Volume 0 (2023), Number 0, pp. 1–23

Monolithic Friction and Contact Handling for Rigid Bodies and Fluids using SPH

T. Probst and M. Teschner

University of Freiburg, Freiburg, Germany
probstt@cs.uni-freiburg.de

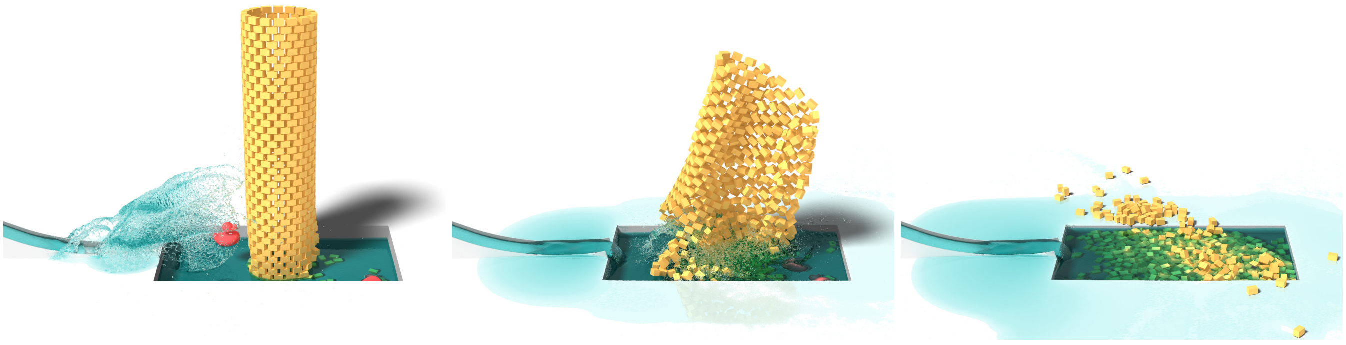


Figure 1: A pure SPH simulation showcasing the capabilities of the proposed particle-based strong coupling between pressure forces applied in the fluid and at rigid bodies as well as dry friction forces between rigid bodies. Particles are used to represent the simulated rigid bodies, the shown fluid and boundary geometry. The tower, consisting of over 600 blocks, is brought to fall by two rigid ducks. When collapsing into a basin filled with fluid, our solver is able to robustly handle fluid incompressibility, fluid-rigid interaction, rigid-rigid collision forces and dry friction forces.

Abstract

We propose a novel monolithic pure SPH formulation to simulate fluids strongly coupled with rigid bodies. This includes fluid incompressibility, fluid-rigid interface handling and rigid-rigid contact handling with a viable implicit particle-based dry friction formulation. The resulting global system is solved using a new accelerated solver implementation that outperforms existing fluid and coupled rigid-fluid simulation approaches. We compare results of our simulation method to analytical solutions, show performance evaluations of our solver and present a variety of new and challenging simulation scenarios.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Physical simulation;*** *Collision detection;*

1. Introduction

Rigid body dynamics, as well as fluid simulations are subject of an extensive amount of research motivated by great interest in applications in computer graphics, robotics and industrial prototyping [BET14, IOS*14, Bri15]. Even though most of the work focuses on exclusively simulating either the dynamics of rigid bodies or the motion of fluids, there exist more recent approaches combining fluids and rigid bodies into one simulation framework [AIA*12, MMCK14, KB17, HFG*18, GPB*19, BKWK19]. It is easy to see that these combined methods vastly expand the range of possible applications and simulation scenarios. Usually, in combined approaches, rigid contacts are resolved using a body representation and simulation method distinct from the fluid representation, fluid-rigid interface handling and internal fluid pressure force computation [AIA*12, MMCK14, KB17, HFG*18]. Even in approaches that resolve rigid body contacts based on a particle representation, a concept well known from e.g. [CS79, BYM05, TSIHK06, Ngu07, MMCK14, Coe17], the rigid-rigid contact handling typically still uses different methods than the fluid-rigid interface handling and the fluid solver. For instance, He et al. [HBH*18] and Peng et al. [PZWZ21] use *smoothed particle hydrodynamics* (SPH) to simulate fluids, yet employ a *discrete element method* to handle rigid contacts and fluid-rigid interaction. Similarly, Macklin et al. [MMCK14] implement a position-based SPH fluid, but on the other hand a collision detection method similar to the *molecular dynamics* method [BYM05] together with shape matching constraints is used for rigid bodies. Due to impulsive collision responses, rigid contact handling strongly influences fluid-rigid constraints and consequently the internal fluid state. Equivalently, internal fluid forces might also have a great effect on neighboring rigid bodies that needs to be considered when solving rigid contacts. By solving fluid and rigid body contact constraints sequentially, the mutual dependencies between the constraints are neglected, leading to unstable and inefficient coupled simulations. However, due to heterogeneous solving procedures for rigid bodies and fluids, mixed simulation methods have difficulties solving fluid and rigid body constraints in a combined manner. To overcome these issues, Gissler et al. [GPB*19] propose to simulate fluids, rigid bodies and their interactions exclusively with SPH and as such are able to build an intuitive and stable strong coupling between particle fluids and rigid bodies. However, even though realistic dry friction forces are crucial in order to authentically replicate rigid body behavior, the dry friction force proposed by Gissler et al. [GPB*19] lacks basic features such as the ability to reproduce stiction and conservation of momentum. Additionally, despite the fact that it is common practice to compute contact and dry friction forces at once in order to maintain simulation stability [KSJP08, BET14], the explicit friction proposed by Gissler et al. [GPB*19] is applied separately from the contact handling procedure.

1.1. Contributions

Motivated by promising results of existing coupled simulation methods, our overall goal in this paper is to develop a monolithic simulation method that implements a strong coupling between fluids and rigid bodies which is as far-reaching as possible. For this, we build one global system that encodes all constraints for fluid volumes, fluid-rigid interface handling and rigid-rigid contacts. In contrast to existing coupled approaches such as the interleaved Jacobi solvers proposed by Gissler et al. [GPB*19], our unified system not only allows a more precise analysis of system properties (e.g. diagonal elements), it also enables us to use a nonlinear conjugate gradient method, promising increased solver convergence speed. Since the capabilities of our rigid contact handling should match those of a designated rigid body simulator, we also develop a new implicit particle-based dry friction formulation for rigid-rigid contacts. This friction formulation addresses many shortcomings of the explicit friction force used by Gissler et al. [GPB*19], including the preservation of momentum and the ability to reproduce stiction. As the major improvement to Gissler et al. [GPB*19], our friction formulation is directly embedded in the global system and as such friction forces are included in the strong coupling between fluid pressure forces and rigid contact forces. To our best knowledge, there exist no other simulation method that implements such a far-reaching strong coupling between fluids and rigid bodies.

In summary, in this paper we present

- a novel particle-based implicit dry friction formulation based on the exact Coulomb friction model
- a new monolithic, implicit SPH solver that unifies fluid pressure, fluid-rigid interface forces and rigid-rigid contact handling including dry friction forces into one system that is solved using a nonlinear conjugate gradient method
- results that validate our friction formulation and additionally demonstrate the capabilities and versatility of the proposed simulation method, as already hinted at in Figure 1.

2. Related Work

Computational methods to simulate rigid body contact dynamics have been studied extensively since quite a long time [MW88, Hah88, Bar89, Bar90, Bar91, Bar93a, Bar93b, Mir96, Bro99, Ste00]. We refer to the report written by Bender et al. [BET14] for a great summary of earlier work and a general introduction to rigid body simulation. Previous work has shown that computing frictional forces is challenging due to their non-linear and non-smooth relation to relative velocities and normal forces [BET14, YSC*18, LDN*18, PAK*19, LFS*20, LJBBD20, LDW*22]. Often, instead of directly using the Coulomb friction model, it is modified such that the resulting optimization problem is better behaved. Linearizing the friction cone allows formulating Coulomb friction as a *linear complementarity problem* (LCP) [ST96, AP97, KSJP08], however, solutions might not satisfy the principle of maximum dissipation as frictional forces do not exactly oppose relative sliding velocities and the system size grows with the number of facets of the linearized cone [MEM*19, AE21]. Nevertheless, LCPs became a popular model to formulate rigid contact problems [Bar89, Bar93a, Bar94, Bar95, Ste00, GZO10, AO11, BET14, AE21] and a number of methods to solve LCPs have been developed [Bar94, KSJP08, CA09, Ebe10, CM11, Erl13]. The *box friction model* [OTSG09, TBV12, GNKT16, PAK*19, ANEK21] can be used to reduce the number of variables by ignoring the coupling between friction directions. As a consequence, friction forces might violate the Coulomb constraint and there are no guarantees concerning the principle of maximum dissipation [Erl17]. Alternatively, smoothing the velocity-force relation eliminates the discontinuous jump of the friction force and the resulting computational burden at stick-slip transitions [PRM19, GHZ*20, LFS*20, FLS*21, MEM*20, CLL*22, LKL*22]. Even though in most approaches the amount of smoothing is parametrized and can be reduced to get a better approximation of the true velocity-force relation at the cost of less computational stability, the friction force at zero velocity is always null and thus, these models cannot exactly reproduce stiction [PRSV16, PRM19, LKL*22]. Some approaches model static friction as linear springs which pull contacts back together over multiple simulation steps [YN06, XZB14]. In contrast to the methods described above, our friction computation is based on the exact Coulomb friction model including stiction, and we are able to directly embed the principle of maximum dissipation into our implementation without any further error-introducing simplifications. Newton-based approaches have shown to be able to implement the exact Coulomb friction model as well, by formulating the complementarity problem as non-smooth functions whose roots are found using a generalized version of Newton's method [BDCDA11, DBDB11, KTS*14, MEM*19]. Newton's method is quadratically convergent, which is especially useful when solving poorly conditioned problems, but in each iteration a linear system needs to be solved [MEM*19, AE21]. The solver used in our implementation can also achieve superlinear convergence with only slightly increased computational costs per solver iteration compared to a standard Jacobi iteration [SHNE10]. *Proximal operators*, as already employed in [JM92, JAJ98], can be used as a tool to implement a friction force computation following the exact Coulomb friction law [Erl17]. Additionally, implementations based on proximal operators have shown to be easily extendable to employ other friction models such as anisotropic friction [EMAK19] and models considering frictional torque [LG03, Erl17]. Our proposed simulation approach utilizes proximal operators as the basis for a more advanced conjugate gradient method.

Even though implicit formulations have gotten more attention in the last few years, explicit pressure solvers are still widely used in the context of SPH fluids [BT07, AIA*12, IOS*14, RHEW17]. For less complex scenarios (e.g. low fluid depth, softer requirements regarding incompressibility), explicit formulations benefit from lower computational costs per simulation step while the increased simulation stability through implicit formulations cannot outweigh their higher computational burden [KBST19, KBST22]. Similarly, explicit penalty methods for rigid body simulations are easy to implement and require little computational cost per step [MW88, BET14]. For our simulation method however, we still chose to employ an implicit formulation. This not only allows us to handle rigid contact constraints similar to fluid incompressibility, we are also able to simultaneously solve for pressure in the fluid, pressure forces at the fluid-rigid interface and between rigid bodies as well as friction. Computing friction and contact forces at once is known to be beneficial for simulation robustness and performance, since frictional forces can have a significant effect on pressure constraints at contacts and vice versa [KSJP08, BET14]. In Section 5.1 we demonstrate that explicit friction formulations, such as the one proposed by Gissler et al. [GPB*19], can have trouble reproducing stiction. We are not aware of an existing explicit simulation method that is able to stably simulate the complex scenarios presented in Section 5, including large mass ratios, fragile structures and high stacks of rigid bodies.

Two-way coupled simulation methods that combine rigid body contact handling with fluid dynamics have been heavily investigated in many contexts including Lattice-Boltzmann fluids [TIR06], height-field fluids [TMFSG07, CM10, SBC*11], Lagrangian vortex methods [VHLL14], MPM fluids [YLCH18, HFG*18], Eulerian fluids [CMT04, GSLF05, BBB07, EWC*10] and Lagrangian fluids [MST*04, BTT09, AIA*12, MMCK14, TL16, KB17, BKWK19, GPB*19]. For a detailed discussion on the collision handling used by the mentioned simulation methods we refer to Gissler et al. [GPB*19] and instead focus on how frictional forces between rigid bodies are considered in coupled simulation approaches. Akinci et al. [AIA*12] sample rigid bodies with particles and handle rigid-fluid contacts by first computing pressure using WCSPH [BT07] or PCISPH [SP09] and deriving pressure forces that are then applied to fluid and rigid particles. Friction is only mentioned to be computed at the fluid-rigid interface employing a laminar artificial viscosity force. Rigid-rigid contacts, including dry frictional forces, are handled separately using an external simulation software such as Bullet [Cou22]. In contrast, our simulation method unifies fluid pressure forces with rigid contact handling and friction, resulting in more stable fluid-rigid interfaces as already demonstrated in [GPB*19]. Macklin et al. [MMCK14] introduce a position-based particle simulation framework that combines many object types, including fluids and rigid bodies. Particles representing rigid bodies are treated as if they were unconnected during a solver iteration and shape matching constrains are applied afterwards to ensure rigidity. It is mentioned that the cost of shape matching grows quickly with the number of particles and impulse propagation is rather slow [MMC*20]. They present a pairwise particle friction model for granular materials which is also applied to rigid body particles. Frictional forces are computed and applied iteratively during the constraint solving procedure of a simulation step. The authors state that the simulation method aims at real-time performance while making compromises in realism, and as such there is no validation shown for the correctness of frictional forces depending on the parameterizable coefficient of friction. Instead, the authors report that frictional forces strongly depend on the iteration count of the constraint solver [MMCK14]. The development of *extended position-based dynamics* (XPBD) [MMC16] mitigates the dependency of constraint forces on the iteration count in position-based simulation approaches but does not explicitly discuss frictional forces. Subsequent publications based on XPBD do not demonstrate the physical correctness of the computed frictional forces either [MMC*20]. Our simulation method uses the coefficient of friction μ as a physically meaningful input parameter and is able to replicate the expected friction behavior. In their coupled fluid-rigid simulation approach, Koschier and Bender [KB17] externalize the rigid-rigid contact handling, but still apply an explicit Coulomb friction force on fluid particles next to a rigid boundary that uses pressure values to estimate normal force magnitudes. While our implicit friction implementation will use a similar idea to estimate normal force magnitudes between rigid bodies, we also determine contact normal directions based on SPH pressure forces. This way, in contrast to Koschier and Bender [KB17], no boundary geometry description other than the particle representation is required. Hu et al. [HFG*18] present an MPM method implementing a two-way coupling between rigid bodies and objects such as fluids, elastic and elastoplastic materials. However, for rigid-rigid contacts they also use an external rigid body dynamics software. Accordingly, frictional forces between rigid bodies are not discussed. In their partitioned approach, Akbay et al. [ANZS18] couple black-box solvers for fluids, other deformable objects as well as rigid bodies through a small reduced-order system. As an example, this way they are able to couple the Lagrangian fluid solver DFSPH [BK15] with the position-based rigid body solver presented by Deul et al. [DCB14]. Partitioned simulation methods have the advantage that existing specialized solvers can be reused for a combined simulation. However, the authors mention that in general, compared to partitioned approaches, monolithic approaches that unify all simulated quantities in one system are expected to be more efficient in computing a strongly coupled solution [ANZS18].

More recently, Gissler et al. [GPB*19] presented a partitioned simulation method to couple particle fluids with particle-based rigid bodies. Both, the fluid solver and the rigid body solver are purely based on SPH, such that interleaving both solvers is particularly straightforward. The rigid body solver detects potential collisions between bodies by considering density deviations at particles and prevents interpenetrations using pressure forces. To model friction between rigid bodies they propose to use an explicit Coulomb friction implementation. As we will demonstrate in Section 5.1, this friction implementation cannot reproduce static friction and is not guaranteed to result in the physically correct amount of friction force. Also, frictional forces are not guaranteed to preserve momentum. Our rigid body solver uses density deviations to detect contacts and pressure to resolve collisions as well, however, in contrast to Gissler et al. [GPB*19], we present a truly monolithic solver that strongly couples rigid body contact handling with a particle fluid simulation. Additionally, the pressure system is extended by a novel momentum-conserving implicit Coulomb friction force formulation that correctly reproduces static friction and stick-slip transitions. This way, we need to solve only one global system to simultaneously handle fluid incompressibility constraints, fluid-rigid interface forces, rigid-rigid contacts as well as rigid-rigid friction forces. The unified system makes it easier to use more advanced solving methods, such as the nonsmooth nonlinear conjugate gradient method (NNCG) proposed by Silcowitz et al. [SHNE10]. Compared to the relaxed Jacobi method employed by Gissler et al. [GPB*19], we are able to show performance gains above a factor of ten.

3. Method

A complete rigid body contact handling procedure includes resolving potential collisions between simulated bodies and computing corresponding frictional effects. In this section, we describe the main concepts of our monolithic simulation method that is purely based on SPH discretizations. Section 3.1 shows how SPH can be used to detect contacts between rigid bodies as well as contacts at the interface between fluid and rigid bodies. The definition of contact is then used in Section 3.2 to construct the pressure *linear complementarity problem* (LCP) which forms the main optimization problem that needs to be solved in order to handle rigid contacts and fluid incompressibility. Similarly, in Section 3.3 we introduce the optimization problem for frictional forces following the exact Coulomb friction model. We will see that both optimization problems form a system since they mutually depend on each other. Afterwards, in Section 4 we describe how this system can be solved in an implementation.

3.1. Contact Detection

Typical simulation methods, that are handling contacts between bodies represented as triangle meshes, perform a collision detection step at the beginning of each simulation iteration in order to identify all contact points in the scene. This is a nontrivial task and the quality of the contacts generated by the collision detection can severely impact performance, robustness and correctness of the simulation result [BET14, Erl18, AE21, WFS*21]. To make matters worse, good quality of contact points is elusive and neither rigorously understood nor defined [Erl18]. Additionally, comparing simulators becomes difficult as often the underlying contact generation method is not sufficiently specified [Erl18]. In our particle-based simulation framework on the other hand, inspired by previous work (e.g. [AIA*12, GPB*19]), the surface of rigid bodies is sampled with SPH particles such that collision detection can be boiled down to a volume estimation of particles using a local particle-neighborhood. Neighboring particles can be reliably found using well studied neighborhood search methods [IABT11, IOS*14, WSG*18, BGT19]. Using SPH, we compute the volume V_r of a particle r that lies on the surface of a rigid body R with

$$V_r = \frac{1}{\sum_{r_r} W_{rr_r} + \sum_{r_k} W_{rr_k} + \sum_{r_b} W_{rr_b}} \tag{3.1}$$

where r_r are neighboring rigid particles belonging to the same rigid body R as r, r_k represent neighboring rigid particles belonging to other rigid bodies and r_b are neighboring particles that are part of a kinematic boundary. W is the SPH smoothing kernel function. We use the notation W_{rr_r} shorthand for $W\left(|\vec{x}_r - \vec{x}_{r_r}|\right)$ with particle positions $\vec{x}$. Additionally, if no time index is given we refer to current positions $\vec{x}(t)$. Following Solenthaler and Pajarola [SP08], Akinci et al. [AIA*12] and Gissler et al. [GPB*19], the rest volume V_r^0 of particle r is approximated at the start of the simulation by considering surrounding particles r_r sampling the same rigid body R:

$$V_r^0 = \frac{\gamma}{\sum_{r_r} W_{rr_r}} \tag{3.2}$$

We now define that a particle r belonging to a rigid body is in a state of collision if

$$V_r < V_r^0. \tag{3.3}$$

In Equation 3.2, γ is a correction coefficient that accounts for the fact that rigid bodies are only sampled at the surface, whereas an SPH approximation of the volume assumes a complete particle-neighborhood. With γ, we can control how many contributions from neighboring particles are necessary for a particle r to be considered in a state of collision [GPB*19] as illustrated in Figure 2. In our simulations γ is set equal to 0.7 as we found this value to be high enough such that contacts between small and thin structures are still detected.

