XSPH smoothing is applied to the velocities. The effect is that neighboring velocities are aligned towards each other, which results in a more viscous behaviour and helps to keep the simulation more stable. The time step Δt is determined by the Courant-Friedrich-Levy (CFL) condition, $\Delta t \leq \lambda \frac{2r}{||v_{max}||}$, with particle radius r and maximum particle velocity v_{max}. The CFL condition sets the timestep dynamically, so that the particles will only move a fraction of their diameter during one iteration of the symplectic euler scheme. To further enhance the stability of the methods, we implemented a simple linear drag force which causes deaccalaration proportional to the velocity;

$$f_{drag,i} = -k_{drag} \cdot v_i. \tag{7}$$

The drag coefficient k_{drag} should ideally be below 0.1 as to not slow down the simulation too much. The implemented drag force is not ideal and in no way realistic. Modelling realistic drag for SPH methods with multiphase coupling and liquid-air interaction remains an active research field.

2.5 Boundary Interaction

To realise the interaction between the fluid and rigid boundaries, we use the boundary particle method by Akinci et al. [**?**]. By taking advantage of the particle based nature of the presented methods we are able to integrate contributions from boundary particles into the calculations of densities, forces and position corrections. Boundaries in a scene are usually represented by triangle mesh models. By sampling each surface triangle, we are able to sample the whole input mesh. To this end is reasonable to dissolve a higher resolution mesh into a low resolution mesh which preserves the topological shape, use the low res mesh for the simulation and keep the original high resolution mesh for rendering the final simulation. This will reduce the overall number of triangles and therefore the number of sampled particles. Figure 1 shows and example of the armadillo mesh used for most simulations in this report.

Figure 1: High resoltuion mesh for rendering (left) and low resolution mesh for simulation (right). The mesh on the left maintains the topological shape of the original.

To sample the particles on a triangle surface we perform a variation of the Pineda algorithm [**?**] in the local coordinate system of the triangle surface. The basis transform that will yield the local triangle coordiantes $p \in \mathbb{R}^2$ for a point in global space $x \in \mathbb{R}^3$ is given in (8).

$$p = \begin{pmatrix} u_x & u_v & u_z & u^T a \\ v_x & v_v & v_z & v^T a \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot x \tag{8}$$

The orthonormal basis vectors of the triangle $u, v \in \mathbb{R}^3$ are parallel to the plane. The basis vector perpendicular to the plane is not used. The origin of the local coordinate system is a triangle corner $a \in \mathbb{R}^3$. Applying the pineda algorithm is now straightforward. We first enclose the triangle by an axis-aligned bounding box (AABB) in the local coordiante system. We then go along the bounding box and check for each sample point if its inside the triangle by checking against its edges, as described by Pineda [**?**]. We use a hexagonal sampling pattern to ensure maximum coverage of the surface. Figure 2 demonstrates the hexagonal sampling in the bounds of the rectangle enclosing the triangle in its local coordinate system. Sample points inside the triangle are shaded opaque blue and sample points outside are transparent. The borders are slightly offset as to allow more samples to be inside the

triangle. This is to ensure that the triangle mesh of the boundary has no holes where the edges of two triangles meet. The oversampling is later compensated by recomputing the volume of each boundary particle depending on the number of particles in its close neighbourhood.

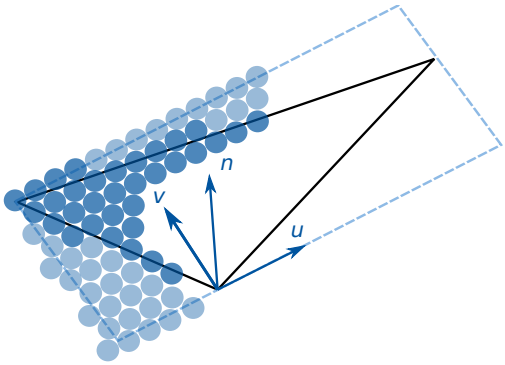

Figure 2: Hexagonal sampling using pineda algorithm in local triangle space.

We transform the sampled points back into three dimensional global space by multiplying them by the inverse of (8). Since the matrix is orthonormal its inverse is given by simply transposing the matrix.

2.6 Tension and Adhesion

Surface tension effects are implemented according the method by Akinci et al. [**?**]. Akinci et al. define two types of forces, namely fluid cohesion and boundary adhesion. Fluid cohesion is an attractive force between particles. Macroscopically, it acts as a force field around the surface of the fluid which keeps the particles together. In this way surface tensions is realised and effects such as waterdroplets become arise. The adhesion force causes fluid particles to stick to solid boundaries, which results in a wetting effect.

2.7 Surface Reconstruction

To extract an explicit representation of the fluid surface, a signed distance function (SDF) $\Phi(\vec{x})$ describing the isosurface is required. We use the SPH method to approximate the isosurface at any arbitrary point $\vec{x}$, depending on the normalized density ρ_j of the neighborhood region $j \in N$;

$$\Phi(\vec{x}) = \sum_{j \in N(\vec{x})} \frac{1}{\rho_j} W(||\vec{x} - x_j||/h) - c. \qquad (9)$$

The parameter c controls the zero level of the isosurface and therefore the enclosed volume of the explicit representation. We set $c = 0.6$ during the remainder of the report. Using this method, we compute $\Phi(\vec{x})$ at points in a grid and extract the triangle mesh representing the surface according to the Marching Cubes algorithm [**?**].

2.8 Implementation

The implementation uses a primarily imperative programming style with the exception of a few object oriented paradigms. Data encapsulation in the form of C++ classes is applied to ensure data consistency during computation. Basic inheritance is used to be able to reuse code for different parts of the program. This style of programming allows us to be flexible enough to incrementially implement the practical assignment tasks while sacrificing little to no performance due to design overload.