Traditionally, collision detection algorithms search for vertex-face and edge-edge collisions with the aim of returning a set of collisions that individually describe the contact between exactly two bodies [BFA02, KSJP08, OTSG09, BET14, AE21]. As a major difference to most existing approaches, we do not explicitly define contact points between rigid bodies since the volume of a particle r can be influenced by particles r_k belonging to multiple rigid bodies nearby, and a contact between rigid bodies can cause multiple particles on both sides of the contact to be in a state of collision. Consequently, our contact handling and friction force computation will also be based on the particle description of rigid bodies instead of generated collision points.

We want to point out that most recent SPH-based pressure solvers use a similar constraint on the volumes of fluid particles f to define incompressibility. Here, the solvers preserve a compression free fluid state where no fluid particle f has

$$V_f < V_f^0 \tag{3.4}$$

throughout the simulation to prevent any volume loss in the fluid [SP09, ICS*14, BK15]. Fluids are sampled volumetrically, so the rest volume V_f^0 of fluid particles f is set to

$$V_f^0 = h^3, \tag{3.5}$$

where h is the particle spacing. The current volume V_f can be approximated using

$$V_f = \frac{1}{\sum_{f_f} W_{ff_f} + \sum_{f_k} W_{ff_k} + \sum_{f_b} W_{ff_b}} \tag{3.6}$$

with neighboring fluid particles f_f, neighboring particles f_k that sample rigid bodies and neighboring particles f_b belonging to a kinematic boundary. In Section 3.2, we will see that the similarity between incompressibility constraints allows us to comfortably combine the volume constraints of rigid and fluid particles into one global system.

3.1.1. Discussion

Using particles to detect and handle rigid body collisions entails interesting differences compared to traditional mesh-based schemes. Precision of contact handling no longer purely depends on mesh geometry, but instead the sampling density of particles governs the resolution of contact handling. Existing work such as Allard et al. [AFC*10] indicate that it can be advantageous to abstract contacts away from traditional vertex-triangle and edge-edge collision pairs. In the specific case of Allard et al. [AFC*10], one intersection volume is constructed per contact which is then subdivided by a regular grid into multiple subvolumes to resolve contacts more accurately. By adjusting the grid resolution and particle sampling density respectively, Allard et al. [AFC*10] and our approach both can tune contact resolution independently of the complexity of the underlying mesh. As a consequence demonstrated in Section 5.4, highly detailed meshes can be simulated using a more appropriate precision. On the other hand, even though at first glance it seems excessive to sample simple meshes with a high number of particles, the increased contact resolution allows us to simulate pressure and friction distributions over contacting surface patches since potentially all particles within the contact area are detected to be in a state of collision. This contrasts with most rigid body simulation methods which only consider a few distinct collision points per contact area [BET14, Erl18]. We refer to Allard et al. [AFC*10] for an interesting demonstration on how the number of contact points per colliding bodies is indeed relevant for replicating realistic body motion.

As a disadvantage of the employed contact detection using particle volumes, it shall be mentioned that if geometrically highly accurate collision points based on mesh descriptions of bodies are required, the number of particles necessary to achieve such a high resolution might significantly reduce simulation performance.

Parameter γ: Next to the particle resolution, parameter $\gamma \in (0,1)$ influences contact detection for rigid bodies by scaling the rest volumes of rigid particles. The effect of γ is illustrated in Figure 2. As already mentioned, decreasing γ towards zero increases the required contributions from neighboring particles before a rigid particle is considered to be in a state of collision. Thus, we need to make sure γ is set sufficiently large such that no thin body features, sampled with few particles, can penetrate through another body surface without causing a collision to be detected. On the flip side, setting γ too close to one causes particles to be already considered in collision even if rigid bodies are still quite far away from each other. This problem could be mitigated by offsetting particles along the surface normal towards the inside of the body. However, similar to Gissler et al. [GPB*19], as a more general solution requiring less hand tuning and that works for open meshes, we found that using $\gamma = 0.7$ results in a robust and accurate contact detection with no need to offset particles. A more theoretical motivation behind $\gamma = 0.7$ is given by Gissler et al. in [GPB*19].

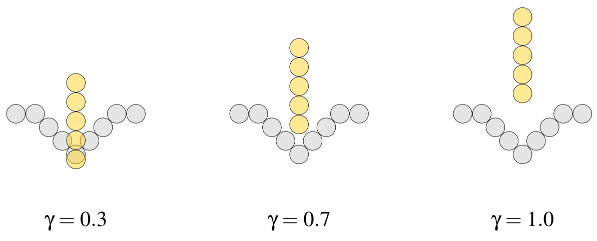


Figure 2: An illustration of the effect of γ on the contact detection between two rigid bodies. With $\gamma = 0.3$, interpenetrations between bodies may occur since contributions from neighboring particles might not be enough to detect a collision. This is especially problematic for thin geometry features such as the yellow rod displayed. Setting $\gamma = 1.0$ causes large gaps between rigid bodies and finer features such as the notch in the gray body are ignored due to the collision between the bodies being detected too early. We achieve the desired result by setting γ around 0.7.

3.2. Pressure System

In this section we define the pressure LCP that constrains all particle volumes. By solving the pressure LCP as shown in Section 4, we obtain pressure forces that preserve the volume of all particles in the simulation and as such we ensure that fluid incompressibility is met and no rigid bodies are intersecting. After defining the pressure LCP, in this section we also derive the relations between pressure and particle volume since they are required in a solver implementation.

To derive the pressure LCP, we start by defining a volume error V_i^{err} at each particle i with

$$V_i^{\text{err}}(t+\Delta t) := 1 - \frac{V_i^0}{V_i(t+\Delta t)} \tag{3.7}$$

and reconsider the incompressibility constraint from Equations 3.3 and 3.4 at the next timestep $t+\Delta t$ for some fluid or rigid particle i:

$$V_i(t+\Delta t) \geq V_i^0 \tag{3.8a}$$

$$\Leftrightarrow \quad V_i^{\text{err}}(t+\Delta t) \geq 0 \tag{3.8b}$$

Note that only negative volume error corresponds to a compression of a particle, while positive volume error indicates that a particle is in a stress-free state with larger volume compared to its rest volume.

The goal of the pressure solver is to compute pressure values p_i such that Equation 3.8b holds true, meaning that the predicted volume error V_i^{err} at the next timestep $t+\Delta t$ must be greater or equal to zero for all rigid and fluid particles i. Since we consider volume errors at particles at the next timestep $V_i^{\text{err}}(t+\Delta t)$, which already depend on current pressure, we obtain an implicit pressure formulation. Similar to Gissler et al. [GPB*19], we solve for unknown pressure values p_i at rigid and fluid particles i and derive pressure forces $\vec{F}_i^P$ based on those pressure values. To prevent unphysical behavior, we further restrict p_i to be non-negative. This is a known method used in SPH pressure solvers [ICS*14, BK15] to prevent attraction forces between particles due to negative pressure and can be compared to the non-negativity constraint of normal forces acting at rigid body contacts [Erl07, Erl13, BET14, MEM*19, AE21]. Additionally, we only allow p_i to be non-zero if the predicted volume error at the next timestep $V_i^{\text{err}}(t+\Delta t)$ equals zero. Together, the constraints on $V_i^{\text{err}}(t+\Delta t)$ and p_i form an LCP [ANE17] which can be written down using the more compact notation

$$0 \le p_i \qquad \perp \qquad V_i^{\text{err}}(t+\Delta t) \ge 0. \tag{3.9}$$

Note that on a high level, this pressure LCP closely resembles LCPs known from traditional mesh-based contact handling methods [BET14, Erl13, MEM*19]. Typically, solving the LCPs means finding scalar values for normal forces, or in our case pressure values, that cause some collision term, often a relative velocity [BET14] but in our case $V_i^{\text{err}}(t+\Delta t)$, to become non-negative. Typical velocity-level contact handling approaches require constraint stabilization methods to prevent bodies from drifting into each other over time [TBV12, BET14, Erl17, AE21]. These methods often introduce ad-hoc forces into the simulation and are known for causing stability issues [BET14, Erl07, MEM*19]. In contrast, our simulation method explicitly computes volumes in each simulation step based on the distances to neighboring particles and thus the necessity for any constraint stabilization that prevents intersections over time is eliminated. Due to the similarity of the LCPs, in principle the same solving mechanisms employed in traditional mesh-based simulation methods can be used to solve Equation 3.9.

To be able to find a solution to Equation 3.9, we must know how $V_i^{\text{err}}(t+\Delta t)$ depends on all unknown pressure values $\mathbf{p}$. Thus, in the following we derive the relation between $V_i^{\text{err}}(t+\Delta t)$ and $\mathbf{p}$ for rigid particles r and fluid particles f.

Rigid Particles: We begin with the definition of the volume error from Equation 3.7 and also consider Equation 3.1 describing the computation of V_r. Distances between particles r and r_r belonging to the same rigid body never change, so contributions from the first sum are constant in time. In our simulation, we employ an Euler-Cromer time integration scheme. Together with a first order Taylor expansion of $W(t+\Delta t)$ we can write

$$\begin{aligned} & V_r^{\text{err}}(t+\Delta t) \ge 0 \\ \Leftrightarrow \qquad & 1 - \frac{V_r^0}{V_r(t)} - V_r^0 \Delta t \sum_{r_k} \left(\vec{v}_r(t+\Delta t) - \vec{v}_{r_k}(t+\Delta t)\right) \cdot \vec{\nabla} W_{rr_k} \\ & -V_r^0 \Delta t \sum_{r_b} \left(\vec{v}_r(t+\Delta t) - \vec{v}_{r_b}(t+\Delta t)\right) \cdot \vec{\nabla} W_{rr_b} \ge 0 \end{aligned} \tag{3.10}$$

Equation 3.10 approximates the dependency of the predicted volume error $V_r^{\text{err}}(t+\Delta t)$ on the velocity field $\vec{v}$ at the next timestep $t+\Delta t$. Velocities of kinematic boundary particles r_b are predefined and not influenced by pressure forces. What remains is finding the relation between the velocity $\vec{v}_r(t+\Delta t)$ of a rigid particle r, pressure forces $\vec{F}^P$ and pressure values p. We predict $\vec{v}_r(t+\Delta t)$ with

$$\vec{v}_r(t+\Delta t) = \vec{v}_R(t+\Delta t) + \vec{\omega}_R(t+\Delta t) \times \left(\vec{x}_r(t) - \vec{x}_R(t)\right) \tag{3.11a}$$

$$\vec{v}_R(t+\Delta t) = \vec{v}_R^* + \Delta t \frac{1}{M_R} \sum_{\tilde{r}\in R} \left(\vec{F}_{\tilde{r}}^P + \vec{F}_{\tilde{r}}^F\right) \tag{3.11b}$$

$$\begin{aligned} \vec{\omega}_R(t+\Delta t) = \vec{\omega}_R^* &+ \Delta t \mathbf{I}_R^{-1}(t) \sum_{\tilde{r}\in R} \left(\vec{x}_{\tilde{r}}(t) - \vec{x}_R(t)\right) \times \vec{F}_{\tilde{r}}^P \\ &+ \Delta t \mathbf{I}_R^{-1}(t) \sum_{\tilde{r}\in R} \vec{\tau}_{\tilde{r}}^F . \end{aligned} \tag{3.11c}$$

Here, $\vec{v}_R$ and $\vec{\omega}_R$ are translational and angular velocities of rigid body R, R also represents the set of all rigid particles $\tilde{r}$ sampling R, M_R is the mass of body R, $\vec{x}_R$ is the center of mass of R, and $\mathbf{I}_R^{-1}$ denotes the inverted inertia tensor of R. The velocities $\vec{v}_R^*$ and $\vec{\omega}_R^*$ represent intermediate translational and angular velocity body R has right before the pressure solve. Thus, $\vec{v}_R^*$ and $\vec{\omega}_R^*$ include all explicitly computed forces and accelerations which, for a rigid body, typically are gravity $\vec{g}$ and the gyroscopic force [Ben07, BET14]:

$$\vec{v}_R^* = \vec{v}_R(t) + \Delta t \vec{g} \tag{3.12a}$$

$$\vec{\omega}_R^* = \vec{\omega}_R(t) + \Delta t \mathbf{I}_R^{-1}(t) \left(\mathbf{I}_R(t)\vec{\omega}_R(t)\right) \times \vec{\omega}_R(t). \tag{3.12b}$$

Now, the last unknown in Equation 3.11 are pressure forces $\vec{F}^P$ and frictional effects $\vec{F}^F$ and $\vec{\tau}^F$. Section 3.3 describes in detail how frictional forces are computed, however, their embedding in the pressure system already indicates the employed strong coupling between frictional and pressure forces. To compute $\vec{F}^P$, we use a symmetric SPH pressure gradient estimation that uses a volume formulation (as shown before by e.g. [BGI*18]):

$$\begin{aligned} \vec{F}_r^P &= -V_r \vec{\nabla} p_r \\ &= -V_r \sum_{r_k} V_{r_k} \left(p_{r_k} + p_r\right) \vec{\nabla} W_{rr_k} \\ &\quad - V_r \sum_{r_b} V_{r_b} \left(p_{r_b} + p_r\right) \vec{\nabla} W_{rr_b} \\ &\quad - V_r \sum_{r_f} 2 V_{r_f} p_{r_f} \vec{\nabla} W_{rr_f} \end{aligned} \tag{3.13}$$

Our method is flexible about the computation of pressure values p_{r_b} at kinematic boundary particles r_b, in the sense that any boundary handling method that defines a pressure value at neighboring boundary particles r_b (e.g. [AIA*12, BGI*18, BGPT18]) could be used to define p_{r_b}. In the following, we will assume pressure is mirrored with $p_{r_b} := p_r$. Also note that pressure values of neighboring fluid particles p_{r_f} are considered in the pressure force estimation, enabling our strong coupling between rigid bodies and fluids. In summary, Equations 3.10 to 3.13 describe the relation between the volume error V_r^{err} of a rigid particle r and the vector of all unknown pressure values $\mathbf{p}$.

Fluid Particles: In our simulation method, pressure at fluid particles f and pressure at rigid particles r are computed simultaneously. Since both have the same constraint on the volume to enforce incompressibility, we can derive the dependency of $V_f^{\text{err}}(t+\Delta t)$ on $\mathbf{p}$ for fluid particles f in a very similar manner. Expanding Equation 3.8b yields

$$\begin{aligned}1 - \frac{V_f^0}{V_f(t)} &- V_f^0 \Delta t \sum_{f_f} \left(\vec{v}_f(t+\Delta t) - \vec{v}_{f_f}(t+\Delta t) \right) \cdot \vec{\nabla} W_{f f_f} \\ &- V_f^0 \Delta t \sum_{f_k} \left(\vec{v}_f(t+\Delta t) - \vec{v}_{f_k}(t+\Delta t) \right) \cdot \vec{\nabla} W_{f f_k} \\ &- V_f^0 \Delta t \sum_{f_b} \left(\vec{v}_f(t+\Delta t) - \vec{v}_{f_b}(t+\Delta t) \right) \cdot \vec{\nabla} W_{f f_b} \geq 0.\end{aligned} \tag{3.14}$$

The computation of $\vec{v}_{f_k}(t+\Delta t)$ for rigid particles f_k is already known from Equation 3.11, the velocity $\vec{v}_f(t+\Delta t)$ of fluid particles f is predicted using

$$\begin{aligned}\vec{v}_f(t+\Delta t) &= \vec{v}_f^* + \Delta t \frac{1}{m_f} \vec{F}_f^P \\ &= \vec{v}_f^* - \Delta t \frac{V_f}{m_f} \sum_{f_f} V_{f_f} \left(p_{f_f} + p_f \right) \vec{\nabla} W_{f f_f} \\ &\quad - \Delta t \frac{V_f}{m_f} \sum_{f_b} V_{f_b} \left(p_{f_b} + p_f \right) \vec{\nabla} W_{f f_b} \\ &\quad - \Delta t \frac{V_f}{m_f} \sum_{f_k} 2 V_{f_k} p_f \vec{\nabla} W_{f f_k}\end{aligned} \tag{3.15}$$

where $m_f = V_f^0 \rho_f^0$ is the mass of fluid particle f with fluid rest density ρ^0. Similar to above, we use pressure mirroring at the boundary with $p_{f_b} := p_f$. Due to their symmetry, Equations 3.13 and 3.15 guarantee that pressure forces conserve momentum exactly. As a small but important detail we want to point out that pressure forces between fluid particle f and neighboring rigid particle f_k are solely defined by fluid pressure p_f. We do not consider p_{f_k} to prevent high pressure accelerations of relatively lightweight fluid particles f next to a rigid-rigid contact that typically causes much higher but punctual pressure at contacting rigid particles f_k. Again, $\vec{v}_f^*$ is the velocity of particle f including all forces computed prior to the pressure solve, so we have

$$\vec{v}_f^* = \vec{v}_f(t) + \Delta t \vec{g} + \Delta t \frac{1}{m_f} \vec{F}_f^E \tag{3.16}$$

where $\vec{F}_f^E$ are additionally computed forces such as viscosity (e.g. [Mon92, WKBB18]) and surface tension (e.g. [AAT13]). Using Equations 3.10 to 3.16, the pressure solver presented in Section 4 aims at computing pressure values that fulfill Equation 3.9 at all fluid and rigid particles, and as such all at once prevents rigid bodies from penetrating into each other, handles interface forces between fluids and rigid bodies and guarantees fluid incompressibility.

3.3. Frictional Forces

In the previous section, we defined the optimization problem that can to be solved for pressure values $\mathbf{p}$ that prevent compressions at all particles. Here, we do the same for friction forces. We first define the underlying optimization problem and then provide details that will be required by the solving procedure described in Section 4.

We chose to model frictional forces according to the exact Coulomb friction constraint. More precisely, we are searching for friction multipliers $\vec{\lambda}_r$ that are bound by the friction cone $\mathcal{F}_r$:

$$\vec{\lambda}_r \in \mathcal{F}_r \tag{3.17a}$$

$$\mathcal{F}_r := \left\{ \vec{\lambda} \in \mathbb{R}^3 \;\middle|\; \vec{\lambda} \cdot \vec{n}_r = 0 \wedge |\vec{\lambda}| \leq \mu_r |\vec{F}_r^N| \right\} \tag{3.17b}$$

where $\vec{n}_r$ is the contact normal direction, μ_r is the coefficient of friction and $\vec{F}_r^N$ represents some measurement of the normal force acting at particle r. Note that we distinguish between frictional multipliers $\vec{\lambda}$ and frictional forces $\vec{F}^F$ or frictional torque $\vec{\tau}^F$ as friction multipliers $\vec{\lambda}_r$ only consider frictional effects at particle r due to normal forces induced by pressure at particle r. Those frictional effects encoded in $\vec{\lambda}_r$ need to be mirrored onto neighboring rigid particles to result in physically correct frictional forces $\vec{F}_r^F$ and frictional torque $\vec{\tau}_r^F$. We will describe this in greater detail later in this section. In addition to the Coulomb constraint, frictional multipliers should follow the principle of maximum dissipation stating that from all possible $\vec{\lambda}$ inside the friction cone $\mathcal{F}_r$, the friction multiplier $\vec{\lambda}_r$ should fulfill [Erl17]

$$\begin{aligned}\vec{\lambda}_r &= \underset{\vec{\lambda} \in \mathcal{F}_r}{\arg\min}\, \vec{\lambda} \cdot \vec{v}_r^{\text{tang}} \\ &= -\mu_r |\vec{F}_r^N| \frac{\vec{v}_r^{\text{tang}}}{|\vec{v}_r^{\text{tang}}|} \qquad \text{if} \quad \vec{v}_r^{\text{tang}} \neq \vec{0}\end{aligned} \tag{3.18}$$

with tangential velocities $\vec{v}_r^{\text{tang}}$. By considering tangential velocities $\vec{v}_r^{\text{tang}}(t+\Delta t)$ at the next timestep $t+\Delta t$ we obtain an implicit friction force formulation.

Particle Framework: Computing frictional forces in a particle framework imposes further challenges concerning the contact geometry. While mesh-based collision detection usually returns one contact point and a contact normal direction [AE21], for a particle that is detected to be in a state of collision, these properties need to be estimated from neighboring particles. Gissler et al. [GPB*19] propose to use an SPH summation to estimate the contact normal direction $\vec{n}_r$ at a particle r:

$$\vec{n}_r = \vec{x}_r - \frac{\sum_{r_k} \vec{x}_{r_k} W_{r r_k}}{\sum_{r_k} W_{r r_k}} \tag{3.19}$$

Here, $\vec{n}$ solely depends on neighboring particle positions and the normal force $\vec{F}_r^N$ needs to be estimated separately using collision impulses. Since our simulation method solves pressure and frictional forces simultaneously, we propose to directly use intermediate pressure values p to estimate $\vec{F}_r^N$ and $\vec{n}_r$. For this, we consider the contribution of p_r to its pressure force $\vec{F}_r^P$ for each particle r as described in Equation 3.13:

$$\begin{aligned}\vec{F}_r^N &= -p_r V_r \sum_{r_k} V_{r_k} \vec{\nabla} W_{r r_k} \\ &\quad - 2 p_r V_r \sum_{r_b} V_{r_b} \vec{\nabla} W_{r r_b}\end{aligned} \tag{3.20a}$$

$$\vec{n}_r = \frac{\vec{F}_r^N}{|\vec{F}_r^N|} \tag{3.20b}$$

This does not only eliminate the need for a separate contact normal estimation process including all associated difficulties, but we can also directly relate pressure forces with the normal force used in the friction computation. In Section 5.2, we show that this new approach does not suffer from problems associated with wrongly estimated contact normals and thus trivially handles all degenerated test cases proposed by K. Erleben [Erl18]. We note that Koschier and Bender [KB17] as well as Winchenbach et al. [WAK20] also use the pressure force magnitude as an estimate for the normal force magnitude when computing frictional forces at the fluid-rigid interface. However, they do not compute the contact normal directions based on normal forces but instead derive them from the boundary geometry.

Tangential Velocities: Equation 3.18 requires the computation of tangential velocities $\vec{v}_r^{\text{tang}}(t+\Delta t)$. They can be estimated with a standard SPH approximation to compute a relative velocity $\vec{v}_r^{\text{rel}}(t+\Delta t)$ at particle r and projecting $\vec{v}_r^{\text{rel}}(t+\Delta t)$ onto the tangential contact plane afterwards using $\vec{n}_r$:

$$\begin{aligned}\vec{v}_r^{\text{rel}}(t+\Delta t) &= \sum_{r_k} V_{r_k}\left(\vec{v}_r(t+\Delta t)-\vec{v}_{r_k}(t+\Delta t)\right)W_{rr_k}\\ &\quad+\sum_{r_b} V_{r_b}\left(\vec{v}_r(t+\Delta t)-\vec{v}_{r_b}(t+\Delta t)\right)W_{rr_b}\end{aligned} \tag{3.21a}$$

$$\vec{v}_r^{\text{tang}}(t+\Delta t) = \left(\mathbb{1}-\vec{n}_r\vec{n}_r^{\top}\right)\vec{v}_r^{\text{rel}}(t+\Delta t) \tag{3.21b}$$

with the identity matrix $\mathbb{1}$. From Equation 3.14 we already know how to compute $\vec{v}_r(t+\Delta t)$ and $\vec{v}_{r_k}(t+\Delta t)$. $\vec{v}_{r_b}(t+\Delta t)$ is independent of pressure and friction forces, so in this context we set $\vec{v}_{r_b}(t+\Delta t)=\vec{v}_{r_b}(t)$.

Friction Force Mirroring: Friction multipliers $\vec{\lambda}_r$ need to be mirrored onto neighboring rigid bodies to result in physically consistent friction forces. Equivalent to pressure forces, we distribute the friction multipliers considering the weighting by the kernel gradient. The actual friction forces $\vec{F}_r^F$ and torques $\vec{\tau}_r^F$ applied to rigid particles r are then given by

$$\begin{aligned}\vec{F}_r^F &= \sum_{r_k}\vec{F}^F_{r\leftarrow r_k}+\sum_{r_b}\vec{F}^F_{r\leftarrow r_b}\\ &= \sum_{r_k}\left(\widetilde{V}_r\vec{\lambda}_r-\widetilde{V}_{r_k}\vec{\lambda}_{r_k}\right)W_{rr_k}\\ &\quad+\sum_{r_b}\widetilde{V}_r\vec{\lambda}_r W_{rr_b}\end{aligned} \tag{3.22a}$$

$$\begin{aligned}\vec{\tau}_r^F &= \sum_{r_k}\vec{\tau}^F_{r\leftarrow r_k}+\sum_{r_b}\vec{\tau}^F_{r\leftarrow r_b}\\ &= \sum_{r_k}\left(\frac{1}{2}\vec{x}_r+\frac{1}{2}\vec{x}_{r_k}-\vec{x}_R\right)\times\left(\widetilde{V}_r\vec{\lambda}_r-\widetilde{V}_{r_k}\vec{\lambda}_{r_k}\right)W_{rr_k}\\ &\quad+\sum_{r_b}\left(\frac{1}{2}\vec{x}_r+\frac{1}{2}\vec{x}_{r_b}-\vec{x}_R\right)\times\widetilde{V}_r\vec{\lambda}_r W_{rr_b}\end{aligned} \tag{3.22b}$$

with

$$\begin{aligned}\widetilde{V}_r &= \frac{1}{\sum_{r_k}W_{rr_k}+\sum_{r_b}W_{rr_b}}\\ &= \frac{1}{\frac{1}{V_r(t)}-\frac{\gamma}{V_r^0}}.\end{aligned} \tag{3.23}$$

It is easy to see that the friction force $\vec{F}_r^F$ includes its own friction multiplier $\vec{\lambda}_r$ as well as mirrored contributions from neighboring rigid particles r_k. The friction force mirroring guarantees exact conservation of linear and angular momentum. Even though momentum conservation is no concern when interacting with kinematic boundaries, neighboring particles r_b belonging to such a boundary are treated consistently. Equations 3.22a and 3.22b describe linear relationships between $\vec{F}_r^F$, $\vec{\tau}_r^F$ and the vector of all unknown friction multipliers $\boldsymbol{\lambda}$, and as such can be smoothly embedded into our system relating $\vec{v}_r^{\text{tang}}(t+\Delta t)$ with $\boldsymbol{\lambda}$. The modified volume $\widetilde{V}_r$ is easily computed on-the-fly as described in Equation 3.23. In Section 5.1 we demonstrate the correctness and accuracy of the computed friction forces.

Summary: Since we employ an implicit force computation model, we need to be able to describe the relation between the vector of all unknown friction multipliers $\boldsymbol{\lambda}$ and $\vec{v}_r^{\text{tang}}(t+\Delta t)$ for all particles r. In the following, we give a short summary on how $\vec{v}_r^{\text{tang}}(t+\Delta t)$ is computed. Given the frictional multipliers $\boldsymbol{\lambda}$ specified by Equations 3.17 and 3.18, we start by mirroring $\boldsymbol{\lambda}$ onto neighboring bodies using Equations 3.22a and 3.22b, to obtain frictional forces $\vec{F}^F$ and torques $\vec{\tau}^F$ for all rigid particles. Frictional forces and torques, together with pressure forces $\vec{F}^P$, can be plugged into Equation 3.11 to obtain an estimate of $\vec{v}_r(t+\Delta t)$ for all r. The newly estimated velocities are inserted into Equation 3.21 to compute $\vec{v}_r^{\text{tang}}(t+\Delta t)$.

We point out that the estimate for $V_i^{\text{err}}(t+\Delta t)$ required in the pressure system as well as the estimate for $\vec{v}_r^{\text{tang}}(t+\Delta t)$ required in the friction system both simultaneously consider pressure forces acting in the fluid, at fluid-rigid interfaces and between rigid bodies together with frictional forces. This way, we achieve a strong coupling between contact forces and friction through one global monolithic system, which is crucial for simulation stability when considering rigid bodies [KSJP08]. While this is common practice in pure rigid body simulation frameworks [BET14, Erl17, PAK*19, MEM*19, MEM*20, FLS*21], we are not aware of an existing simulation method that strongly couples internal fluid pressure forces, fluid-rigid interface forces, rigid-rigid contact forces with dry friction forces. In Section 4, we provide an illustration of an efficient solver implementation that is able to compute $\mathbf{p}$ and $\boldsymbol{\lambda}$ satisfying the constraints given in Equations 3.9, 3.17 and 3.18.

4. Implementation

The optimization problems presented in Equations 3.9, 3.17 and 3.18 form the foundation of the pressure and friction solving procedure. In this section we now build a solver implementation that is able to efficiently solve the previously presented optimization problems for pressure $\mathbf{p}$ and friction $\boldsymbol{\lambda}$. For this purpose, both, the pressure LCP (Equation 3.9) and the friction computation problem (Equations 3.17 and 3.18) are translated into an equivalent fixed point formulation [JAJ98, SNT11, Erl17]:

$$p_i = \operatorname{prox}_{\mathcal{P}_i}\left[p_i-\alpha_i^P V_i^{\text{err}}(t+\Delta t)\right] \tag{4.1a}$$

$$\vec{\lambda}_r = \operatorname{prox}_{\mathcal{F}_r}\left[\vec{\lambda}_r-\alpha_r^F\vec{v}_r^{\text{tang}}(t+\Delta t)\right] \tag{4.1b}$$

where $\mathcal{P}_i$ is the allowed solution space of p_i and $\mathcal{F}_r$ the set of all friction forces inside the friction cone:

$$\mathcal{P}_i := \left\{\, p\in\mathbb{R} \;\middle|\; p\geq 0 \,\right\} \tag{4.2}$$

$$\mathcal{F}_r := \left\{\, \vec{\lambda}\in\mathbb{R}^3 \;\middle|\; \vec{\lambda}\cdot\vec{n}_r=0 \wedge |\vec{\lambda}|\leq\mu_r|\vec{F}_r^N| \,\right\} \quad \text{(3.17b revisited)}$$

The *proximal operator* $\mathrm{prox}_{\mathcal{S}}$ projects its input onto the nearest point in some given set $\mathcal{S}$:

$$\mathrm{prox}_{\mathcal{S}}[x] := \underset{s\in\mathcal{S}}{\arg\min}\,|s-x|^2 \tag{4.3}$$

We refer to Schindler et al. [SNT11] for a thorough derivation of Equation 4.1 from the pressure LCP and the friction constraints. In Equation 4.1, the scalar parameters α^P and α^F have no effect on the analytical solution of a fixed point as long as $\alpha^P, \alpha^F > 0$. In an implementation however, they greatly influence convergence speed and stability of an iterative solver. By iteratively updating p_i and $\vec{\lambda}_r$ using Equation 4.1, one generates a sequence of iterates that converges locally if the spectral radius of the Jacobian of the right-hand side of Equation 4.1 is smaller than one [Erl17]. This requirement can be met by setting α^P and α^F sufficiently small [FGNU06, PB14]. While there exist approaches that employ backtracking strategies to find safe values for α^P and α^F [Erl17], we found that using values that correspond to a modified Jacobi update similar to Tonge et al. [TBV12] and Gissler et al. [GPB*19] also leads to a robust and performant simulation.

4.1. Jacobi Scheme

We can now build an iterative Jacobi scheme to solve the fixed point problem in Equation 4.1 by choosing appropriate values for α_i^P and α_r^F. Similar to Gissler et al. [GPB*19], we compute α_i^P such that Equation 4.1a corresponds to a relaxed Jacobi update step with relaxation coefficient υ where for rigid particles, the number of contacts n_R of body R is additionally taken into account:

$$\alpha_r^P = \frac{\upsilon}{\mathrm{n}_R}\left(\frac{\partial}{\partial p_r}V_r^{\mathrm{err}}(t+\Delta t)\right)^{-1} \tag{4.4a}$$

$$\alpha_f^P = \upsilon\left(\frac{\partial}{\partial p_f}V_f^{\mathrm{err}}(t+\Delta t)\right)^{-1} \tag{4.4b}$$

The terms $\partial/\partial p_i\, V_i^{\mathrm{err}}(t+\Delta t)$ are an inherent part of the Jacobi update. They are used as an estimate of how much $V_i^{\mathrm{err}}(t+\Delta t)$ changes depending on the unknowns p_i. Contrary to the suggestion of K. Erleben [Erl17], we do not use a Gauss-Seidel variant to solve Equations 4.1a and 4.1b. While the increased stability might outweigh the loss of parallel computation power for mesh-based rigid body simulators, in our particle-based contact handling method including fluids, there are easily hundreds of thousands of simultaneously active contacts at each simulation step. Handling this large number of contacts requires a parallelized solving procedure. It has been shown that Jacobi methods require additional stabilizing when treating rigid contacts [TBV12, GPB*19], which is why we divide the relaxation coefficient by n_R. The number of contacts n_R is updated in each Jacobi iteration using

$$\mathrm{n}_R = \sum_{\tilde{r}\in R}\begin{cases}1 & \text{if}\quad V_{\tilde{r}}^{\mathrm{err}}(t+\Delta t) < 0\\ 0 & \text{else}\end{cases} \tag{4.5}$$

where $V_{\tilde{r}}^{\mathrm{err}}(t+\Delta t)$ is the current prediction for the volume error at particle $\tilde{r}$ at the next timestep. The interlinked pressure computation method by Gissler et al. [GPB*19] uses a generic pressure solver to compute pressure in fluids. However, generic fluid solvers are not able to capture the effect velocity changes of rigid particles f_k due to pressure p_f have onto the volume error V_f^{err}. Thus, there are missing contributions in the computation of $\partial/\partial p_f\, V_f^{\mathrm{err}}(t+\Delta t)$ that can impact the convergence behavior of the Jacobi solver. In contrast, our method unifies the pressure computation for rigid and fluid particles into one system, and thus the dependency of $\vec{v}_{f_k}(t+\Delta t)$ on p_f can be correctly included in the computation of $\partial/\partial p_f\, V_f^{\mathrm{err}}(t+\Delta t)$. The derivation and an efficient computation implementation of the elements $\partial/\partial p_i\, V_i^{\mathrm{err}}(t+\Delta t)$ are shown in Appendix A.

Friction Update: When applying a fixed point iteration for frictional forces as shown in Equation 4.1b, we found it to be important that α^F truly is a positive scalar value. Blocked Jacobi schemes that use a matrix for α^F [CPS09] often cause $\alpha_r^F\vec{v}_r^{\mathrm{tang}}(t+\Delta t)$ to point into a different direction than $\vec{v}_r^{\mathrm{tang}}(t+\Delta t)$. This has the effect that frictional multipliers $\vec{\lambda}_r$ fulfilling the principle of maximum dissipation are no longer a solution to the fixed point problem described in Equation 4.1b. K. Erleben [Erl17] evaluated the convergence behavior of blocked solver strategies in a very similar context and the results seem to support our claim. Similar to the blocked approach, considering α^F independently for each coordinate direction of $\vec{\lambda}_r$ introduces the same problem, which is why in our implementation we choose

$$\alpha^F = \frac{\upsilon}{\mathrm{n}_R}\left(\frac{1}{3}\mathrm{tr}\left[\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r^{\mathrm{tang}}(t+\Delta t)\right]\right)^{-1} \tag{4.6}$$

where tr is the trace operator. Again, an efficient implementation to compute $\partial/\partial\vec{\lambda}_r\,\vec{v}_r^{\mathrm{tang}}(t+\Delta t)$ is derived in Appendix A.

Further, we slightly modify the Jacobi update for frictional multipliers $\vec{\lambda}_r$ given in Equation 4.1b. Based on the observation that for two rigid particles r_1 and r_2 with

$$\begin{aligned}\vec{v}_{r_2}^{\mathrm{tang}}(t+\Delta t) &= c\vec{v}_{r_1}^{\mathrm{tang}}(t+\Delta t)\quad\text{with}\quad c\gg 1\\ \vec{v}_{r_1}^{\mathrm{tang}}(t+\Delta t) &\neq \vec{0}\\ \mu_{r_2}|\vec{F}_{r_2}^N| &= \mu_{r_1}|\vec{F}_{r_1}^N|,\end{aligned} \tag{4.7}$$

$\vec{\lambda}_{r_1}$ and $\vec{\lambda}_{r_2}$ should converge to the same vector. However, by looking at Equation 4.1b we notice that since $\vec{v}_{r_2}^{\mathrm{tang}}(t+\Delta t)$ is much larger than $\vec{v}_{r_1}^{\mathrm{tang}}(t+\Delta t)$, $\vec{\lambda}_{r_2}$ will converge much faster compared to $\vec{\lambda}_{r_1}$. This causes severe stability issues in the solving process for large relaxation coefficients υ or alternatively impractically slow convergence for smaller υ. To stabilize the solving process we introduce a friction target $\vec{\lambda}_r^*$ into the solving process whose magnitude is clamped at $\mu_r|\vec{F}_r^N|$ and reformulate the fixed point problem from Equation 4.1b to

$$\begin{aligned}\vec{\lambda}_r^* &= \mathrm{prox}_{\mathcal{F}_r}\left[-\left(\frac{1}{3}\mathrm{tr}\left[\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r^{\mathrm{tang}}(t+\Delta t)\right]\right)^{-1}\vec{v}_r^{\mathrm{tang}}(t+\Delta t)\right]\\ \vec{\lambda}_r^{k+1} &= \mathrm{prox}_{\mathcal{F}_r}\left[\vec{\lambda}_r^k+\frac{\upsilon}{\mathrm{n}_R}\vec{\lambda}_r^*\right].\end{aligned} \tag{4.8}$$

Again, this does not change the analytical solution of the fixed point problem but merely improves convergence behavior.