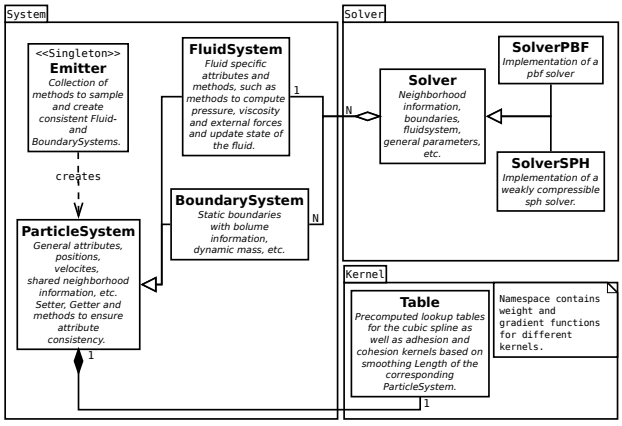

Figure 3: Schematic overview of the implementation.

The implementation is divided into two broad categories, systems and solvers. What we define as a system, includes fluid- and boundaryparticles, their generalization as particlesystems and auxilary constructs concerning particles. These represent the creation, state and behaviour of fluid- and boundary particles. On the other hand, we have two seperate specific solvers for

WCSPH and PBF. They aggregate fluid- and boundarysystems to model the progression of the given system. The specific solvers share a some functions and general parameters over their superclass. Since the function that starts a simulation run is defined over the superclass, it is possible to determine the type of solver at runtime. Figure 3 gives a schematic UML diagram of the relation between the different parts of the implementation. A particlesystem is a collection of particles which are represented by basic attributes, such as mass, density, position, velocity, etc. Since it is vital to keep all attributes of a particlesystem consistent during computation, a particlesystem object is only allowed to be created by the emitter object. The emitter object is a singleton, which means that there is always one and only one single instance of the object during the runtime of the program. A particlesystem is present in two forms, either as fluid particles or as boundary particles. These subtypes posess attributes and methods responsible for the behaviour of fluid or boundary particles in a system. For example, boundary particles possess a seperate variable volume attribute and a method to compute the volume for each particle depending on the density, e.g. the number of boundary particles in close neighbourhood. All particlesystems make use of precomputed lookup tables of the kernels. Since the kernel is evaluated a large number of times per particle, the lookup tables provide a significant speedup. In addition, each solver has been parallelized using OpenMP. To reduce the overhead from excessive creation and deletion of threads, a parallel region is created once upon entering an integration step. In this case special care has to be taken to avoid race conditions due to shared memory between threads.

3 Results

In this section, we run simple simulations to illustrate the differences between the WCSPH and the PBF solver. In Sec. 3.1, we provide a general comparison of the effects of fluid viscosity, boundary viscosity, surface tension and adhesion on both solvers by looking at dam break scenarios with different parameters. In Sec. 3.2 and Sec. 3.3 we take a closer look at minimal scenarios that showcase surface tension and adhesion. In Sec. 3.4, Sec. 3.5 and Sec. 3.6 we return to dam break simulations to show the impact of XSPH smoothing and stiffness on the WCSPH solver and the impact of XSPH smoothing and the number of iterations on the PBF solver.

3.1 Comparison: General

In order to compare the characteristics of the WCSPH and the PBF solver and to showcase the impact of the parameters `fluid_visc`, `boundary_visc`, `tension` and `adhesion`, we simulated simple dam break scenarios with 74589 fluid particles using the WCSPH and the PBF solver. Fig. 4 shows frame 36 of these simulations. In the simulations shown in panel 1 and panel 2, `tension` and `adhesion` were turned off and `fluid_visc` and `boundary_visc` were set to 0.001 to prevent their impact from covering up differences between the two solvers. When the fluid system is simulated using the WCSPH solver, it can be seen that there are some fluid particles have higher velocities than their counterparts when the PBF solver was used. Consequently, fluid particles move up further along the y-axis - opposite to the direction of gravitational acceleration - when the WCSPH solver is used. The maximum particle densities during every frame of the simulations that are shown in panel 1 and panel 2 are plotted in Fig. 5. Over the first sixty frames of the simulations, the maximum particle density per frame is significantly higher when the WCSPH solver was used. The biggest deviation between the respective maximum densities was observed in frame 16, with maximum densities of approx. 1100 kg/m^3 using the WCSPH solver and 1010 kg/m^3 using the PBF solver.

In panels 3 and 4 of Fig. 4, `fluid_visc` was increased from 0.001 to 0.025. Increasing `fluid_visc` hides the differences between the WCSPH and PBF solver that were observed in panels 1 and 2. Further, it can be seen that the height of the wave is smaller in panel 3 and 4 in comparison to panels 1 and 2 respectively, espe-

cially so for the simulations using the WCSPH solver (panel 1, panel 3).

In panels 5 and 6 of Fig. 4, `fluid_visc` was decreased back to 0.001 and `boundary_visc` was increased to 0.025. As `fluid_visc` is decreased, the differences between the WCSPH and PBF solver that were observed in panel 1 and panel 3 manifest themselves again. For both the WCSPH and PBF solver, there are more particles in proximity of the left boundary of the container when compared to panel 1 and panel 3. Additionally, the wave is narrower in panels 5 and 6.

In panels 7 and 8 of Fig. 4, `fluid_visc` and `boundary_visc` are set to 0.001 again and `tension` and `adhesion` are set to 0.25 and 10 respectively. Compared to panel 1 and panel 3 respectively, a fluid tongue forms as the wave crashes. As indicated by the darker color of the fluid particles, the densities of fluid particles at the top of the wave are higher for the WCSPH and PBF solver when `tension` and `adhesion` are turned on. Especially for the simulation using WCSPH, there are much less isolated fluid particles when `tension` is turned on. Comparing panels 5 and 6 with panels 7 and 8 respectively reveals the differences between boundary viscosity and adhesion: Particles in proximity of the boundary have lower density when boundary viscosity is used than when adhesion is used. To further illustrate the difference between `boundary_visc` and `adhesion`, frame 160 of the same simulations that are shown in panels 5 to 8 are shown with a different camera angle in panels 1 to 4 of Fig. 6, respectively. It can be seen here that fluid particles that are in the proximity of the boundary sink back into the fluid again when using boundary viscosity with both the WCSPH and PBF solver. However, particles stay stuck to the container when adhesion is used for both solvers.