4.2. Nonsmooth Nonlinear Conjugate Gradient

Relaxation methods such as the Jacobi method are known for their simplicity, robustness and flexibility, but also come with slow convergence speed, especially considering poorly conditioned problems [CPS09, SNE09, PAE10, Erl13]. To improve the performance of the proximal operator method, K. Erleben [Erl17] suggests studying a combination of proximal operators, embedded into a generalized conjugate gradient method. We follow this suggestion by extending our Jacobi iteration into a nonsmooth nonlinear conjugate gradient (NNCG) method as described by Silcowitz-Hansen et al. [SHNE10]. Here, differences between iterates of the projected Jacobi solver are seen as residuals $\mathbf{r}$

$$\begin{aligned} \mathbf{r}^k &:= \left(\mathbf{p}^{k+1} - \mathbf{p}^k, \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\right) \\ &= \mathrm{PJ}\left[\mathbf{p}^k, \boldsymbol{\lambda}^k\right] - \left(\mathbf{p}^k, \boldsymbol{\lambda}^k\right) \end{aligned} \tag{4.9}$$

where PJ returns the iterates after applying one projected Jacobi iteration as described in Equations 4.1a and 4.8. The residuals simultaneously equal the negative gradient of some function $f(\mathbf{r}^k) := \frac{1}{2}|\mathbf{r}^k|^2$. Finding $\mathbf{p}$ and $\boldsymbol{\lambda}$ such that f becomes zero is equivalent to solving the fixed point problems given in Equations 4.1a and 4.8. To find a local minimum of f, the Fletcher-Reeves nonlinear conjugate gradient method can be employed [NW06, SHNE10, AE21] as it only requires information about the gradient $\vec{\nabla} f = -\mathbf{r}$. The implementation of the NNCG method is illustrated in detail in Section 4.3. Performance comparisons between the standard Jacobi method and NNCG are shown in Section 5.3.

4.3. Algorithm

We use this section to give an overall view of our simulation method and discuss some remaining implementation details. Algorithm 1 shows a summary of our particle-based fluid-rigid simulation loop, which closely resembles the typical structure of a simulation based on SPH [IOS*14]. As we can see, a simulation step starts by searching and storing particle neighborhoods as these are required multiple times later on in various SPH summations. Similarly, the current volume $V_i(t)$ is precomputed for all rigid and fluid particles i using Equations 3.1 and 3.6, as its value is repeatedly used. Using SPH, we are able to explicitly compute forces such as viscosity and surface tension and apply them using Equation 3.16 to get intermediate velocities $\vec{v}_f^*$. Similarly, we compute intermediate velocities $\vec{v}_R^*$ and $\vec{\omega}_R^*$ for all rigid bodies R as shown in Equation 3.12. The intermediate velocities are used in the pressure and friction solver which is embedded into the simulation loop in line 11. For now, we treat the solver as a black box and end the simulation step by integrating the resulting pressure and frictional forces onto fluid particles f, rigid bodies R and rigid particles r. A precise description of how velocities and positions are integrated in time is moved to Appendix B as the expressions become quite lengthy without adding much to the understanding of the simulation method.

The pressure and friction solver shown in Algorithm 2 is the core of our simulation method. Based on predicted particle velocities $\vec{v}_i(t+\Delta t)$ we implicitly compute pressure values that prevent collisions and matching frictional forces. The implemented NNCG solver bases on Jacobi updates which are described in Algorithm 3.

```
1  $t \leftarrow 0$
2  while simulating do
3      foreach particle $i$ do
4          Find all neighbors of $i$
5          Compute $V_i(t)$                         ▷ eq. 3.1 and 3.6
6      foreach fluid particle $f$ do
7          Compute $\vec{v}_f^*$                    ▷ eq. 3.16
8      foreach rigid body $R$ do
9          Compute $\vec{v}_R^*$                    ▷ eq. 3.12a
10         Compute $\vec{\omega}_R^*$               ▷ eq. 3.12b
11     Solve for $\mathbf{p}$ and $\boldsymbol{\lambda}$   ▷ alg. 2
12     Integrate $\mathbf{x}$, $\mathbf{v}$, $\mathbf{q}$, $\boldsymbol{\omega}$
13     $t \leftarrow t + \Delta t$
```

Algorithm 1: A fluid and rigid body simulation loop based on SPH particles. The pressure and friction solver embedded in line 11 is presented in detail in Algorithm 2.

Before each Jacobi step we need to evaluate predicted volume errors $V_i^{\mathrm{err}}(t+\Delta t)$ and tangential velocities $\vec{v}_r^{\mathrm{tang}}(t+\Delta t)$ based on current pressure vector $\mathbf{p}$ and friction vector $\boldsymbol{\lambda}$. While this procedure is computationally expensive, we are able to implement it in an efficient and fully parallelized manner as demonstrated in Algorithm 4. To make Algorithm 2 easy to reimplement, a detailed explanation on how to compute the diagonal elements is given in Appendix A.

Typical SPH pressure solvers use the average predicted volume error of particles as a measurement for convergence [SP09, ICS*14, IOS*14, BK15, GPB*19]. While this ensures that volumes are preserved, there is no guarantee that resulting pressure values are a solution to the LCP shown in Equation 3.9. More precisely, pressure values that are too high are not considered as an error, what can result in an overcorrection of volume errors. To ensure that pressure values $\mathbf{p}$ are a true solution to Equation 3.9, we follow Macklin et al. [MEM*19] and use the average absolute Fischer-Burmeister function as a measurement of convergence of the pressure iterates. Similarly, to measure how close friction multipliers $\boldsymbol{\lambda}$ are to a true solution, we use the same friction constraint function as Macklin et al. [MEM*19].

5. Results

In this section we demonstrate the capabilities of our monolithic solver. We begin with a validation of our friction computation method by comparing simulation results to analytical solutions in Section 5.1 and perform some basic robustness tests in Section 5.2. To motivate the usage of the NNCG solver, we compare its convergence speed to that of a standard projected relaxed Jacobi solver in Section 5.3. More challenging scenarios are showcased in Section 5.4, including difficult frictional settings, large mass ratios, complex geometries, interaction with fluids and large numbers of simulated rigid bodies. Please refer to the supplementary video for a more detailed insight into the simulation scenarios.

```
Function Solve for $\mathbf{p}$ and $\boldsymbol{\lambda}$:
    $\mathbf{p}^0 \leftarrow \mathbf{0}$
    $\boldsymbol{\lambda}^0 \leftarrow \mathbf{0}$
    Compute Diagonals
    Update $\mathbf{V}^{\text{err}}$ and $\mathbf{v}^{\text{tang}}$ ($\mathbf{p}^0$, $\boldsymbol{\lambda}^0$)          ▷ alg. 4
    $\mathbf{p}^1, \boldsymbol{\lambda}^1 \leftarrow$ PJ ($\mathbf{p}^0$, $\boldsymbol{\lambda}^0$)          ▷ alg. 3
    $\vec{\nabla} f^0 \leftarrow -(\mathbf{p}^1 - \mathbf{p}^0, \boldsymbol{\lambda}^1 - \boldsymbol{\lambda}^0)$
    $\mathbf{s}^0 \leftarrow -\vec{\nabla} f^0$
    $k \leftarrow 1$
    Update $\mathbf{V}^{\text{err}}$ and $\mathbf{v}^{\text{tang}}$ ($\mathbf{p}^1$, $\boldsymbol{\lambda}^1$)          ▷ alg. 4
    while not converged do
        $\mathbf{p}^{k+1}, \boldsymbol{\lambda}^{k+1} \leftarrow$ PJ ($\mathbf{p}^k$, $\boldsymbol{\lambda}^k$)          ▷ alg. 3
        $\vec{\nabla} f^k \leftarrow -(\mathbf{p}^{k+1} - \mathbf{p}^k, \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k)$
        $\beta \leftarrow |\vec{\nabla} f^k| / |\vec{\nabla} f^{k-1}|$
        if $\beta > 1$ then
            $\mathbf{s}^k \leftarrow \mathbf{0}$
        else
            $(\mathbf{p}^{k+1}, \boldsymbol{\lambda}^{k+1}) \leftarrow (\mathbf{p}^{k+1}, \boldsymbol{\lambda}^{k+1}) + \beta \mathbf{s}^{k-1}$
            $\mathbf{s}^k \leftarrow \beta \mathbf{s}^{k-1} - \vec{\nabla} f^k$
        $k \leftarrow k+1$
        Update $\mathbf{V}^{\text{err}}$ and $\mathbf{v}^{\text{tang}}$ ($\mathbf{p}^k$, $\boldsymbol{\lambda}^k$)          ▷ alg. 4
```

Algorithm 2: A NNCG implementation to solve for unknown $\mathbf{p}$ and $\boldsymbol{\lambda}$ using the projected Jacobi step as an estimate for the function gradient. Afterwards, we can compute the desired pressure forces $\vec{F}_i^P$ and frictional effects $\vec{F}_r^F$ and $\vec{\tau}_r^F$ from $\mathbf{p}$ and $\boldsymbol{\lambda}$.

```
Function PJ ($\mathbf{p}^k$, $\boldsymbol{\lambda}^k$):
    foreach fluid particle $f$ do
        Compute $p_f^{k+1}$          ▷ eq. 4.1a
    foreach rigid particle $r$ do
        Compute $p_r^{k+1}$          ▷ eq. 4.1a
        Compute $\vec{\lambda}_r^{k+1}$          ▷ eq. 4.8
    return $\mathbf{p}^{k+1}$, $\boldsymbol{\lambda}^{k+1}$
```

Algorithm 3: The projected relaxed Jacobi step to update pressure $\mathbf{p}$ and friction multipliers $\boldsymbol{\lambda}$. Note that in line 6 we use the improved Jacobi step from Equation 4.8 to update $\vec{\lambda}_r$.

```
Function Update $\mathbf{V}^{\text{err}}$ and $\mathbf{v}^{\text{tang}}$ ($\mathbf{p}^k$, $\boldsymbol{\lambda}^k$):
    foreach fluid particle $f$ do
        Compute $\vec{v}_f(t+\Delta t)$          ▷ eq. 3.15
    foreach rigid body $R$ do
        foreach rigid particle $r \in R$ do
            Compute $\vec{F}_r^P$          ▷ eq. 3.13
            Compute $\vec{F}_r^F$          ▷ eq. 3.22a
            Compute $\vec{\tau}_r^F$          ▷ eq. 3.22b
        Compute $\vec{v}_R(t+\Delta t)$          ▷ eq. 3.11b
        Compute $\vec{\omega}_R(t+\Delta t)$          ▷ eq. 3.11c
    foreach fluid particle $f$ do
        Compute $V_f^{\text{err}}(t+\Delta t)$          ▷ eq. 3.11a and 3.14
    foreach rigid particle $r$ do
        Compute $V_r^{\text{err}}(t+\Delta t)$          ▷ eq. 3.10 and 3.11a
        Compute $\vec{v}_r^{\text{tang}}(t+\Delta t)$          ▷ eq. 3.11a and 3.21
```

Algorithm 4: The procedure to compute predicted volume errors $\mathbf{V}^{\text{err}}(t+\Delta t)$ and tangential velocities $\mathbf{v}^{\text{tang}}(t+\Delta t)$ based on current pressure values $\mathbf{p}^k$ and friction multipliers $\boldsymbol{\lambda}^k$. Since predicted velocities of rigid particles $\vec{v}_r(t+\Delta t)$ can be computed on-the-fly based on $\vec{v}_R(t+\Delta t)$ and $\vec{\omega}_R(t+\Delta t)$ using Equation 3.11a, we only have to iterate twice over all particles.

5.1. Validation

We use a rigid body placed onto an inclined plane as a basic test for the precision of our friction formulation. Given the slope of the plane θ, it is known that the rigid body R should slide down the ramp if $\mu_R < \tan\theta$ with coefficient of friction μ_R. In our test scenario, the plane has a slope of $\theta = 30°$, we start with $\mu_R = 1$ and reduce μ_R with constant speed over time. Figure 3 visualizes the scenario and gives a first impression of the test results. As we can see, the friction model proposed by Gissler et al. [GPB*19] is not able to replicate stiction as the rigid body slides down the slope even for $\mu_R = 1$. On the other hand, our friction formulation correctly causes the body to stick to the surface as long as $\mu_R > \tan 30° \approx 0.577$. Figure 4 gives a more detailed insight into the relation between coefficient of friction and velocity of the rigid body simulated using our friction formulation.

5.2. Solver Robustness

To hint at the robustness of our particle based contact handling, we simulate the degenerated test cases proposed by K. Erleben in [Erl18]. These special-cases of contacts between bodies are challenging for traditional mesh-based contact handling schemes [FLS*21]. As shown in Figure 5, our simulation method does not show any difficulties when simulating these constellations. However, we still like to mention that contact normals computed in our simulation only approximately match the ones listed by K. Erleben in [Erl18] due to the particle sampling of surfaces. The small errors in the contact normal computation have negligible effect on the overall simulation results which is why we do not consider them any further, but instead refer to work published by Koschier and Bender [KB17] as well as Bender et al. [BKWK19] that addresses a very similar issue.

5.3. Solver Performance

Large mass ratios between interacting bodies are traditionally hard to handle. As shown in Figure 6, our method is able to stably simulate a rigid cube resting on top of a second cube, whereby the upper cube is one thousand times heavier. We use the same scenario to evaluate the performance of our NNCG solver, in comparison to a default Jacobi solver as used by Gissler et al. [GPB*19]. For that, we simulate one second of the blocks dropping and resting on each

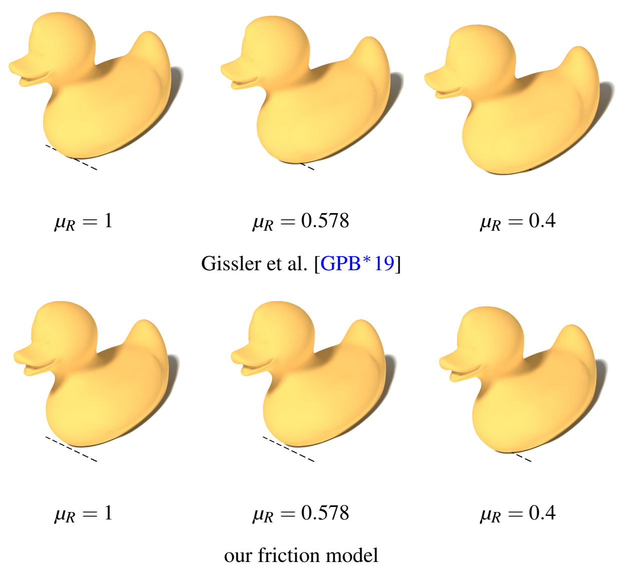


Figure 3: A rigid duck is placed onto an inclined surface. The friction coefficient μ_R of the rigid duck is reduced over time until the duck starts sliding down the slope. As we can see, the friction model by Gissler et al. [GPB*19] is not able to replicate stiction, the duck moves no matter how high μ_R is set. Our friction model on the other hand is able to simulate stiction and correctly reproduces the stick-slip transition around $\mu_R \approx 0.577$.

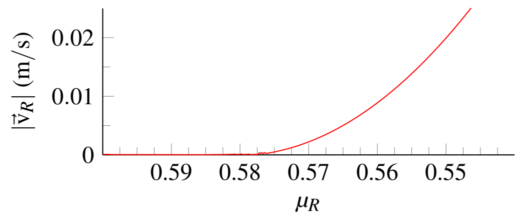


Figure 4: The relation between the coefficient of friction μ_R of rigid body R and the magnitude of its translational velocity $\vec{v}_R$ using our friction model. We can see that R starts accelerating as soon as $\mu_R < 0.577$ which is in good agreement with the analytical solution. Note that the velocity increases in a quadratic manner as μ_R decreases.

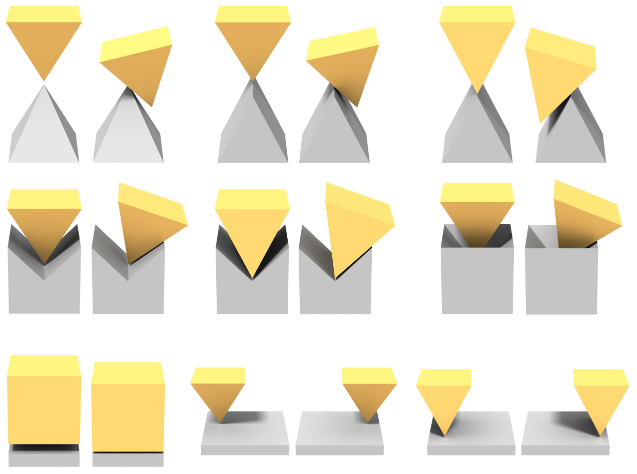


Figure 5: Degenerated test cases proposed by K. Erleben [Erl18] which can be challenging for mesh-based contact handling methods. For each of the nine constellation, we show the initial setting and right next to it a snapshot of the simulation. The bodies are given some initial velocity as described in [Erl18]. Visually, our particle-based contact handling shows no problems whatsoever when simulating the shown constellations.

other. In both cases we set $\Delta t = 0.001\,\mathrm{s}$, $\upsilon = 0.5$ with particle distance $h = 0.02\,\mathrm{m}$ and the same desired residual. As we can see in the top diagram in Figure 7, the default Jacobi solver on average requires approximately five times more iterations compared to the NNCG solver. This speedup by far overcompensates for the slightly higher computational cost per NNCG iteration, resulting in a significant performance advantage for the NNCG solver in this scenario. However, we note that this scenario represents a special case of a very challenging body constellation, which is why we additionally evaluate the solver performance at a more common and better behaved body arrangement. We simulate a stack of eight rigid boxes with equal masses using the same parameter setting as before, and again compare the solver iterations of the NNCG solver against the iterations required by the Jacobi solver. The result is visualized in the center of Figure 7. As we can see, the difference in the number of solver iterations is a little less severe compared to the previous scenario and in some simulation steps the NNCG solver takes almost as many iterations as the Jacobi solver. However, conforming to the previous scenario, on average, the Jacobi solver still requires more than three times as many iterations as the NNCG solver when simulating the stacked boxes. To evaluate the solver performance in a coupled fluid-rigid simulation, we simulate a fluid pillar with a rigid duck swimming on the surface. State-of-the-art particle fluid solvers such as PCISPH [SP09], PBF [MM13], IISPH [ICS*14] and DFSPH [BK15] use a Jacobi-style solving method comparable to the one presented by Gissler et al. [GPB*19]. We evaluate our implementation in relation to a standard Jacobi solver similar to Gissler et al. [GPB*19] in order to classify the performance of the NNCG solver in the context of fluid and coupled fluid-rigid simulations. The required solver iterations of the Jacobi and NNCG solver are displayed at the bottom of Figure 7. The difference in number of solver iterations in this scenario is especially significant as the Jacobi solver requires more than ten times the iterations the NNCG solver needs to reach an equal residual.

Motivated by the presented performance advantages, as a last test scenario we evaluate how the Jacobi and NNCG solver scale for increasing scenario complexity. A commonly used factor to determine scenario complexity is depth of a fluid body under gravity [KBST19]. We simulate a fluid pillar that grows in height during the simulation and measure the required solver iterations over time. The results are displayed in Figure 8. As we can see, the Jacobi iterations increase approximately linearly with increasing fluid depth, while the number of NNCG iterations show sublinear growth. Thus, in this specific scenario, for rising fluid depth, the significance of the performance advantage of the NNCG solver increases. In summary, in all tested scenarios, the NNCG solver maintains a considerable performance advantage over the Jacobi solver.

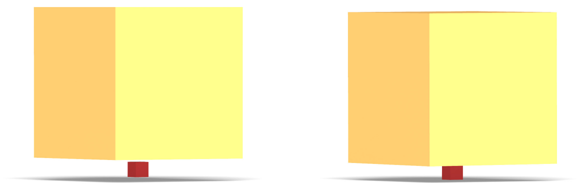


Figure 6: A heavy rigid block (yellow) is dropped on a much lighter block (red) where it then comes to rest. The mass ratio between the blocks is 1000 : 1. Our contact handling method is able to robustly handle this challenging constellation.

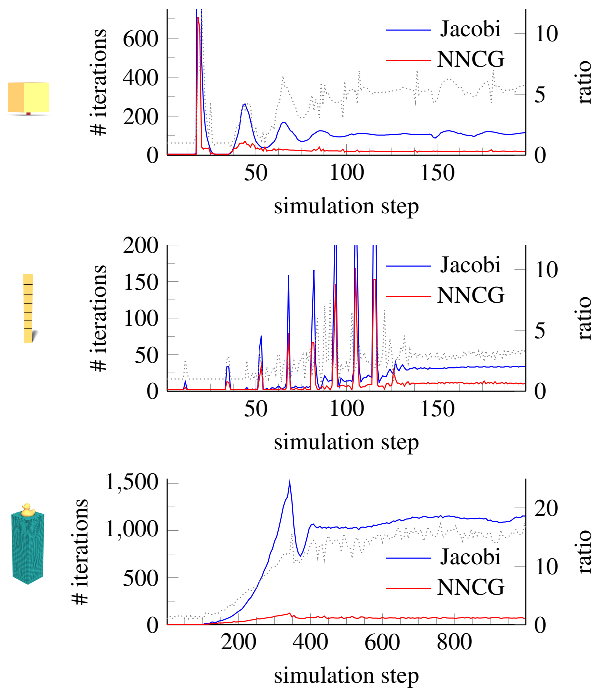


Figure 7: The required solver iterations of the default Jacobi solver (blue) as well as the NNCG solver (red), to simulate a heavy block falling on a lighter block (top), to simulate a stack of eight boxes (center) and to simulate a fluid pillar (bottom). The ratios between both iteration counts are indicated by the gray dotted line. In the upper two scenarios, we can clearly see the first contacts between the bodies as a spike in solver iterations at the start of the simulation. In the example with the high body mass ratio, at the initial contact, the Jacobi solver requires approximately ten times more iterations compared to the NNCG solver to achieve the same residual error. After the bodies came to rest, the Jacobi solver on average still requires five times as many iterations. To simulate the stacked boxes, the Jacobi solver requires a little less than four times as many solver iterations, and in the fluid pillar example the Jacobi solver even requires over ten times more iterations. In all three test scenarios our NNCG solver has a significant performance advantage over a standard Jacobi method.

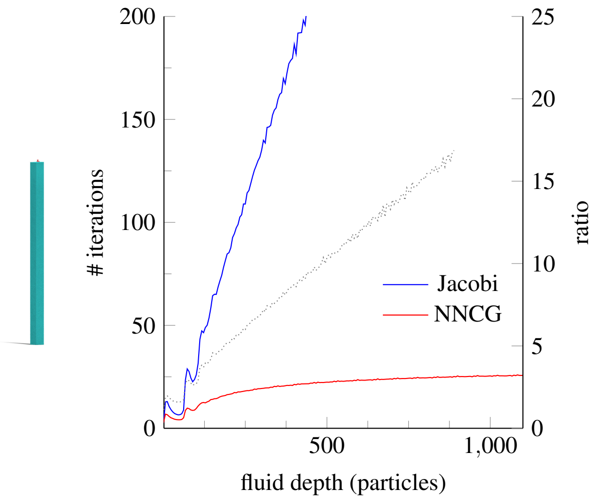


Figure 8: The required solver iterations of the default Jacobi solver (blue) and the NNCG solver (red) in relation to an estimate of the fluid pillar depth in particles. The ratio between the Jacobi iterations and NNCG iterations is indicated as a gray dotted line. Over time, the fluid pillar grows in height and as such the number of required solver iterations increase. We can see that for the Jacobi solver the required solver iterations increase approximately linearly with fluid depth, while the NNCG iterations only show sublinear growth. Thus, in the shown scenario, for high fluid depth the performance advantage of the NNCG solver becomes increasingly significant.

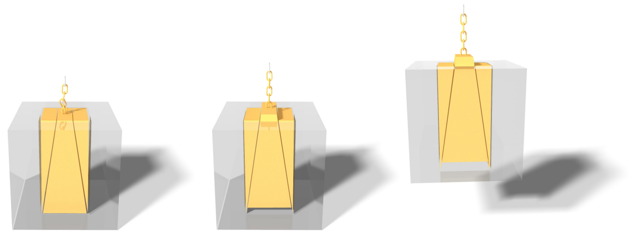


Figure 9: A Lewis lifting mechanism. When pulled upwards, the inner pyramid shaped piece presses the wedges against the outer weight. Solely due to frictional forces, the outer weight is lifted.

5.4. Versatility and Scalability

We demonstrate the capabilities of our simulation method by showcasing a number of simulation scenarios we deemed to be challenging to the friction and pressure solver due to many interacting rigid bodies, complex interactions between fluids and rigid geometry as well as scenarios requiring precisely computed friction forces. For each scenario, average timesteps, required solver iterations and computation times are listed in Table 1.

Lewis Lifting Mechanism: Inspired by Ferguson et al. [FLS*21], we simulate a Lewis lift which is a mechanism that can lift weights relying on frictional forces. As Figure 9 shows, our friction solver is able to handle static friction in a correct and precise manner without revealing any visible artifacts.

Modified Card House: Simulating houses of cards is a popular way to demonstrate correctness and precision of frictional forces [KSJP08, BET14, LFS*20, FLS*21]. To show that our fric-

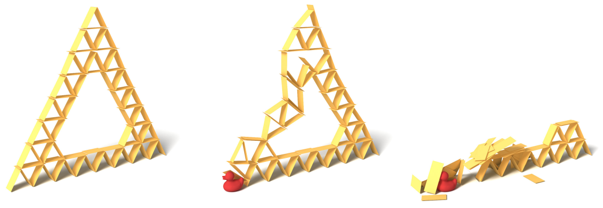


Figure 10: A modified card house is simulated using our contact handling method. Frictional forces enable the structure to stand upright by holding all cards in place. All cards are given the same mass and are actively simulated without being put to sleep. The card house collapses after a rigid duck (red) crashes into it.

tion handling matches the ability of state-of-the-art friction computation methods, we simulate a seven-story card house. To further increase difficulty, we added a hole in the middle of the card house making the structure even more fragile. Figure 10 shows the card house and its destruction.

Breaking Dam: A dam consisting of stacked rigid blocks is holding back a body of water whose depth is continuously rising. Here, the strong coupling between fluid pressure forces and dry friction is especially important to ensure a stable and robust simulation at the interface between fluid and dam. As soon as pressure forces caused by the fluid are growing larger than the static friction forces between the blocks can compensate for, the dam starts to break. As illustrated in Figure 11, the released fluid causes a flood and parts of the dam knock over a bridge standing downstream. During the simulation, the fluid is sampled by a maximum of 2.5 million particles, the dam and bridge are build from 128 rigid blocks sampled by 600 thousand particles and the kinematic boundary is sampled with 1.2 million particles.

Tower: Simulating high stacks of rigid bodies is inherently difficult as contact information needs to be propagated through the stack to ensure incompressibility of all rigid bodies and correct friction handling [Erl07]. Our solver is able to stably simulate an over 40-story high tower standing upright. By shooting two rigid bodies into the tower we let it collapse into a basin filled with fluid, showcasing the robustness of the monolithic simulation method. Figure 12 gives an overview of the simulation using 3.6 million fluid particles, 1.4 million rigid particles sampling 658 rigid bodies and 4.9 million particles belonging to kinematic boundaries.

Bulk Simulation: We showcase a bulk simulation to hint at the wide range of possible applications of our simulation method. As shown in Figure 13, the bulk consists of thousands of rigid bodies in the shape of armadillos. Each armadillo is sampled with 622 rigid particles, giving a good approximation of the complex underlying geometry. To demonstrate the robustness of our solver when confronted with complex interacting geometry under difficult conditions, the armadillos are caught in a net consisting of actively simulated rigid chain links. Over time, a pile of armadillos is formed which is held in place by static friction forces. In total, there are 8700 armadillos and 670 chain links sampled with 5.7 million rigid particles. The kinematic geometry is sampled using 5.0 million particles.

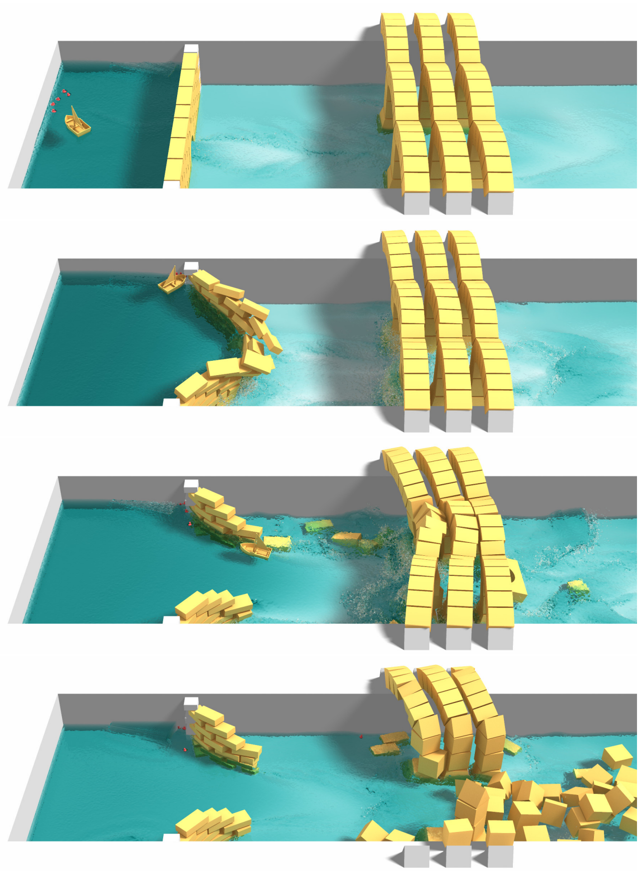


Figure 11: The outlet of a dam can not release fluid volume fast enough, so the water level behind the dam keeps rising. At some point, the pressure forces exerted by the fluid body onto the dam become larger than the static friction forces that are keeping the dam in place. The dam breaks, causing the destruction of a bridge built downstream. The lighter parts of the bridge are carried away by the flow. It is easy to see that our proposed strong coupling between fluid pressure and rigid friction forces helps to guarantee a stable simulation of the water-dam interface.

Chain: Simulating long chains of rigid bodies is inherently difficult due to mutual dependencies of individual chain links over long distances. Contacts need to be resolved very precisely to prevent elongations and elastic behavior of the chain. Illustrated in Figure 14, we demonstrate that our solver is able to simulate a one hundred rigid links long chain, with no need for an extra long-range constraint stabilization technique such as proposed in [MCMJ17]. Additionally, to further increase difficulty, some chain links are shaped as hollow letters containing fluid and small rigid ducks. The chain is rolled up on an axle causing challenging simulation circumstances with high pressure and friction forces between contacting chain links, diverse sizes and geometries of links and rapid velocity changes in the chain including the contained fluid. In total, the chain consists of over 100 rigid bodies, sampled by 600 thousand particles. The fluids are represented using 270 thousand particles.

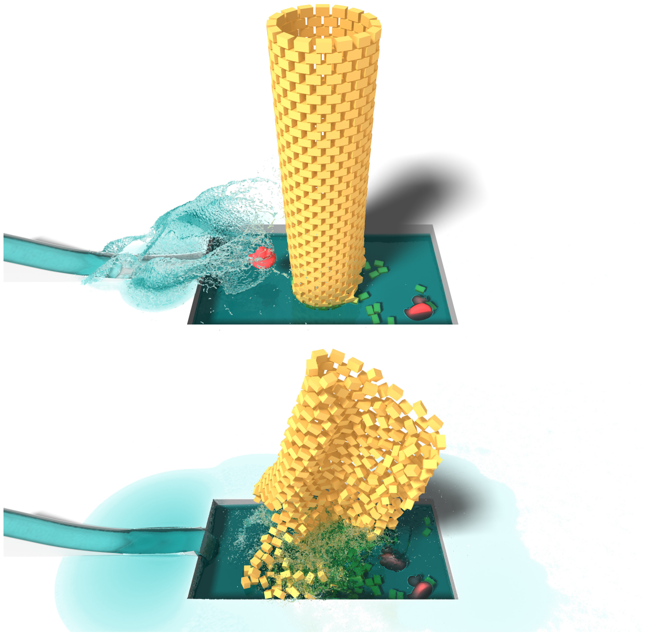
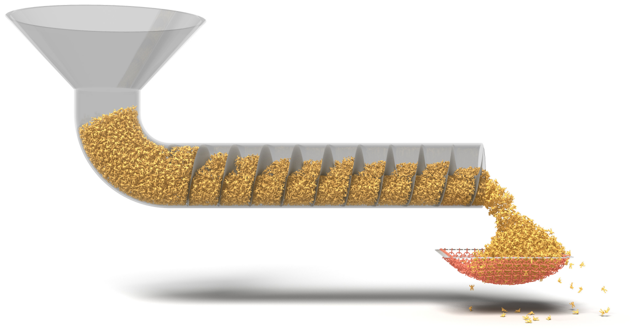


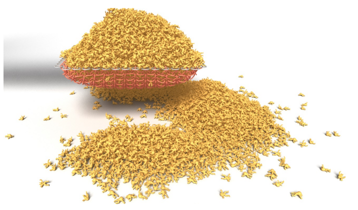
Figure 12: Over 600 rigid blocks are used to build a tower. Frictional forces ensure that the tower is able to stand upright, even with fluid pushing against the base of the tower. Using a waterslide, two rigid ducks crash into the tower causing it to collapse.

	Δt	solver iter.	computation time per	
			step	simulated s
Lewis Lift	2.0 ms	15.9	1.38 s	689 s
Card House	0.5 ms	20.0	0.35 s	691 s
Dam	1.0 ms	20.0	5.44 s	5436 s
Tower	1.9 ms	10.0	3.73 s	2013 s
Bulk	0.3 ms	4.0	4.59 s	18 547 s
Chain	0.2 ms	20.0	1.52 s	6826 s

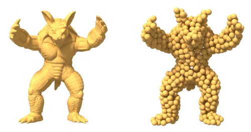
Table 1: Averaged computational costs of the named simulation scenarios. From left to right we list the average timestep Δt, required solver iterations, computation time required per simulation step and computation time required per simulated second. All computations are performed on a 24-core 2.7 GHz Intel Xeon E5-2697 v2 workstation. Note that for all listed scenarios, except for the Lewis Lift, we chose a relatively high number of minimum solver iterations such that the convergence criterion is typically fulfilled before the minimum number of solver iterations is reached. This has the effect that solver iterations are constant most of the time during the simulation, which we found to be beneficial for simulating fragile structures.



Nine thousand rigid armadillos are pushed through a pipe using a spiral feeder. Armadillos falling out of the pipe are caught in a net consisting of 670 actively simulated rigid chain links.



The armadillos collected in the net are able to form a stable pile due to static friction forces.

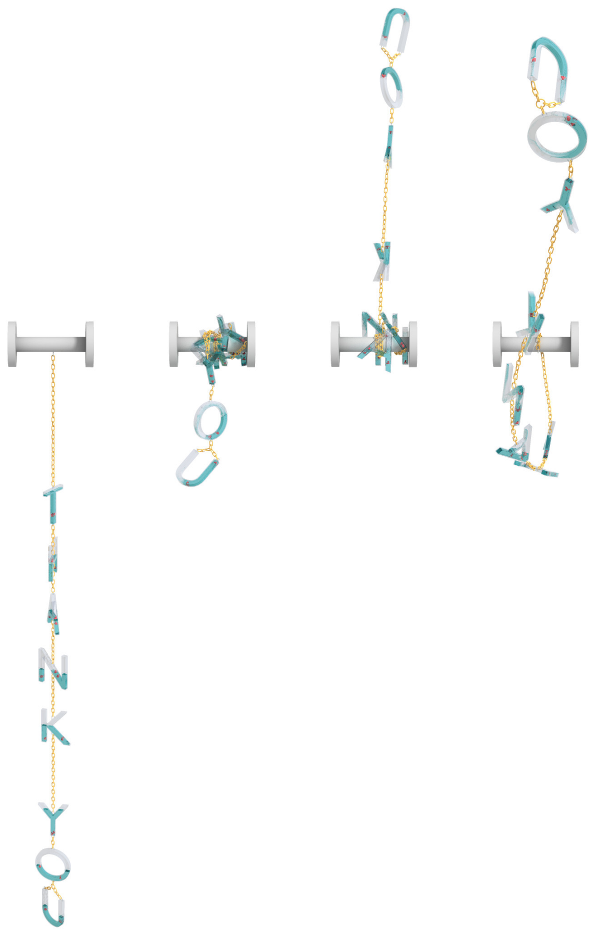


Each armadillo is sampled with 622 rigid particles.

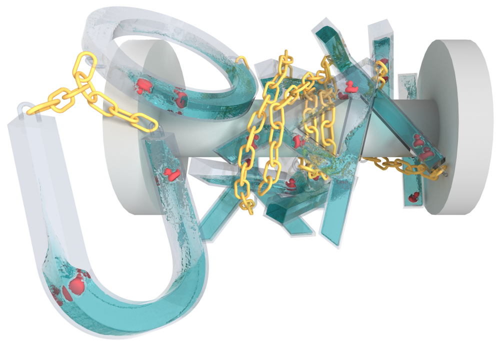
Figure 13: A bulk material simulation using our rigid body solver. Each grain is simulated as a rigid body in the shape of an Armadillo sampled with particles. The concave and complex Armadillo geometry causes the bodies to get caught up in the net. Stable piles of Armadillos are formed in the net and on the surrounding ground surface.