3.2 Comparison: Tension

We simulated fluid cubes of 54872 particles in the absence of gravity to investigate the impact of surface tension on simulations using the WCSPH and PBF solver. Frame 53 of these simulations with `tension` set to 0, 0.25 and 1 is shown in Fig. 7. For both the WCSPH and the PBF solver, the fluid cube stays in its initial configuration if `tension` is turned off. When `tension` is turned on, the fluid contracts into a spherical shape for both solvers. The higher the `tension` value, the faster this transition occurs. Generally, it seems like the spherical configuration is reached slightly faster when the PBF solver is used. It is important to note that when `tension` is turned on, the fluid reaches a spherical configuration in all of the simulations presented in Fig. 7 eventually. An earlier frame was chosen in order to show how quickly the fluid cube contracts with different `tension` values. Lastly, the densities of the fluid particles are more uniform as the spherical configuration is assumed when the PBF solver is used (see panels 2, 3, 5, 6).

3.3 Comparison: Adhesion

In order to clearly demonstrate the effect of `adhesion` on the fluid simulation, we simulated a block of 67868 fluid particles falling onto an adhesive icosahedron using the WCSPH solver and the PBF solver. Fig. 11 shows frame 160 of these simulations. For simulations using WCSPH, `adhesion` was set to 1, 2 and 5 in panels 1, 2 and 3 respectively. As the `adhesion` value is increased, the amount of fluid that is stuck to the icosahedron increases. In panel 3, the `adhesion` value is high enough to cause fluid particles that directly touch the icosahedron to almost resist gravity entirely. When `adhesion` is set to 2 (panel 2), access fluid drips down the icosahedron in a stream that is narrower than the diameter of the icosahedron. This effect is still present when `adhesion` is set to 1 (panel 1), even though it is much less pronounced. For simulations using PBF, adhesion was set to 1, 3 and 10 in panels 4, 5 and 6, respectively. Again, more fluid particles are attached to the icosahedron as `adhesion` increases. The streams of liquid that drip off the icosahedron are much smaller than the icosahedron diameter when `adhesion` is set to 3 and 10. However, the stream is narrower when the `adhesion` is set to 3 instead of 10. It is important to note that fluid still drops off when

PBF is used with high `adhesion` values. This was not the cases for simulations using WCSPH.

3.4 Comparison: Smoothing

Multiple dam break scenarios 74589 were simulated using the WCSPH and PBF solver to investigate the impact of XSPH smoothing. Fig. 9 shows frame 31 of these simulations. It is notable, that the differences that were observed between the WCSPH and the PBF solver in panels 1 and 2 of Fig. 4 are less pronounced when XSPH smoothing is deactivated: Even in the simulation that used the PBF solver (panel 2), there are single particles with higher velocities than others. Increasing `xsph_smoothing` to 1 leads to different outcomes depending on the used solver (panel 3, panel 4). Increasing XSPH smoothing only affects the behavior of the simulation to a small extent when the WCSPH solver is used. However, we observed fewer particles on the left side and more particles on the right side of the container when XSPH smoothing was activated. When the PBF solver is used, increasing XSPH smoothing causes the fluid particles to lose energy and have a more viscous appearance. There are barely any fluid particles that split off from the rest of the fluid body, when `xsph_smoothing` is set to 1 and the PBF solver is used.

3.5 SPH Parameter: Stiffness

When the WCSPH solver is used, the `stiffness` parameter can be tuned to impact the behavior of the simulation. Dam break simulations with 74589 fluid particles were run with `stiffness` set to 2500, 10000 and 25000. Frame 36 of the resulting simulations is shown in panels 1, 2 and 3 of Fig. 10, respectively. When `stiffness` is set to 2500 and 10000 the simulations are stable and look very similar to each other. However, increasing the stiffness to 25000 causes the simulation to become unstable.

3.6 PBF Parameter: Iterations

When the PBF solver is used, the `nr_iterations` that the PBF solver uses to minimize the density constraint can be set as a parameter. Dam break simulation with 74589 fluid particles were run with `nr_iterations` set to 1, 3, 5, 7, 9 and 11. When `nr_iterations` was set to 1 and 3, the simulations were stopped after five hours on frame 4 and 12, respectively. For both of these simulations, the timestep that was dynamically calculated using the CFL condition became very small and prevented the simulation from progressing at a reasonble pace. When `nr_iterations` was set to 5, 7, 9 and 11, the resulting simulations were very similar and hard to distinguish from each other. `max_timestep` was set to 0.002s in all of the simulations in Fig. 8.

4 Discussion

4.1 Comparison: General

A solution to the Navier-Stokes equation can be numerically approximated using WCSPH (Sec. 2.1) and PBF (Sec. 2.2). In order to showcase the differences between these two solvers, their general characteristics were investigated in Sec. 3.1. As described in Sec. 3.1, it is more common to find single particles with a lot of energy when WCSPH is used. Hand in hand, we observed that the maximum particle density tends to be higher when the WCSPH solver is used (Fig. 5). As big positive deviations of the a particle's density from rest density cause big pressure terms in WCSPH (Eq. 4), we suggest that the increased particle velocity in WCSPH results from violations of the fluid incompressibility during the simulation. This is in accordance with [**?**], where the maximum fluid density is used as a measure for fluid compressibility. On the other hand, PBF approximates a solution to the density constraint (Eq. 5) during every simulation step. This explains why the maximum densities in Fig. 5 are smaller when the PBF solver is used. As a result, fluid incompressibility is not violated as much as when the WCSPH solver is used. Consequently, single particles with increased energy are rare with the PBF solver.