6. Limitations

Since the number of particles used to sample rigid body geometry mostly depends on body and particle size instead of complexity of the geometry, simple shapes with large flat surfaces still might require thousands of sample particles. In this situation, a pure rigid body simulation ignoring rigid-fluid interaction is likely to be more efficient when using a mesh-based rigid body representation. Our simulation method uses the same size for rigid body particles, fluid particles and boundary particles. While this allows a straightforward interaction between particles, the resolution of the contact handling between rigid bodies is directly coupled to the fluid resolutions. In the future, approaches that implement different particle sizes such as described in [WHK17, WK21] could be investigated to enable a more flexible simulation resolution. Due to the velocity-based friction formulation drift might occur between contacting rigid bodies even if static friction forces are applied. This drift can be reduced at the cost of more solver iterations, but it cannot be eliminated entirely. In the future, it might be interesting to additionally implement a stabilization scheme that guarantees static contacts such as described by Xu et al. [XZB14] or O. Ding & C. Schroeder [DS20]. In our simulation, we employed a simple Coulomb friction model. However, we believe that in the future, the described solver could be easily extended to handle more sophisti-



The chain is rolled up. While the axle continues spinning, the chain is able to untangle itself until its momentum does not suffice anymore to keep it stretched out while rotating.



A closeup view on the entangled chain.

Figure 14: A simulation of a chain. Each of the over 100 chain links is formed by an individual rigid body. The hollow letters are filled with fluid and small rigid ducks (red). Our simulation method is able to robustly handle the challenging mixture of large pressure and friction forces propagated over long distances and complex interactions between fluid and rigid bodies, while high precision solver results are indispensable.

cated friction models such as anisotropic friction [Erl17,EMAK19] or torque due to dry friction resisting spinning [BNT*15].

7. Conclusion

We have introduced a monolithic SPH solver for particle-based fluids and rigid bodies including dry frictional forces. Our simulation method calculates predicted volume errors as a simple and robust way to detect collisions between rigid bodies as well as compressed fluid particles. Volume errors are resolved by computing pressure and pressure forces, a concept well known from existing particle-based pressure solvers [SP09,ICS*14,BK15,GPB*19]. Strong coupling between pressure and friction forces is achieved by directly incorporating the effects of pressure and fictional forces into the computation of predicted velocities, which in turn influence the current pressure and friction computation. A friction mirroring step is build into the pressure and friction system to guarantee conservation of momentum. Building on top of the relaxed-Jacobi solver, a nonlinear nonsmooth conjugate gradient method is employed to accelerate the solving procedure. Depending on the scenario, this can result in a significant reduction of solver iterations. Further, we have shown that our implicit friction formulation can handle static friction, produces correct behavior at stick-slip transitions and scales to large numbers of simultaneous rigid body contacts. The strong coupling between pressure and friction allows us to stably simulate complex structures of rigid bodies interacting with fluids, opening the door for a whole range of new interesting simulation scenarios.

Acknowledgements

The rubber duck model by 3DWP is licensed under CC BY-NC-SA 4.0. The boat model by devonjones is licensed under CC BY-SA 3.0. The Armadillo model is courtesy of the Stanford University Computer Graphics Laboratory. We thank FIFTY2 Technology for providing an academic license of PreonLab [FIF22], which was used to generate all shown images of simulations.

References

[AAT13] AKINCI N., AKINCI G., TESCHNER M.: Versatile surface tension and adhesion for SPH fluids. *ACM Trans. Graph. 32*, 6 (nov 2013). doi:10.1145/2508363.2508395. 7

[AE21] ANDREWS S., ERLEBEN K.: Contact and friction simulation for computer graphics. In *ACM SIGGRAPH 2021 Courses* (New York, NY, USA, 2021), SIGGRAPH '21, Association for Computing Machinery. doi:10.1145/3450508.3464571. 2, 4, 6, 7, 10

[AFC*10] ALLARD J., FAURE F., COURTECUISSE H., FALIPOU F., DURIEZ C., KRY P. G.: Volume contact constraints at arbitrary resolution. *ACM Trans. Graph. 29*, 4 (July 2010). doi:10.1145/1778765.1778819. 5

[AIA*12] AKINCI N., IHMSEN M., AKINCI G., SOLENTHALER B., TESCHNER M.: Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans. Graph. 31*, 4 (jul 2012). doi:10.1145/2185520.2185558. 1, 3, 4, 6, 21

[ANE17] ANDERSEN M., NIEBE S., ERLEBEN K.: A fast linear complementarity problem solver for fluid animation using high level algebra interfaces for GPU libraries. *Computers & Graphics 69* (2017), 36–48. doi:10.1016/j.cag.2017.09.006. 6

[ANEK21] ANDREWS S., NASSIF L., ERLEBEN K., KRY P. G.: Coupling friction with visual appearance. *Proc. ACM Comput. Graph. Interact. Tech. 4*, 3 (sep 2021). doi:10.1145/3480138. 2

[ANZS18] AKBAY M., NOBLES N., ZORDAN V., SHINAR T.: An extended partitioned method for conservative solid-fluid coupling. *ACM Trans. Graph. 37*, 4 (jul 2018). doi:10.1145/3197517.3201345. 3

[AO11] ALDUÁN I., OTADUY M. A.: SPH granular flow with friction and cohesion. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, Association for Computing Machinery, p. 25–32. doi:10.1145/2019406.2019410. 2

[AP97] ANITESCU M., POTRA F. A.: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics 14* (1997), 231–247. doi:10.1023/A:1008292328909. 2

[Bar89] BARAFF D.: Analytical methods for dynamic simulation of non-penetrating rigid bodies. *SIGGRAPH Comput. Graph. 23*, 3 (July 1989), 223–232. doi:10.1145/74334.74356. 2

[Bar90] BARAFF D.: Curved surfaces and coherence for non-penetrating rigid body simulation. *SIGGRAPH Comput. Graph. 24*, 4 (sep 1990), 19–28. doi:10.1145/97880.97881. 2

[Bar91] BARAFF D.: Coping with friction for non-penetrating rigid body simulation. *SIGGRAPH Comput. Graph. 25*, 4 (jul 1991), 31–41. doi:10.1145/127719.122722. 2

[Bar93a] BARAFF D.: Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica 10*, 2–4 (oct 1993), 292–352. doi:10.1007/BF01891843. 2

[Bar93b] BARAFF D.: Non-penetrating rigid body simulation. In *Proceedings of Eurographics '93 State of the Art Reports* (sep 1993). 2

[Bar94] BARAFF D.: Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), SIGGRAPH '94, Association for Computing Machinery, p. 23–34. doi:10.1145/192161.192168. 2

[Bar95] BARAFF D.: Interactive simulation of solid rigid bodies. *IEEE Comput. Graph. Appl. 15*, 3 (may 1995), 63–75. doi:10.1109/38.376615. 2

[BBB07] BATTY C., BERTAILS F., BRIDSON R.: A fast variational framework for accurate solid-fluid coupling. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, Association for Computing Machinery, p. 100–es. doi:10.1145/1275808.1276502. 3

[BDCDA11] BERTAILS-DESCOUBES F., CADOUX F., DAVIET G., ACARY V.: A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies. *ACM Trans. Graph. 30*, 1 (feb 2011). doi:10.1145/1899404.1899410. 2

[Ben07] BENDER J.: *Impulsbasierte Dynamiksimulation von Mehrkörpersystemen in der virtuellen Realität*. PhD thesis, Universität Fridericiana zu Karlsruhe, 01 2007. doi:10.5445/KSP/1000006079. 6

[BET14] BENDER J., ERLEBEN K., TRINKLE J.: Interactive simulation of rigid body dynamics in computer graphics. *Comput. Graph. Forum 33*, 1 (Feb. 2014), 246–270. doi:10.1111/cgf.12272. 1, 2, 3, 4, 5, 6, 8, 13

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3 (July 2002), 594–603. doi:10.1145/566654.566623. 4

[BGI*18] BAND S., GISSLER C., IHMSEN M., CORNELIS J., PEER A., TESCHNER M.: Pressure boundaries for implicit incompressible SPH. *ACM Trans. Graph. 37*, 2 (feb 2018). doi:10.1145/3180486. 6

[BGPT18] BAND S., GISSLER C., PEER A., TESCHNER M.: MLS pressure boundaries for divergence-free and viscous SPH fluids. *Computers & Graphics 76* (2018), 37–46. doi:10.1016/j.cag.2018.08.001. 6

[BGT19] BAND S., GISSLER C., TESCHNER M.: Compressed neighbour lists for SPH. *Computer Graphics Forum 39* (11 2019). doi:10.1111/cgf.13890. 4

[BK15] BENDER J., KOSCHIER D.: Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2015), SCA '15, Association for Computing Machinery, p. 147–155. doi:10.1145/2786784.2786796. 3, 4, 6, 10, 12, 16

[BKWK19] BENDER J., KUGELSTADT T., WEILER M., KOSCHIER D.: Volume maps: An implicit boundary representation for SPH. In *Motion, Interaction and Games* (New York, NY, USA, 2019), MIG '19, Association for Computing Machinery. doi:10.1145/3359566.3360077. 1, 3, 11

[BNT*15] BOUCHARD C., NESME M., TOURNIER M., WANG B., FAURE F., KRY P. G.: 6d frictional contact for rigid bodies. In *Proceedings of the 41st Graphics Interface Conference* (CAN, 2015), GI '15, Canadian Information Processing Society, p. 105–114. 16

[Bri15] BRIDSON R.: *Fluid simulation for computer graphics*. CRC press, 2015. 1

[Bro99] BROGLIATO B.: *Nonsmooth mechanics*. Springer, New York, 1999. 2

[BT07] BECKER M., TESCHNER M.: Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2007), SCA '07, Eurographics Association, p. 209–217. 3

[BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct forcing for lagrangian rigid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics 15*, 3 (2009), 493–503. doi:10.1109/TVCG.2008.107. 3

[BYM05] BELL N., YU Y., MUCHA P. J.: Particle-based simulation of granular materials. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), SCA '05, Association for Computing Machinery, p. 77–86. doi:10.1145/1073368.1073379. 1, 2

[CA09] COURTECUISSE H., ALLARD J.: Parallel dense gauss-seidel algorithm on many-core processors. In *2009 11th IEEE International Conference on High Performance Computing and Communications* (2009), pp. 139–147. doi:10.1109/HPCC.2009.51. 2

[CLL*22] CHEN Y., LI M., LAN L., SU H., YANG Y., JIANG C.: A unified newton barrier method for multibody dynamics. In *ACM SIGGRAPH 2022 Papers* (2022), SIGGRAPH '22, Association for Computing Machinery. 2

[CM10] CHENTANEZ N., MÜLLER M.: Real-time simulation of large bodies of water with small scale details. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2010), SCA '10, Eurographics Association, p. 197–206. 3

[CM11] CHENTANEZ N., MÜLLER M.: A multigrid fluid pressure solver handling separating solid boundary conditions. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, Association for Computing Machinery, p. 83–90. doi:10.1145/2019406.2019418. 2

[CMT04] CARLSON M., MUCHA P. J., TURK G.: Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Trans. Graph. 23*, 3 (aug 2004), 377–384. doi:10.1145/1015706.1015733. 3

[Coe17] COETZEE C.: Review: Calibration of the discrete element method. *Powder Technology 310* (2017), 104–142. doi:10.1016/j.powtec.2017.01.015. 1

[Cou22] COUMANS E.: Bullet, 2022. URL: https://pybullet.org/wordpress/. 3

[CPS09] COTTLE R. W., PANG J.-S., STONE R. E.: *The linear complementarity problem*. SIAM, 2009. doi:10.1137/1.9780898719000. 9, 10

[CS79] CUNDALL P. A., STRACK O. D. L.: A discrete numerical model for granular assemblies. *Géotechnique 29*, 1 (1979), 47–65. doi:10.1680/geot.1979.29.1.47. 1

[DBDB11] DAVIET G., BERTAILS-DESCOUBES F., BOISSIEUX L.: A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 1–12. doi:10.1145/2070781.2024173. 2

[DCB14] DEUL C., CHARRIER P., BENDER J.: Position-based rigid body dynamics. *Computer Animation and Virtual Worlds 27*, 2 (2014), 103–112. doi:10.1002/cav.1614. 3

[DS20] DING O., SCHROEDER C.: Penalty force for coupling materials with coulomb friction. *IEEE Transactions on Visualization and Computer Graphics 26*, 7 (2020), 2443–2455. doi:10.1109/TVCG.2019.2891591. 15

[Ebe10] EBERLY D. H.: Chapter 14 - linear complementarity and mathematical programming. In *Game Physics (Second Edition)*, Eberly D. H., (Ed.), second edition ed. Morgan Kaufmann, Boston, 2010, pp. 813–856. doi:10.1016/B978-0-12-374903-1.00014-1. 2

[EMAK19] ERLEBEN K., MACKLIN M., ANDREWS S., KRY P.: The matchstick model for anisotropic friction cones. *Computer Graphics Forum 39* (11 2019). doi:10.1111/cgf.13885. 3, 16

[Erl07] ERLEBEN K.: Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph. 26*, 2 (June 2007), 12–es. doi:10.1145/1243980.1243986. 6, 14

[Erl13] ERLEBEN K.: Numerical methods for linear complementarity problems in physics-based animation. In *ACM SIGGRAPH 2013 Courses* (New York, NY, USA, 2013), SIGGRAPH '13, Association for Computing Machinery. doi:10.1145/2504435.2504443. 2, 6, 10

[Erl17] ERLEBEN K.: Rigid body contact problems using proximal operators. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2017), SCA '17, Association for Computing Machinery. doi:10.1145/3099564.3099575. 2, 3, 6, 7, 8, 9, 10, 16

[Erl18] ERLEBEN K.: Methodology for assessing mesh-based contact point methods. *ACM Trans. Graph. 37*, 3 (jul 2018). doi:10.1145/3096239. 4, 5, 8, 11, 12

[EWC*10] ESSA I., WOJTAN C., CARLSON M., KWATRA N., MUCHA P. J., TURK G.: Fluid simulation with articulated bodies. *IEEE Transactions on Visualization & Computer Graphics 16*, 01 (jan 2010), 70–80. doi:10.1109/TVCG.2009.66. 3

[FGNU06] FOERG M., GEIER T., NEUMANN L., ULBRICH H.: r-factor strategies for the augmented lagrangian approach in multi-body contact mechanics. In *III European Conference on Computational Mechanics* (Dordrecht, 2006), Motasoares C. A., Martins J. A. C., Rodrigues H. C., Ambrósio J. A. C., Pina C. A. B., Motasoares C. M., Pereira E. B. R., Folgado J., (Eds.), Springer Netherlands, pp. 316–316. 9

[FIF22] FIFTY2 TECHNOLOGY: PreonLab, 2022. URL: www.fifty2.eu. 16

[FLS*21] FERGUSON Z., LI M., SCHNEIDER T., GIL-URETA F., LANGLOIS T., JIANG C., ZORIN D., KAUFMAN D. M., PANOZZO D.: Intersection-free rigid body dynamics. *ACM Trans. Graph. 40*, 4 (July 2021). doi:10.1145/3450626.3459802. 2, 8, 11, 13

[GHZ*20] GEILINGER M., HAHN D., ZEHNDER J., BÄCHER M., THOMASZEWSKI B., COROS S.: ADD: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Trans. Graph. 39*, 6 (Nov. 2020). doi:10.1145/3414685.3417766. 2

[GNKT16] GHOLAMI F., NASRI M., KÖVECSES J., TEICHMANN M.: A Fast Algorithm for Contact Dynamics of Multibody Systems Using the Box Friction Model. *Journal of Computational and Nonlinear Dynamics 12*, 1 (09 2016). 011016. doi:10.1115/1.4034396. 2

[GPB*19] GISSLER C., PEER A., BAND S., BENDER J., TESCHNER M.: Interlinked SPH pressure solvers for strong fluid-rigid coupling. *ACM Trans. Graph. 38*, 1 (Jan. 2019). doi:10.1145/3284980. 1, 2, 3, 4, 5, 7, 9, 10, 11, 12, 16, 21

[GSLF05] GUENDELMAN E., SELLE A., LOSASSO F., FEDKIW R.: Coupling water and smoke to thin deformable and rigid shells. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, Association for Computing Machinery, p. 973–981. doi:10.1145/1186822.1073299. 3

[GZO10] GASCÓN J., ZURDO J. S., OTADUY M. A.: Constraint-based simulation of adhesive contact. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2010), SCA '10, Eurographics Association, p. 39–44. 2

[Hah88] HAHN J. K.: Realistic animation of rigid bodies. *SIGGRAPH Comput. Graph. 22*, 4 (June 1988), 299–308. doi:10.1145/378456.378530. 2

[HBH*18] HE Y., BAYLY A. E., HASSANPOUR A., MULLER F., WU K., YANG D.: A GPU-based coupled SPH-DEM method for particle-fluid flow with free surfaces. *Powder Technology 338* (2018), 548–562. doi:10.1016/j.powtec.2018.07.043. 1

[HFG*18] HU Y., FANG Y., GE Z., QU Z., ZHU Y., PRADHANA A., JIANG C.: A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph. 37*, 4 (jul 2018). doi:10.1145/3197517.3201293. 1, 3

[IABT11] IHMSEN M., AKINCI N., BECKER M., TESCHNER M.: A parallel SPH implementation on multi-core CPUs. *Comput. Graph. Forum 30* (03 2011), 99–112. doi:10.1111/j.1467-8659.2010.01832.x. 4

[ICS*14] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 426–435. doi:10.1109/TVCG.2013.105. 4, 6, 10, 12, 16

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH fluids in computer graphics. In *Eurographics 2014 - State of the Art Reports* (2014), Lefebvre S., Spagnuolo M., (Eds.), The Eurographics Association. doi:10.2312/egst.20141034. 1, 3, 4, 10

[JAJ98] JOURDAN F., ALART P., JEAN M.: A gauss-seidel like algorithm to solve frictional contact problems. *Computer Methods in Applied Mechanics and Engineering 155*, 1 (1998), 31–47. doi:10.1016/S0045-7825(97)00137-0. 3, 8

[JM92] JEAN M., MOREAU J. J.: Unilaterality and dry friction in the dynamics of rigid body collections. In *1st Contact Mechanics International Symposium* (1992), pp. 31–48. URL: https://hal.archives-ouvertes.fr/hal-01863710. 3

[KB17] KOSCHIER D., BENDER J.: Density maps for improved SPH boundary handling. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2017), SCA '17, Association for Computing Machinery. doi:10.1145/3099564.3099565. 1, 3, 8, 11

[KBST19] KOSCHIER D., BENDER J., SOLENTHALER B., TESCHNER M.: Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. *Eurographics 2019 - Tutorials* (2019). doi:10.2312/EGT.20191035. 3, 12

[KBST22] KOSCHIER D., BENDER J., SOLENTHALER B., TESCHNER M.: A Survey on SPH Methods in Computer Graphics. *Computer Graphics Forum 41*, 2 (2022). doi:10.1111/cgf.14508. 3

[KSJP08] KAUFMAN D. M., SUEDA S., JAMES D. L., PAI D. K.: Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph. 27*, 5 (Dec. 2008). doi:10.1145/1409060.1409117. 2, 3, 4, 8, 13

[KTS*14] KAUFMAN D. M., TAMSTORF R., SMITH B., AUBRY J.-M., GRINSPUN E.: Adaptive nonlinearity for collisions in complex rod assemblies. *ACM Trans. Graph. 33*, 4 (jul 2014). doi:10.1145/2601097.2601100. 2

[LDN*18] LI J., DAVIET G., NARAIN R., BERTAILS-DESCOUBES F., OVERBY M., BROWN G. E., BOISSIEUX L.: An implicit frictional contact solver for adaptive cloth simulation. *ACM Trans. Graph. 37*, 4 (jul 2018). doi:10.1145/3197517.3201308. 2

[LDW*22] Li Y., Du T., Wu K., Xu J., Matusik W.: Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM Trans. Graph. 42*, 1 (oct 2022). doi:10.1145/3527660. 2

[LFS*20] Li M., Ferguson Z., Schneider T., Langlois T., Zorin D., Panozzo D., Jiang C., Kaufman D. M.: Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph. 39*, 4 (July 2020). doi:10.1145/3386569.3392425. 2, 13

[LG03] Leine R., Glocker C.: A set-valued force law for spatial coulomb-contensou friction. *European Journal of Mechanics - A/Solids 22* (03 2003), 193–216. doi:10.1016/S0997-7538(03)00025-1. 3

[LJBBD20] Ly M., Jouve J., Boissieux L., Bertails-Descoubes F.: Projective dynamics with dry frictional contact. *ACM Trans. Graph. 39*, 4 (July 2020). doi:10.1145/3386569.3392396. 2

[LKL*22] Lan L., Kaufman D. M., Li M., Jiang C., Yang Y.: Affine body dynamics: Fast, stable intersection-free simulation of stiff materials. In *ACM SIGGRAPH 2022 Papers* (2022), SIGGRAPH '22, Association for Computing Machinery. 2

[MCMJ17] Müller M., Chentanez N., Macklin M., Jeschke S.: Long range constraints for rigid body simulations. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2017), SCA '17, Association for Computing Machinery. doi:10.1145/3099564.3099574. 14

[MEM*19] Macklin M., Erleben K., Müller M., Chentanez N., Jeschke S., Makoviychuk V.: Non-smooth newton methods for deformable multi-body dynamics. *ACM Trans. Graph. 38*, 5 (Oct. 2019). doi:10.1145/3338695. 2, 6, 8, 10

[MEM*20] Macklin M., Erleben K., Müller M., Chentanez N., Jeschke S., Kim T. Y.: Primal/dual descent methods for dynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2020), SCA '20, Eurographics Association. doi:10.1111/cgf.14104. 2, 8

[Mir96] Mirtich B. V.: *Impulse-Based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California, Berkeley, 1996. AAI9723116. 2

[MM13] Macklin M., Müller M.: Position based fluids. *ACM Trans. Graph. 32*, 4 (jul 2013). doi:10.1145/2461912.2461984. 12

[MMC16] Macklin M., Müller M., Chentanez N.: XPBD: Position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (New York, NY, USA, 2016), MIG '16, Association for Computing Machinery, p. 49–54. doi:10.1145/2994258.2994272. 3

[MMC*20] Müller M., Macklin M., Chentanez N., Jeschke S., Kim T.-Y.: Detailed rigid body simulation with extended position based dynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2020), SCA '20, Eurographics Association. doi:10.1111/cgf.14105. 3

[MMCK14] Macklin M., Müller M., Chentanez N., Kim T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July 2014). doi:10.1145/2601097.2601152. 1, 3

[Mon92] Monaghan J. J.: Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics 30*, 1 (1992), 543–574. doi:10.1146/annurev.aa.30.090192.002551. 7

[MST*04] Müller M., Schirm S., Teschner M., Heidelberger B., Gross M.: Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 159–171. doi:10.1002/cav.18. 3

[MW88] Moore M., Wilhelms J.: Collision detection and response for computer animation. *SIGGRAPH Comput. Graph. 22*, 4 (jun 1988), 289–298. doi:10.1145/378456.378528. 2, 3

[Ngu07] Nguyen H.: *GPU Gems 3*, first ed. Addison-Wesley Professional, 2007. 1

[NW06] Nocedal J., Wright S. J.: *Numerical Optimization*, second ed. Springer, New York, NY, USA, 2006. 10

[OTSG09] Otaduy M., Tamstorf R., Steinemann D., Gross M.: Implicit contact handling for deformable objects. *Comput. Graph. Forum 28* (04 2009), 559–568. doi:10.1111/j.1467-8659.2009.01396.x. 2, 4

[PAE10] Poulsen M., Abel S., Erleben K.: Heuristic convergence rate improvements of the projected gauss-seidel method for frictional contact problems. In *WSCG 2010* (2010), Skala V., (Ed.), Vaclav Skala - Union Agency, pp. 135–142. 18th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG 2010 ; Conference date: 01-02-2010 Through 04-02-2010. 10

[PAK*19] Peiret A., Andrews S., Kövecses J., Kry P. G., Teichmann M.: Schur complement-based substructuring of stiff multibody systems with contact. *ACM Trans. Graph. 38*, 5 (Oct. 2019). doi:10.1145/3355621. 2, 8

[PB14] Parikh N., Boyd S.: Proximal algorithms. *Foundations and Trends® in Optimization 1*, 3 (2014), 127–239. doi:10.1561/2400000003. 9

[PRM19] Pan Z., Ren B., Manocha D.: GPU-based contact-aware trajectory optimization using a smooth force model. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2019), SCA '19, Association for Computing Machinery. doi:10.1145/3309486.3340246. 2

[PRSV16] Pennestri E., Rossi V., Salvini P., Valentini P. P.: Review and comparison of dry friction force models. *Nonlinear Dynamics 83* (03 2016). doi:10.1007/s11071-015-2485-3. 2

[PZWZ21] Peng C., Zhan L., Wu W., Zhang B.: A fully resolved SPH-DEM method for heterogeneous suspensions with arbitrary particle shape. *Powder Technology 387* (2021), 509–526. doi:10.1016/j.powtec.2021.04.044. 1

[RHEW17] Reinhardt S., Huber M., Eberhardt B., Weiskopf D.: Fully asynchronous sph simulation. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (New York, NY, USA, 2017), SCA '17, Association for Computing Machinery. doi:10.1145/3099564.3099571. 3

[SBC*11] Solenthaler B., Bucher P., Chentanez N., Müller M., Gross M.: SPH based shallow water simulation. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2011)* (2011), Bender J., Erleben K., Galin E., (Eds.), The Eurographics Association. doi:10.2312/PE/vriphys/vriphys11/039-046. 3

[SHNE10] Silcowitz-Hansen M., Niebe S., Erleben K.: A nonsmooth nonlinear conjugate gradient method for interactive contact force problems. *Vis. Comput. 26*, 6–8 (June 2010), 893–901. doi:10.1007/s00371-010-0502-6. 3, 4, 10

[SNE09] Silcowitz M., Niebe S., Erleben K.: Nonsmooth newton method for fischer function reformulation of contact force problems for interactive rigid body simulation. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2009)* (2009), Prautzsch H., Schmitt A., Bender J., Teschner M., (Eds.), The Eurographics Association. doi:10.2312/PE/vriphys/vriphys09/105-114. 10

[SNT11] Schindler T., Nguyen B., Trinkle J.: Understanding the difference between prox and complementarity formulations for simulation of systems with contact. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011), pp. 1433–1438. doi:10.1109/IROS.2011.6094779. 8, 9

[SP08] Solenthaler B., Pajarola R.: Density contrast SPH interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2008), SCA '08, Eurographics Association, p. 211–218. 4

[SP09] Solenthaler B., Pajarola R.: Predictive-corrective incompressible SPH. In *ACM SIGGRAPH 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH '09, Association for Computing Machinery. doi:10.1145/1576246.1531346. 3, 4, 10, 12, 16

[ST96] STEWART D., TRINKLE J. C.: An implicit time-stepping scheme for rigid body dynamics with coulomb friction. *INTERNATIONAL JOURNAL OF NUMERICAL METHODS IN ENGINEERING 39* (1996), 2673–2691. 2

[Ste00] STEWART D.: Rigid-body dynamics with friction and impact. *SIAM Rev. 42* (03 2000), 3–39 (electronic). doi:10.1137/S0036144599360110. 2

[TBV12] TONGE R., BENEVOLENSKI F., VOROSHILOV A.: Mass splitting for jitter-free parallel rigid body simulation. *ACM Trans. Graph. 31*, 4 (July 2012). doi:10.1145/2185520.2185601. 2, 6, 9

[TIR06] THÜREY N., IGLBERGER K., RÜDE U.: Free surface flows with moving and deforming objects for LBM. In *Proceedings of Vision, Modeling and Visualization* (2006), vol. 2006, pp. 193–200. 3

[TL16] TAKAHASHI T., LIN M. C.: A multilevel SPH solver with unified solid boundary handling. *Computer Graphics Forum 35*, 7 (2016), 517–526. doi:10.1111/cgf.13048. 3

[TMFSG07] THUREY N., MULLER-FISCHER M., SCHIRM S., GROSS M.: Real-time breaking waves for shallow water simulations. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)* (2007), pp. 39–46. doi:10.1109/PG.2007.33. 3

[TSIHK06] TANAKA M., SAKAI M., ISHIKAWAJIMA-HARIMA, KOSHIZUKA S.: Rigid body simulation using a particle method. In *ACM SIGGRAPH 2006 Research Posters* (New York, NY, USA, 2006), SIGGRAPH '06, Association for Computing Machinery, p. 132–es. doi:10.1145/1179622.1179775. 1

[VHLL14] VINES M., HOUSTON B., LANG J., LEE W.-S.: Vortical inviscid flows with two-way solid-fluid coupling. *IEEE Transactions on Visualization and Computer Graphics 20*, 2 (2014), 303–315. doi:10.1109/TVCG.2013.95. 3

[WAK20] WINCHENBACH R., AKHUNOV R., KOLB A.: Semi-analytic boundary handling below particle resolution for smoothed particle hydrodynamics. *ACM Trans. Graph. 39*, 6 (nov 2020). doi:10.1145/3414685.3417829. 8

[WFS*21] WANG B., FERGUSON Z., SCHNEIDER T., JIANG X., ATTENE M., PANOZZO D.: A large-scale benchmark and an inclusion-based algorithm for continuous collision detection. *ACM Trans. Graph. 40*, 5 (sep 2021). doi:10.1145/3460775. 4

[WHK17] WINCHENBACH R., HOCHSTETTER H., KOLB A.: Infinite continuous adaptivity for incompressible sph. *ACM Trans. Graph. 36*, 4 (jul 2017). doi:10.1145/3072959.3073713. 15

[WK21] WINCHENBACH R., KOLB A.: Optimized refinement for spatially adaptive SPH. *ACM Trans. Graph. 40*, 1 (jan 2021). doi:10.1145/3363555. 15

[WKBB18] WEILER M., KOSCHIER D., BRAND M., BENDER J.: A physically consistent implicit viscosity solver for SPH fluids. *Computer Graphics Forum 37* (05 2018), 145–155. doi:10.1111/cgf.13349. 7

[WSG*18] WILLIS J. S., SCHALLER M., GONNET P., BOWER R. G., DRAPER P. W.: An efficient SIMD implementation of pseudo-verlet lists for neighbour interactions in particle-based codes. *Advances in Parallel Computing 32* (2018), 507–516. doi:10.3233/978-1-61499-843-3-507. 4

[XZB14] XU H., ZHAO Y., BARBIC J.: Implicit multibody penalty-based distributed contact. *Visualization and Computer Graphics, IEEE Transactions on 20* (09 2014), 1266–1279. doi:10.1109/TVCG.2014.2312013. 2, 15

[YLCH18] YAN X., LI C.-F., CHEN X., HU S.-M.: MPM simulation of interacting fluids and solids. *Computer Graphics Forum 37* (12 2018), 183–193. doi:10.1111/cgf.13523. 3

[YN06] YAMANE K., NAKAMURA Y.: Stable penalty-based model of frictional contacts. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* (2006), pp. 1904–1909. doi:10.1109/ROBOT.2006.1641984. 2

[YSC*18] YUE Y., SMITH B., CHEN P. Y., CHANTHARAYUKHONTHORN M., KAMRIN K., GRINSPUN E.: Hybrid grains: Adaptive coupling of discrete and continuum simulations of granular media. *ACM Trans. Graph. 37*, 6 (dec 2018). doi:10.1145/3272127.3275095. 2

[ZW13] ZHAO F., WACHEM B.: A novel quaternion integration approach for describing the behaviour of non-spherical particles. *Acta Mechanica 224* (12 2013). doi:10.1007/s00707-013-0914-2. 23

Appendix A: Diagonal Elements

Deriving the relationship between p_i and $V_i^{\text{err}}(t+\Delta t)$ as well as the relation between $\vec{\lambda}_r$ and $\vec{v}_r^{\text{tang}}(t+\Delta t)$ is required when building a Jacobi iteration from Equation 4.1 in order to compute correct values for α^P and α^F. As indicated in Algorithm 2, the diagonal elements are precomputed at the beginning of each solving procedure. Algorithm 5 presents an overview of the computation steps.

```
1 Function Compute Diagonals :
2     foreach fluid particle $f$ do
3         Compute $\frac{\partial}{\partial p_f} V_f^{\text{err}}(t+\Delta t)$        ▷ alg. 7
4     foreach rigid particle $r$ do
5         Compute $\frac{\partial}{\partial p_r} V_r^{\text{err}}(t+\Delta t)$        ▷ alg. 6
6         Compute $\frac{\partial}{\partial \vec{\lambda}_r} \vec{v}_r^{\text{tang}}(t+\Delta t)$        ▷ alg. 8
```

Algorithm 5: The relation between p_i and $V_i^{\text{err}}(t+\Delta t)$ as well as between $\vec{\lambda}_r$ and $\vec{v}_r^{\text{tang}}(t+\Delta t)$ are computed and stored for later use in the Jacobi updates.

Pressure

In the case of pressure, we have to distinguish between rigid particles r and fluid particles f. We start by deriving an expression for

$$\frac{\partial}{\partial p_r} V_r^{\text{err}}(t+\Delta t). \tag{A.1}$$

Plugging in Equation 3.10 and applying the chain rule yields

$$\begin{aligned}
&\frac{\partial}{\partial p_r} V_r^{\text{err}}(t+\Delta t) \\
&= -V_r^0 \Delta t \sum_{r_k} \left(\frac{\partial}{\partial p_r} \vec{v}_r(t+\Delta t) - \frac{\partial}{\partial p_r} \vec{v}_{r_k}(t+\Delta t) \right) \cdot \vec{\nabla} W_{rr_k} \\
&\quad - V_r^0 \Delta t \sum_{r_b} \left(\frac{\partial}{\partial p_r} \vec{v}_r(t+\Delta t) - \frac{\partial}{\partial p_r} \vec{v}_{r_b}(t+\Delta t) \right) \cdot \vec{\nabla} W_{rr_b}
\end{aligned} \tag{A.2}$$

which, together with Equation 3.11 gives us

$$\begin{aligned}
\frac{\partial}{\partial p_r} \vec{v}_r(t+\Delta t) &= \frac{\partial}{\partial p_r} \vec{v}_R(t+\Delta t) \\
&\quad + \frac{\partial}{\partial p_r} \vec{\omega}_R(t+\Delta t) \times (\vec{x}_r - \vec{x}_R)
\end{aligned} \tag{A.3a}$$

$$\frac{\partial}{\partial p_r} \vec{v}_R(t+\Delta t) = \Delta t \frac{1}{M_R} \sum_{\tilde{r} \in R} \frac{\partial}{\partial p_r} \vec{F}_{\tilde{r}}^P \tag{A.3b}$$

$$\frac{\partial}{\partial p_r}\vec{\omega}_R(t+\Delta t) = \Delta t \mathbf{I}_R^{-1}\sum_{\tilde{r}\in R}(\vec{x}_{\tilde{r}} - \vec{x}_R) \times \frac{\partial}{\partial p_r}\vec{F}^P_{\tilde{r}} \tag{A.3c}$$

$$\begin{aligned}\frac{\partial}{\partial p_r}\vec{v}_{r_k}(t+\Delta t) = &\frac{\partial}{\partial p_r}\vec{v}_K(t+\Delta t)\\ &+\frac{\partial}{\partial p_r}\vec{\omega}_K(t+\Delta t)\times(\vec{x}_{r_k}-\vec{x}_K)\end{aligned} \tag{A.3d}$$

$$\frac{\partial}{\partial p_r}\vec{v}_K(t+\Delta t) = \Delta t\frac{1}{M_K}\sum_{\tilde{k}\in K}\frac{\partial}{\partial p_r}\vec{F}^P_{\tilde{k}} \tag{A.3e}$$

$$\frac{\partial}{\partial p_r}\vec{\omega}_K(t+\Delta t) = \Delta t \mathbf{I}_K^{-1}\sum_{\tilde{k}\in K}\left(\vec{x}_{\tilde{k}} - \vec{x}_K\right) \times \frac{\partial}{\partial p_r}\vec{F}^P_{\tilde{k}} \tag{A.3f}$$

$$\frac{\partial}{\partial p_r}\vec{v}_{r_b}(t+\Delta t) = \vec{0} \tag{A.3g}$$

with all particles $\tilde{r}$ belonging to the same rigid body R as r does and with all particles $\tilde{k}$ belonging to the same rigid body K as r_k does. Pressure p_r influences $\vec{F}^P_{\tilde{r}}$ only for $\tilde{r} = r$, so we can simplify Equations A.3b and A.3c to

$$\frac{\partial}{\partial p_r}\vec{v}_R(t+\Delta t) = \Delta t\frac{1}{M_R}\frac{\partial}{\partial p_r}\vec{F}^P_r \tag{A.4a}$$

$$\frac{\partial}{\partial p_r}\vec{\omega}_R(t+\Delta t) = \Delta t\mathbf{I}_R^{-1}(\vec{x}_r - \vec{x}_R)\times\frac{\partial}{\partial p_r}\vec{F}^P_r \tag{A.4b}$$

The last remaining pieces are the dependencies of $\vec{F}^P_r$ and $\vec{F}^P_{\tilde{k}}$ on p_r. By using Equation 3.13 we get

$$\begin{aligned}\frac{\partial}{\partial p_r}\vec{F}^P_r =& -V_r\sum_{r_k}V_{r_k}\vec{\nabla}W_{rr_k}\\ &-V_r\sum_{r_b}2V_{r_b}\vec{\nabla}W_{rr_k}\end{aligned} \tag{A.5a}$$

$$\frac{\partial}{\partial p_r}\vec{F}^P_{\tilde{k}} = -V_{\tilde{k}}V_r\vec{\nabla}W_{\tilde{k}r}. \tag{A.5b}$$

Note that $\partial/\partial p_r\ \vec{F}^P_{\tilde{k}}$ equals zero if particle $\tilde{k}$ is not a neighbor of r, so in Equations A.3e and A.3f it suffices to sum over particles $\tilde{k}\in K$ neighboring r. We made the assumption that $p_{r_b} = p_r$, which corresponds to the boundary handling by Akinci et al. [AIA*12]. When employing a different boundary handling scheme one may accurately use the exact expression for p_{r_b}, or use $p_{r_b} = p_r$ nonetheless as it still approximates p_{r_b} well enough. In combination, Equations A.2, A.3a, A.3d to A.3g, A.4 and A.5 can be used to evaluate $\partial/\partial p_r\ V^{\text{err}}_r(t+\Delta t)$. Even though this procedure looks complex, it can be implemented efficiently as shown in Algorithm 6.