Increasing the fluid viscosity reduces the energy in the system and increases the stability of the simulation (Fig. 4, panels 3, 4). This

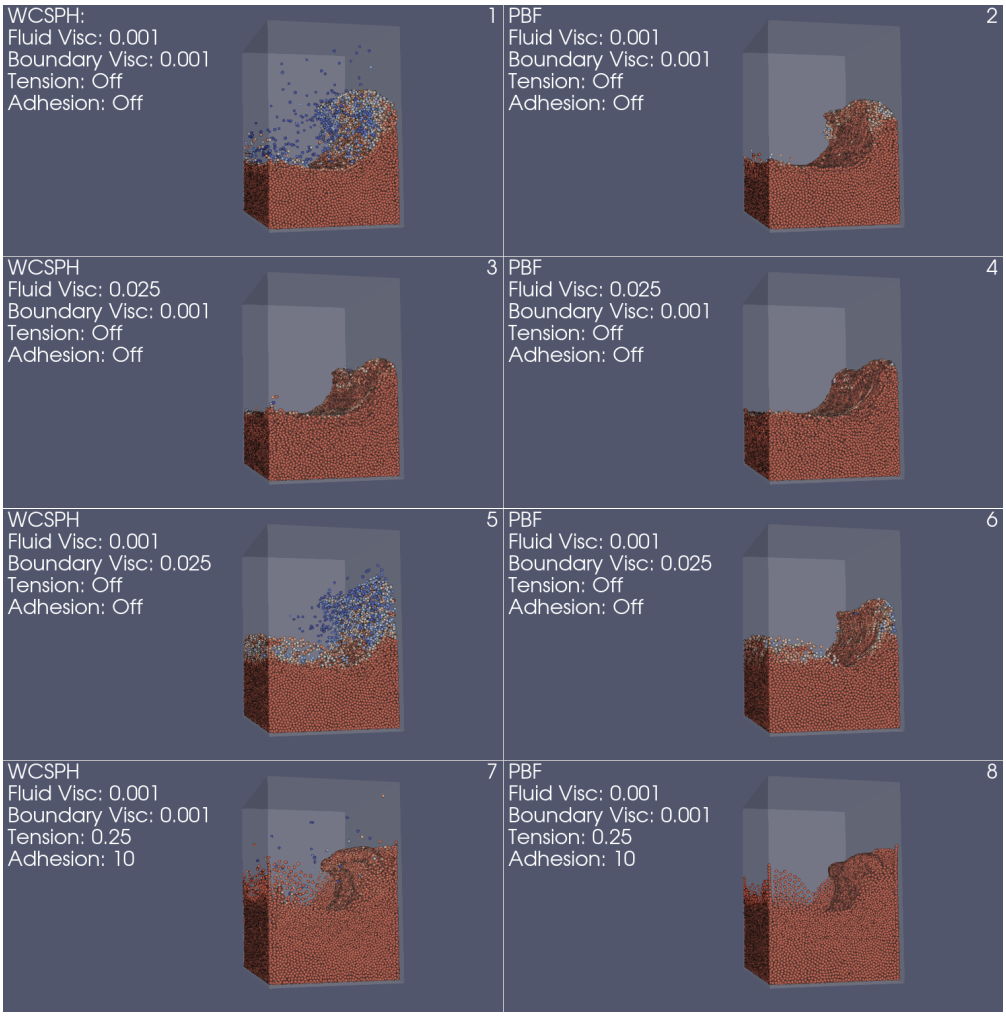

Figure 4: Frame 36 of simulations of 74589 fluid particles in a dam break scenario using the WCSPH and the PBF solver. In all simulations, a linear drag force of 0.02 was applied. For simulations using WCSPH, `max_timestep = 0.0005s, xsph_smoothing = 0.5, stiffness = 2500`. For simulations using PBF, `max_timestep = 0.002s, xsph_smoothing = 0.1, nr_iterations = 7`. The other parameters were set as noted in the panels.

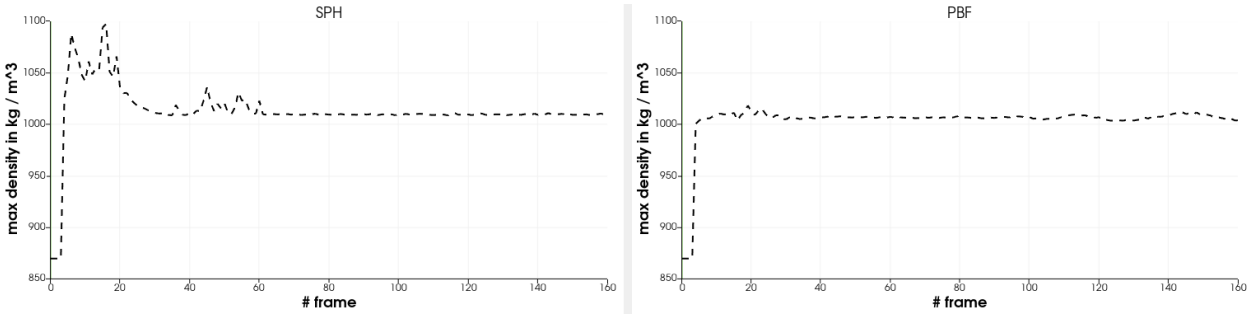

Figure 5: Max. density in kg/m^3 during every frame of the simulations in panel 1 and panel 2 of Fig. 4.

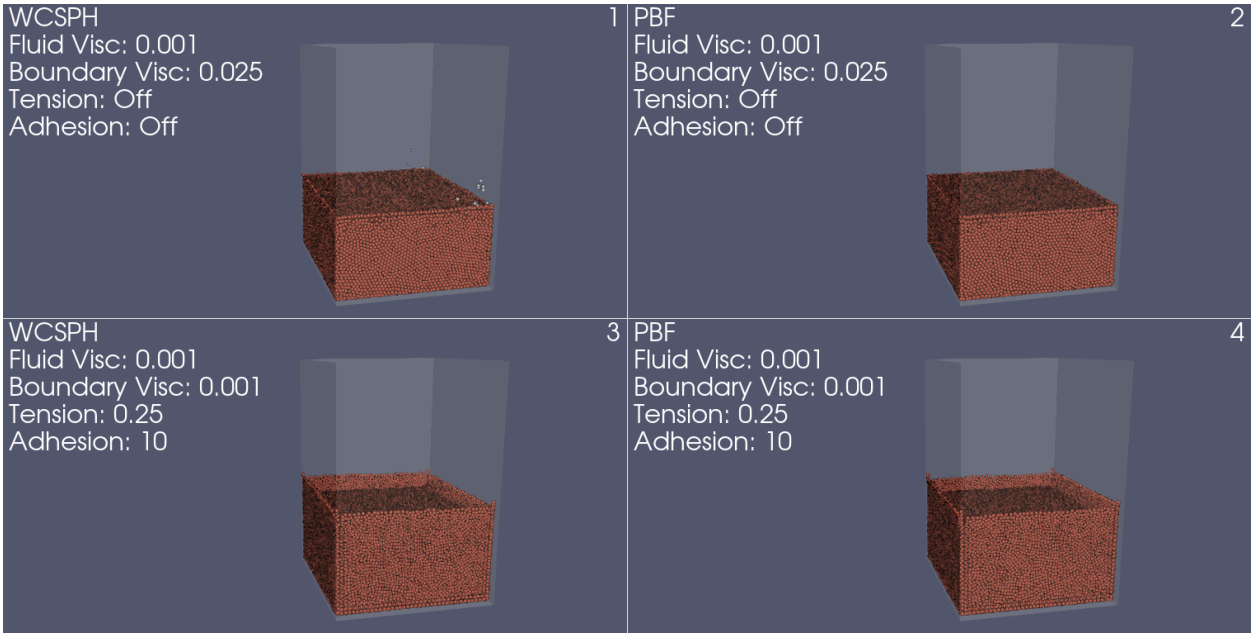

Figure 6: Frame 160 of simulations of 74589 fluid particles in a dam break scenario using the WCSPH and the PBF solver. In all simulations, a linear drag force of 0.02 was applied. For simulations using WCSPH, `max_timestep = 0.0005s, xsph_smoothing = 0.5, stiffness = 2500`. For simulations using PBF, `max_timestep = 0.002s, xsph_smoothing = 0.1, nr_iterations = 7`. The other parameters were set as noted in the panels.

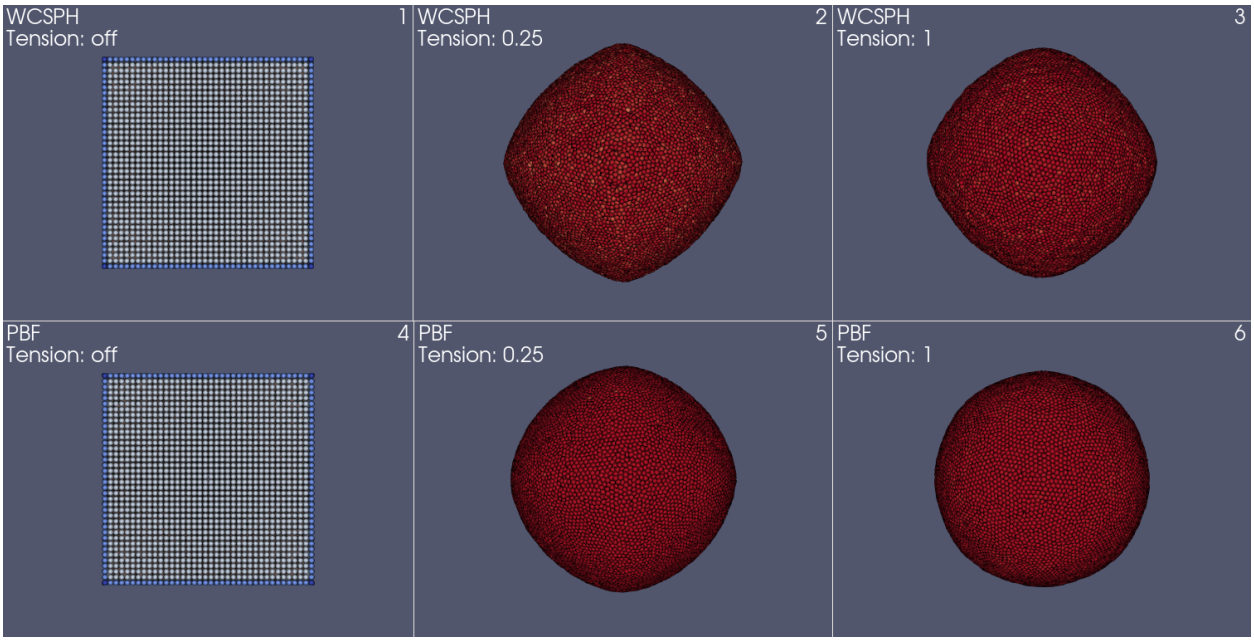

Figure 7: Frame 160 of simulations of 54872 particles arranged in a block without gravity using the WCSPH and the PBF solver. In all simulations, smoothing and gravity was disabled. For simulations using WCSPH, `max_timestep = 0.0005s, stiffness = 2500, fluid_visc = 0.001` . For simulations using PBF, `max_timestep = 0.002s, nr_iterations = 7, fluid_visc = 0.001`. `tension` is set as indicated in the panels.