For fluid particles f, the relation between pressure p_f and predicted volume error $V^{\text{err}}_f(t+\Delta t)$ can be computed equivalently. Similar to Equation A.2, we first plug in the expression for the volume error of a fluid particle f given in Equation 3.14:

$$\begin{aligned}&\frac{\partial}{\partial p_f}V^{\text{err}}_f(t+\Delta t)\\ &= -V^0_f\Delta t\sum_{f_f}\left(\frac{\partial}{\partial p_f}\vec{v}_f(t+\Delta t) - \frac{\partial}{\partial p_f}\vec{v}_{f_f}(t+\Delta t)\right)\cdot\vec{\nabla}W_{ff_f}\\ &\quad -V^0_f\Delta t\sum_{f_k}\left(\frac{\partial}{\partial p_f}\vec{v}_f(t+\Delta t) - \frac{\partial}{\partial p_f}\vec{v}_{f_k}(t+\Delta t)\right)\cdot\vec{\nabla}W_{ff_k}\\ &\quad -V^0_f\Delta t\sum_{f_b}\left(\frac{\partial}{\partial p_f}\vec{v}_f(t+\Delta t) - \frac{\partial}{\partial p_f}\vec{v}_{f_b}(t+\Delta t)\right)\cdot\vec{\nabla}W_{ff_b}\end{aligned} \tag{A.6}$$

Here, we can clearly see that in contrast to the generic fluid solvers used by Gissler et al. [GPB*19], our method indeed does consider the effect of p_f onto $\vec{v}_{f_k}(t+\Delta t)$ during the computation of $\partial/\partial p_f\ V^{\text{err}}_f(t+\Delta t)$. By considering Equations 3.11 and 3.15 we have

$$\frac{\partial}{\partial p_f}\vec{v}_f(t+\Delta t) = \Delta t\frac{1}{m_f}\frac{\partial}{\partial p_f}\vec{F}^P_f \tag{A.7a}$$

$$\frac{\partial}{\partial p_f}\vec{v}_{f_f}(t+\Delta t) = \Delta t\frac{1}{m_{f_f}}\frac{\partial}{\partial p_f}\vec{F}^P_{f_f} \tag{A.7b}$$

$$\begin{aligned}\frac{\partial}{\partial p_f}\vec{v}_{f_k}(t+\Delta t) = &\frac{\partial}{\partial p_f}\vec{v}_K(t+\Delta t)\\ &+\frac{\partial}{\partial p_f}\vec{\omega}_K(t+\Delta t)\times\left(\vec{x}_{f_k}-\vec{x}_K\right)\end{aligned} \tag{A.7c}$$

$$\frac{\partial}{\partial p_f}\vec{v}_K(t+\Delta t) = \Delta t\frac{1}{M_K}\sum_{\tilde{k}\in K}\frac{\partial}{\partial p_f}\vec{F}^P_{\tilde{k}} \tag{A.7d}$$

$$\frac{\partial}{\partial p_f}\vec{\omega}_K(t+\Delta t) = \Delta t\mathbf{I}_K^{-1}\sum_{\tilde{k}\in K}\left(\vec{x}_{\tilde{k}}-\vec{x}_K\right)\times\frac{\partial}{\partial p_f}\vec{F}^P_{\tilde{k}} \tag{A.7e}$$

$$\frac{\partial}{\partial p_f}\vec{v}_{f_b}(t+\Delta t) = \vec{0}. \tag{A.7f}$$

Here, $\tilde{k}$ are all particles belonging to the same rigid body K as f_k does. Similar to Equations A.3e and A.3f, in order to evaluate Equations A.7d and A.7e it suffices to sum over particles $\tilde{k}\in K$ neighboring f. The dependency of a fluid particle's pressure force $\vec{F}^P$ on

```
1 Function Pressure - Volume Error Relation of r:
2     Compute $\frac{\partial}{\partial p_r}\vec{v}_r(t+\Delta t)$          ▷ eq. A.3a, A.4 and A.5a
3     foreach neighboring rigid body K do
4         Compute $\frac{\partial}{\partial p_r}\vec{v}_K(t+\Delta t)$      ▷ eq. A.3e and A.5b
5         Compute $\frac{\partial}{\partial p_r}\vec{\omega}_K(t+\Delta t)$      ▷ eq. A.3f and A.5b
6     Compute $\frac{\partial}{\partial p_r}V^{\text{err}}_r(t+\Delta t)$          ▷ eq. A.2, A.3d and A.3g
7     return $\frac{\partial}{\partial p_r}V^{\text{err}}_r(t+\Delta t)$
```

Algorithm 6: The procedure to efficiently compute the relation between pressure p_r and volume error $V^{\text{err}}_r(t+\Delta t)$ for a rigid particle r. All quantities that are explicitly computed in the algorithm are precomputed and used again later while quantities that are not mentioned are computed on-the-fly.

p_f is also given in Equation 3.15:

$$\begin{aligned}\frac{\partial}{\partial p_f}\vec{F}_f^P = &-V_f\sum_{f_f}V_{f_f}\vec{\nabla}W_{ff_f}\\ &-V_f\sum_{f_b}2V_{f_b}\vec{\nabla}W_{ff_b}\\ &-V_f\sum_{f_k}2V_{f_k}\vec{\nabla}W_{ff_k}\end{aligned}\qquad\text{(A.8a)}$$

$$\frac{\partial}{\partial p_f}\vec{F}_{f_f}^P = -V_{f_f}V_f\vec{\nabla}W_{f_f f}\qquad\text{(A.8b)}$$

Again, we assume that $p_{f_b} = p_f$ is used as an estimate for pressure values at kinematic boundary particles f_b. By using Equation 3.13 we find the expression

$$\frac{\partial}{\partial p_f}\vec{F}_{\tilde{k}}^P = -2V_{\tilde{k}}V_f\vec{\nabla}W_{\tilde{k}f}.\qquad\text{(A.9)}$$

Together, Equations A.6 to A.9 describe the relation between p_f and volume error V_f^{err}. Again, to maintain an overview, we summarized the computation procedure in Algorithm 7.

```
1 Function Pressure - Volume Error Relation of $f$:
2     Compute $\frac{\partial}{\partial p_f}\vec{v}_f(t+\Delta t)$          ▷ eq. A.7a and A.8a
3     foreach neighboring rigid body $K$ do
4         Compute $\frac{\partial}{\partial p_f}\vec{v}_K(t+\Delta t)$      ▷ eq. A.7d and A.9
5         Compute $\frac{\partial}{\partial p_f}\vec{\omega}_K(t+\Delta t)$      ▷ eq. A.7e and A.9
6     Compute $\frac{\partial}{\partial p_f}V_f^{\text{err}}(t+\Delta t)$      ▷ eq. A.6, A.7b, A.7c, A.7f and A.8b
7     return $\frac{\partial}{\partial p_f}V_f^{\text{err}}(t+\Delta t)$
```

Algorithm 7: Similar to Algorithm 6, we can efficiently compute the relation between pressure p_f and volume error $V_f^{\text{err}}(t+\Delta t)$ for a fluid particle f. Again, all quantities that are explicitly computed in the algorithm are precomputed and used again later while quantities that are not mentioned are computed on-the-fly.

Friction

Inside the modified Jacobi iteration for friction given in Equation 4.8, we need to compute $\partial/\partial\vec{\lambda}_r\,\vec{v}_r^{\text{tang}}(t+\Delta t)$. Similar to before, we find an expression for $\partial/\partial\vec{\lambda}_r\,\vec{v}_r^{\text{tang}}(t+\Delta t)$ by successively applying the chain rule. Starting with Equation 3.21b we have

$$\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r^{\text{tang}}(t+\Delta t) = \left(\mathbb{1}-\vec{n}_r\vec{n}_r^\top\right)\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r^{\text{rel}}(t+\Delta t).\qquad\text{(A.10)}$$

Note that $\vec{n}_r$ depends on normal forces $\vec{F}_r^N$ which in turn depend on pressure values $\mathbf{p}$ that are computed simultaneously to frictional forces. However, the direction of $\vec{F}_r^N$ does not change during the solving procedure, so we can precompute the term $\vec{F}_r^N/p_r$ and use it to estimate $\vec{n}_r$ as well as an efficient way to compute $\vec{F}_r^N$ given p_r:

$$\begin{aligned}\frac{\vec{F}_r^N}{p_r} = &-V_r\sum_{r_k}V_{r_k}\vec{\nabla}W_{rr_k}\\ &-2V_r\sum_{r_b}V_{r_b}\vec{\nabla}W_{rr_b}\end{aligned}\qquad\text{(A.11a)}$$

$$\vec{n}_r = \frac{\vec{F}_r^N/p_r}{|\vec{F}_r^N/p_r|}\qquad\text{(A.11b)}$$

The term $\partial/\partial\vec{\lambda}_r\,\vec{v}_r^{\text{rel}}(t+\Delta t)$ in Equation A.10 is also constant during one simulation step. Using Equation 3.21a, we can write

$$\begin{aligned}&\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r^{\text{rel}}(t+\Delta t)\\ &=\sum_{r_k}V_{r_k}\left(\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r(t+\Delta t)-\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_{r_k}(t+\Delta t)\right)W_{rr_k}\\ &+\sum_{r_b}V_{r_b}\left(\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r(t+\Delta t)-\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_{r_b}(t+\Delta t)\right)W_{rr_b}\end{aligned}\qquad\text{(A.12)}$$

where velocities can be written as described in Equation 3.11:

$$\begin{aligned}\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r(t+\Delta t) = &\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_R(t+\Delta t)\\ &-[\vec{x}_r-\vec{x}_R]_\times\frac{\partial}{\partial\vec{\lambda}_r}\vec{\omega}_R(t+\Delta t)\end{aligned}\qquad\text{(A.13a)}$$

$$\begin{aligned}\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_{r_k}(t+\Delta t) = &\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_K(t+\Delta t)\\ &-[\vec{x}_{r_k}-\vec{x}_K]_\times\frac{\partial}{\partial\vec{\lambda}_r}\vec{\omega}_K(t+\Delta t)\end{aligned}\qquad\text{(A.13b)}$$

$$\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_{r_b}(t+\Delta t) = \mathbf{0}\qquad\text{(A.13c)}$$

Note that we switched the order of the cross products to express them as matrix multiplications. Frictional forces $\vec{F}^F$ and torque $\vec{\tau}^F$ are computed in the friction mirroring step described in Equation 3.22. This gives us

$$\begin{aligned}\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_R(t+\Delta t) &= \Delta t\frac{1}{M_R}\sum_{\tilde{r}\in R}\frac{\partial}{\partial\vec{\lambda}_r}\vec{F}_{\tilde{r}}^F\\ &=\Delta t\frac{1}{M_R}\sum_{r_k}\tilde{V}_r\mathbb{1}W_{rr_k}\\ &+\Delta t\frac{1}{M_R}\sum_{r_b}\tilde{V}_r\mathbb{1}W_{rr_b}\end{aligned}\qquad\text{(A.14a)}$$

$$\begin{aligned}\frac{\partial}{\partial\vec{\lambda}_r}\vec{\omega}_R(t+\Delta t) &= \Delta t\mathbf{I}_R^{-1}\sum_{\tilde{r}\in R}\frac{\partial}{\partial\vec{\lambda}_r}\vec{\tau}_{\tilde{r}}^F\\ &=\Delta t\mathbf{I}_R^{-1}\sum_{r_k}\left[\frac{1}{2}\vec{x}_r+\frac{1}{2}\vec{x}_{r_k}-\vec{x}_R\right]_\times\tilde{V}_r\mathbb{1}W_{rr_k}\\ &+\Delta t\mathbf{I}_R^{-1}\sum_{r_b}\left[\frac{1}{2}\vec{x}_r+\frac{1}{2}\vec{x}_{r_b}-\vec{x}_R\right]_\times\tilde{V}_r\mathbb{1}W_{rr_b}\end{aligned}\qquad\text{(A.14b)}$$

$$\begin{aligned}\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_K(t+\Delta t) &= \Delta t\frac{1}{M_K}\sum_{\tilde{k}\in K}\frac{\partial}{\partial\vec{\lambda}_r}\vec{F}_{\tilde{k}}^F\\ &=\Delta t\frac{1}{M_K}\sum_{\tilde{k}\in K}-\tilde{V}_r\mathbb{1}W_{\tilde{k}r}\end{aligned}\qquad\text{(A.14c)}$$

$$\frac{\partial}{\partial\vec{\lambda}_r}\vec{\omega}_K(t+\Delta t)=\Delta t\mathbf{I}_K^{-1}\sum_{\tilde{k}\in K}\frac{\partial}{\partial\vec{\lambda}_r}\vec{\tau}^F_{\tilde{k}}$$
$$=\Delta t\mathbf{I}_K^{-1}\sum_{\tilde{k}\in K}-\left[\frac{1}{2}\vec{x}_{\tilde{k}}+\frac{1}{2}\vec{x}_r-\vec{x}_K\right]_\times\widetilde{V}_r\mathbb{1}W_{\tilde{k}r}. \tag{A.14d}$$

Again, we see that in Equations A.14c and A.14d it suffices to sum over particles $\tilde{k}\in K$ that neighbor r. In combination, Equations A.10 to A.14 can be used to compute $\partial/\partial\vec{\lambda}_r\ \vec{v}_r^{\text{tang}}(t+\Delta t)$. The computation procedure is summarized in Algorithm 8.

```
1 Function Friction - Tangential Velocity Relation of r:
2     Compute $\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r(t+\Delta t)$          ▷ eq. A.13a, A.14a and A.14b
3     foreach neighboring rigid body K do
4         Compute $\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_K(t+\Delta t)]$      ▷ eq. A.14c
5         Compute $\frac{\partial}{\partial\vec{\lambda}_r}\vec{\omega}_K(t+\Delta t)$  ▷ eq. A.14d
6     Compute $\vec{F}_r^N/p_r$ and $\vec{n}_r$                                       ▷ eq. A.11
7     Compute $\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r^{\text{tang}}(t+\Delta t)$  ▷ eq. A.10, A.12, A.13b and A.13c
8     return $\frac{\partial}{\partial\vec{\lambda}_r}\vec{v}_r^{\text{tang}}(t+\Delta t)$
```

Algorithm 8: An efficient implementation to compute the relation between frictional multiplier $\vec{\lambda}_r$ and tangential velocity $\vec{v}_r^{\text{tang}}(t+\Delta t)$ for a rigid particle r. Again, quantities that are explicitly mentioned in the algorithm are precomputed for repeated use later on, all others are computed ad-hoc.

Appendix B: Time Integration

During the derivation of our implicit pressure and friction force computation model, we assumed that positions $\vec{x}_i(t+\Delta t)$ and velocities $\vec{v}_i(t+\Delta t)$ depend on forces computed at timestep t. To reflect this, we integrate positions and velocities in a semi implicit manner. Starting with fluid particles, the integration step is quite simple:

$$\vec{v}_f(t+\Delta t)=\vec{v}_f^*+\Delta t\frac{1}{m_f}\vec{F}_f^P \tag{3.15 revisited}$$
$$\vec{x}_f(t+\Delta t)=\vec{x}_f(t)+\Delta t\vec{v}_f(t+\Delta t) \tag{B.1a}$$

Integrating velocities and positions of rigid particles by using the predicted velocities from Equation 3.11 would result in particle velocities $\vec{v}_r$ that do not match the body's velocities $\vec{v}_R$ and $\vec{\omega}_R$ at timestep $t+\Delta t$ since positions $\vec{x}_r(t)$ are used in the prediction. Additionally, integrating particle positions using $\vec{x}_r(t+\Delta t)=\vec{x}_r(t)+\Delta t\vec{v}_r(t+\Delta t)$ would have the effect that distances between particles sampling the same rigid body change over time. To prevent this, we first integrate the body's position $\vec{x}_R$, orientation $\mathbf{q}_R$, translational velocity $\vec{v}_R$ and angular velocity $\vec{\omega}_R$ using

$$\vec{v}_R(t+\Delta t)=\vec{v}_R^*+\Delta t\frac{1}{M_R}\sum_{\tilde{r}\in R}\left(\vec{F}^P_{\tilde{r}}+\vec{F}^F_{\tilde{r}}\right) \tag{3.11b revisited}$$

$$\vec{\omega}_R(t+\Delta t)=\vec{\omega}_R^*+\Delta t\mathbf{I}_R^{-1}\sum_{\tilde{r}\in R}\left(\vec{x}_{\tilde{r}}(t)-\vec{x}_R(t)\right)\times\vec{F}^P_{\tilde{r}}+\Delta t\mathbf{I}_R^{-1}\sum_{\tilde{r}\in R}\vec{\tau}^F_{\tilde{r}}. \tag{3.11c revisited}$$

$$\vec{x}_R(t+\Delta t)=\vec{x}_R+\Delta t\vec{v}_R(t+\Delta t) \tag{B.2a}$$

$$\mathbf{q}_R(t+\Delta t)=\left[\cos\frac{\Delta t\left|\vec{\omega}_R(t+\Delta t)\right|}{2},\ \sin\frac{\Delta t\left|\vec{\omega}_R(t+\Delta t)\right|}{2}\frac{\vec{\omega}_R(t+\Delta t)}{\left|\vec{\omega}_R(t+\Delta t)\right|}\right]\mathbf{q}_R(t). \tag{B.2b}$$

We use quaternions to represent orientations and employed a time integration scheme proposed by F. Zhao & B. Wachem [ZW13] that is able to integrate quaternions with no need for renormalization. Afterwards, we compute the new particle positions and velocities directly based on $\vec{x}_R(t+\Delta t)$, $\mathbf{q}_R(t+\Delta t)$, $\vec{v}_R(t+\Delta t)$ and $\vec{\omega}_R(t+\Delta t)$:

$$\vec{x}_r(t+\Delta t)=\vec{x}_R(t+\Delta t)+\mathbf{q}_R(t+\Delta t)\left[0,\vec{x}_r(0)-\vec{x}_R(0)\right]\mathbf{q}_R^{-1}(t+\Delta t) \tag{B.3a}$$
$$\vec{v}_r(t+\Delta t)=\vec{v}_R(t+\Delta t)+\vec{\omega}_R(t+\Delta t)\times\left(\vec{x}_r(t+\Delta t)-\vec{x}_R(t+\Delta t)\right) \tag{B.3b}$$

Algorithm 9 summarizes the time integration step.

```
1  Function Integrate x, v, q, ω:
2      foreach fluid particle f do
3          Compute $\vec{v}_f(t+\Delta t)$        ▷ eq. 3.15
4          Compute $\vec{x}_f(t+\Delta t)$        ▷ eq. B.1a
5      foreach rigid body R do
6          Compute $\vec{v}_R(t+\Delta t)$        ▷ eq. 3.11b
7          Compute $\vec{\omega}_R(t+\Delta t)$   ▷ eq. 3.11c
8          Compute $\vec{x}_R(t+\Delta t)$        ▷ eq. B.2a
9          Compute $\mathbf{q}_R(t+\Delta t)$     ▷ eq. B.2b
10     foreach rigid particle r do
11         Compute $\vec{x}_r(t+\Delta t)$        ▷ eq. B.3a
12         Compute $\vec{v}_r(t+\Delta t)$        ▷ eq. B.3b
```

Algorithm 9: The implementation of the Euler-Cromer time integration scheme used in our simulation.