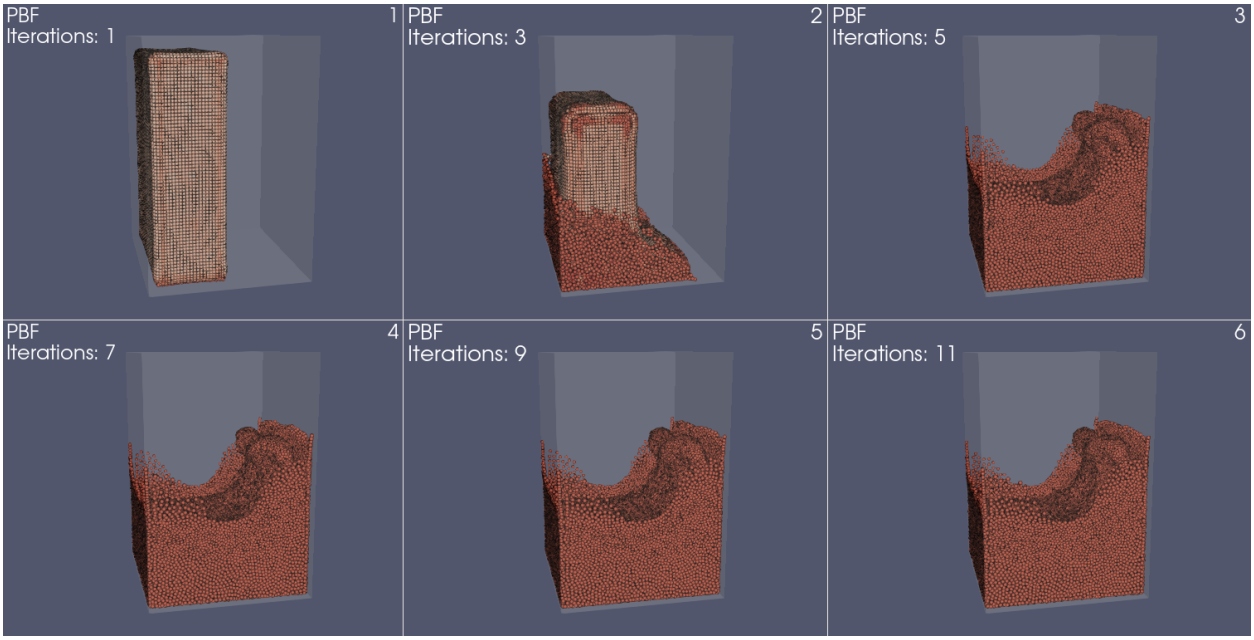

Figure 8: Frame 36 of simulations of 74589 fluid particles in a dam break scenario using the PBF solver. In all simulations, a linear drag force of 0.02 was applied and `max_timestep = 0.002s`, `xsph_smoothing = 0.1`, `fluid_visc = 0.001`, `boundary_visc = 0.001`, `tension = 0`, `adhesion = 0`. `nr_iterations` was set as noted in the panels. The simulations in panel 1 and panel 2 were interrupted after five hours of simulation on frame 4 and 12 respectively.

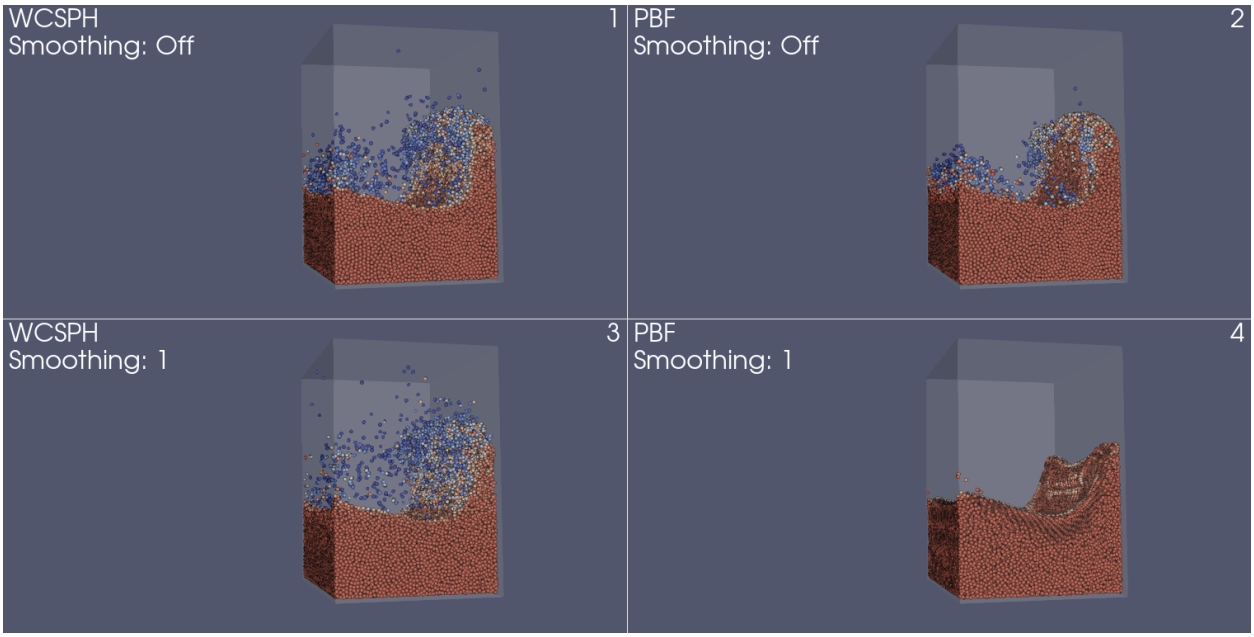

Figure 9: Frame 31 of simulations of 74589 fluid particles in a dam break scenario using the WCSPH and the PBF solver. In all simulations, a linear drag force of 0.02 was applied and `fluid_visc = 0.001`, `boundary_visc = 0.001`, `tension = 0`, `adhesion = 0`. For simulations using WCSPH, `max_timestep = 0.0005s`, `stiffness = 2500`. For simulations using PBF, `max_timestep = 0.002s`, `nr_iterations = 7`. `xsph_smoothing` was set as noted in the panels.

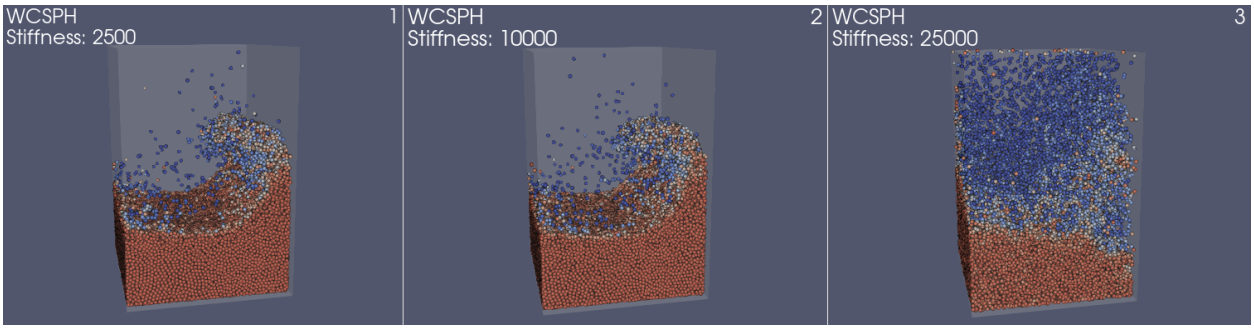

Figure 10: Frame 36 of simulations of 74589 fluid particles in a dam break scenario using the WCSPH solver. In all simulations, a linear drag force of 0.02 was applied and `max_timestep = 0.0005s`, `xsph_smoothing = 0.5`, `fluid_visc = 0.001`, `boundary_visc = 0.001`, `tension = 0`, `adhesion = 0`. `stiffness` was set as noted in the panels.

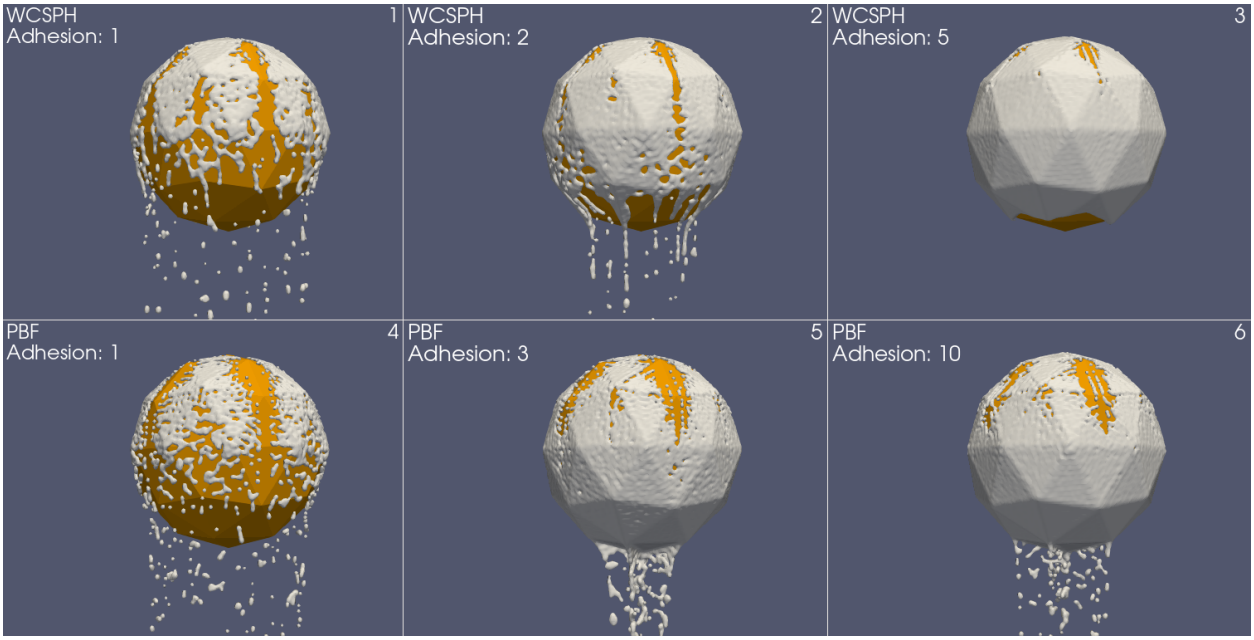

Figure 11: Frame 160 of simulations of 67868 fluid particles falling onto an icosahedron using the WCSPH and the PBF solver with surface reconstruction. In all simulations, a linear drag force of 0.025 was applied. For simulations using WCSPH, `max_timestep = 0.0005s`, `xsph_smoothing = 0.25`, `stiffness = 2000`, `fluid_visc = 0.01`, `boundary_visc = 0.01`, `tension = 0.15`. For all simulations using the PBF solver `max_timestep = 0.002s`, `xsph_smoothing = 0.1`, `nr_iterations = 7`, `fluid_visc = 0.01`, `boundary_visc = 0.005`, `tension = 0.1`. `adhesion` is set as noted in the panels.

was to be expected, as fluid viscosity is added to model internal friction of the fluid. Especially for WCSPH, the results of the simulation change notably when fluid viscosity is increased. Due to the interal friction, fluid particles move slower, consequently making it harder for fluid particles to get too close to each other during one simulation step. Thus, it is less likely that the fluid incompressibility is violated when fluid viscosity is used in the simulation, which would blow up pressure terms unreasonably (see. Eq. 4).

While investigating the difference between the effects of boundary viscosity and adhesion (Fig. 4, panels 5, 6, 7, 8), we noticed that fluid particles that are close to the boundary have higher densities when adhesion is used in comparison to boundary viscosity. One likely explanation for this phenomenon is that adhesion models an attractive force between boundary and fluid particles [**?**], whereas boundary viscosity merely models friction between boundary and fluid particles [**?**]. Thus, the distance between boundary and fluid particles when adhesion is used is smaller than their distance when boundary viscosity is used, leading to a higher particle density. This would be in line with the observations from Fig. 6, where fluid particles stick to boundary particles when adhesion is used, but not when boundary viscosity is used.

4.2 Comparison: Tension

Surface tension is caused by cohesive forces among neighboring fluid particles [**?**]. Fluid particles on the surface of the fluid do not have neighbors on all sides and are thus pulled inwards. This matches the observations that were made while simulating a fluid cube with surface tension in the absence of gravity (Fig. 7). As expected, fluid particles were not pulled inwards when surface tension was turned off, causing the cube to stay in its initial configuration. The higher the `tension` value, the faster the cube assumed a spherical configuration. The fact that the spherical configuration was assumed slower and that the particle densities were less uniform when using the WCSPH solver could have the same underlying cause: As discussed in Sec. 4.1, the WCSPH is prone to violating fluid incompressibility. As surface tension draws particles closer to each other, their densities become too high and they might push particles slightly outwards again. However, this hypothesis would have to be tested more thoroughly, possibly by taking a closer look at particle densities throughout the simulations using WCSPH.

4.3 Comparison: Adhesion

As briefly discussed in Sec. 4.1, adhesion models an attractive force between boundary and fluid particles [**?**]. In Fig. 11 we observed that the higher the `adhesion` value, the more fluid is attached to the icosahedron for both WCSPH and PBF, which matches our expectations. Interestingly, when `adhesion` is set to 5 with the WCSPH solver, the fluid particles that touched the icosahedron became almost entirely stuck to it, even resisting gravitational forces. This suggests that adhesion forces are stronger when the WCSPH solver is used in comparison to the PBF solver. It is important to note that the opposite observation was made in Sec. 4.2 while discussing surface tension. One possible hypothesis could be that fluid particles are dragged down the icosahedron by neighboring fluid particles due to stronger tension forces, which overcome even higher adhesion values when the PBF solver is used.

Further, it is remarkable that the diameter of the fluid stream that drips down the icosahedron does not decrease as `adhesion` is increased from 5 to 10 with the PBF solver. One possible explanation could be that an `adhesion` of 10 causes one layer of fluid particles to be stuck to the surface of the icosahedron entirely. The fluid particles in the stream that drips down the icosahedron would thus be blocked off from the boundary, leading to a smaller adhesive force. However, this would raise the question why there are areas on the surface of the icosahedron that were clearly touched by fluid before, but do not have any attached to it in the current frame. Such areas could be found e.g. at the top of the icosahedron. Finally, if this hypothesis were true, we would expect a similar behavior for the simula-

tion using WCSPH and an `adhesion` of 5.

4.4 Comparison: Smoothing

According to [**?**], XSPH smoothing can be used to ensure that particles that are close to each other will have nearly identical velocities. This helps preventing regions where SPH particles pass through each other without increasing fluid viscosity. In our experiments in Fig. 9, we observed that particles velocities indeed seem to be closer to their neighbors' velocities when XSPH smoothing is used. However, this effect was much more pronounced for the PBF solver. Most likely, this can be attributed to the fact that smoothing is applied without taking pressure forces into account in PBF. Consequently, most particle velocities that are used for XSPH smoothing in the PBF solver have smaller magnitudes than their counterparts in the WCSPH solver. As a result, the absolute difference between velocities of neighboring particles after XSPH smoothing is much smaller for the PBF solver. If the density constraint was already minimized in the previous timestep, the Δx that are calculated during the constraint minimization of the current timestep will be small. Overall, this leads to a loss in energy when PBF and XSPH smoothing are used together. In [**?**], XSPH smoothing is used to artificially introduce viscosity in fluid simulations using an SPH solver. We were not able to observe more viscous fluid behavior when XSPH smoothing was used with the WCSPH solver. Maybe it is worth investigating the effects of smoothing on the WCSPH solver in simulations with a smaller stiffness value again.

4.5 SPH Parameter: Stiffness

In the WCSPH solver, the particle pressures are computed by multiplying positive deviations in particle density from rest density with a `stiffness` constant (Eq. 4). Thus, the higher the `stiffness` constant, the higher the resulting particle pressures, which in turn leads to higher particle velocities. In Sec. 3.5 we observed that the difference between simulations with `stiffness` set to 2500 and 10000 were minor. However, if the `stiffness` is set to 25000 the simulation becomes instable. [**?**] suggest that higher `stiffness` values require lower timesteps. It is possible that the simulation with `stiffness` set to 25000 would be stable, if the timestep was decreased accordingly. On the other hand, if the timestep was increased, it is plausible that the simulation with `stiffness` set to 10000 would become instable as well.

4.6 PBF Parameter: Iterations

During every simulation step with the PBF solver, the density constraints (Eq. 5) are minimized iteratively [**?**]. Increasing the number of iterations trades runtime performance of the simulation for lower compressibility of the fluid. In Sec. 3.6, we observed that simulations are almost identical regardless of the exact number of iterations, as long as the number of iterations is sufficiently high. In Fig. 8, this was the case for all simulations that used at least five iterations per simulation step. Simulations that used an insufficient number of iterations stopped progressing at a reasonable pace after few frames, as the timestep that was dynamically calculated using the CFL condition approached 0s. This is due to excessive particle velocities. Particle velocities are expected to explode if an insufficient number of iterations is used in the PBF solver, as a particle that has high density after a position update without pressure forces will have a significant position changes in the first couple of iterations. As the particle velocity directly depends on these position changes, the new particle velocity will be big as well.

5 Conclusion

6 Outlook

The WCSPH and PBF implementation as well as the surface reconstruction offers a high level of concurency in the form of data parallelism. This means, that there are many parts in the programm where sequentially processed data can be split into independant groups which can be processed in parallel.

References