Fast Corotated Elastic SPH Solids with Implicit Zero-Energy Mode Control

TASSILO KUGELSTADT, RWTH Aachen University, Germany
JAN BENDER, RWTH Aachen University, Germany
JOSÉ ANTONIO FERNÁNDEZ-FERNÁNDEZ, RWTH Aachen University, Germany
STEFAN RHYS JESKE, RWTH Aachen University, Germany
FABIAN LÖSCHNER, RWTH Aachen University, Germany
ANDREAS LONGVA, RWTH Aachen University, Germany

Fig. 1. Left: Stable simulation of eight walrus models (210k particles) that are pushed through a tight funnel and impact the water in a container with 1.2M fluid particles. Right: To showcase the coupling capabilities of our method 10 deformable solids and 4 rigid tori are dropped into a bowl while water and a highly viscous fluid are poured on top. A total of 800k particles are used for the fluids and 252k for the elastic objects.

We develop a new operator splitting formulation for the simulation of corotated linearly elastic solids with Smoothed Particle Hydrodynamics (SPH). Based on the technique of Kugelstadt et al. [2018] originally developed for the Finite Element Method (FEM), we split the elastic energy into two separate terms corresponding to stretching and volume conservation, and based on this principle, we design a splitting scheme compatible with SPH. The operator splitting scheme enables us to treat the two terms separately, and because the stretching forces lead to a stiffness matrix that is constant in time, we are able to prefactor the system matrix for the implicit integration step. Solid-solid contact and fluid-solid interaction is achieved through a unified pressure solve. We demonstrate more than an order of magnitude improvement in computation time compared to a state-of-the-art SPH simulator for elastic solids.

We further improve the stability and reliability of the simulation through several additional contributions. We introduce a new implicit penalty mechanism that suppresses zero-energy modes inherent in the SPH formulation for elastic solids, and present a new, physics-inspired sampling algorithm for generating high-quality particle distributions for the rest shape of an elastic solid. We finally also devise an efficient method

Authors' addresses: Tassilo Kugelstadt, kugelstadt@cs.rwth-aachen.de, RWTH Aachen University, Aachen, Germany; Jan Bender, bender@cs.rwth-aachen.de, RWTH Aachen University, Aachen, Germany; José Antonio Fernández-Fernández, fernandez@cs.rwth-aachen.de, RWTH Aachen University, Aachen, Germany; Stefan Rhys Jeske, jeske@cs.rwth-aachen.de, RWTH Aachen University, Aachen, Germany; Fabian Löschner, loeschner@cs.rwth-aachen.de, RWTH Aachen University, Aachen, Germany; Andreas Longva, longva@cs.rwth-aachen.de, RWTH Aachen University, Aachen, Germany.

for interpolating vertex positions of a high-resolution surface mesh based on the SPH particle positions for use in high-fidelity visualization.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Smoothed Particle Hydrodynamics, deformable solids, fluid simulation, solid-fluid coupling

ACM Reference Format:
Tassilo Kugelstadt, Jan Bender, José Antonio Fernández-Fernández, Stefan Rhys Jeske, Fabian Löschner, and Andreas Longva. 2021. Fast Corotated Elastic SPH Solids with Implicit Zero-Energy Mode Control. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 3 (September 2021), 21 pages. https://doi.org/10.1145/3480142

1 INTRODUCTION

Many of the current state-of-the-art methods for simulating fluids in computer graphics are meshless, either by discretizing the dynamics exclusively with particles (e.g. SPH) or in hybrid synergy with a Eulerian discretization (FLIP, APIC, etc.). Though meshless methods are considered somewhat less efficient than mesh-based methods for the simulation of elastic solids, they excel in their flexibility. A unified representation for solids and fluids significantly simplifies the coupling between the different physical models involved, and furthermore facilitates state transitions, such as melting or solidification.

SPH is an established simulation method for fluids in computer graphics. Recent work has made SPH an increasingly compelling alternative also for elastic solids. Our work extends the state-of-the-art simulation of elastic solids with SPH in several ways.

Our main contribution is a new SPH operator splitting formulation of the corotated linear elastic material model. The technique was initially introduced by Kugelstadt et al. [2018] for the simulation of elastic solids with the FEM, and relies on the realization that the stiffness matrix can be decomposed into a sum of a *constant* matrix associated with the stretch forces and a time-dependent matrix corresponding to the volume conserving terms. For many compressible objects, the constant matrix term is generally dominating. We demonstrate that this decomposition leads to a constant matrix for the stretching term also when we use an SPH discretization, and we introduce an operator splitting scheme tailored to the SPH formulation for elastic solids so that the stretching force terms are handled separately by a prefactored direct solve, while the volume conserving terms are incorporated into a later Conjugate Gradient solve. Finally, solid-fluid coupling and solid-solid contact handling — including self-contact — are achieved by integrating the solid particles into the preexisting pressure solver. We demonstrate that our formulation enables more than an order of magnitude faster time-stepping than the state-of-the-art method by Peer et al. [2018].

A well-known deficiency of the SPH solid formulation used here is the presence of *zero-energy modes*, which leads to unstable simulations in the form of strong local oscillations and an inability to return to the rest state. Ganzenmüller [2015] introduced a control mechanism that suppresses the zero-energy modes by introducing a correction force through explicit time integration. Inspired by this approach, we introduce an improved zero-energy mode penalty force that conserves linear and angular momentum, and whose stiffness parameter can be controlled independently of the material parameters of the solid. Unless small time steps are taken, the explicit integration used by Ganzenmüller can itself lead to instabilities. In contrast, we ensure stability by implicitly integrating our penalty force along with the stretching terms of the elastic model. The contributions to the system matrix are constant in time and can therefore be included in the prefactored system matrix, and hence have no impact on the time required to solve the linear system. We demonstrate through multiple experiments the benefits of implicit zero-energy mode suppression.

By using SPH to simulate solids, it is not necessary to obtain a high-quality simulation mesh as is the case with the FEM. Instead only a sampling of points in the solid interior is sufficient. This can most simply be obtained by sampling points on a regular grid and discarding points outside the solid. However, as we show in the accompanying video, regular sampling might lead to problematic particle configurations in thin features, which results in significant simulation artifacts. To alleviate this problem, we introduce a physics-inspired sampling algorithm that we demonstrate to significantly reduce the likelihood of simulation artifacts due to the particle sampling. Moreover, we experimentally verify that our sampling method produces visually higher quality results than the state-of-the-art blue noise sampling algorithm [Jiang et al. 2015b].

In order to visualize deformed solids, we use a high-fidelity surface mesh for the rest shape that is deformed along with the simulation. Our final contribution is a straightforward and efficient algorithm for computing vertex positions of a high-resolution mesh based on the positions of the SPH solid particles. Our interpolation algorithm preserves the fidelity of the rest pose mesh while adhering to the deformation represented by the SPH particles.

We demonstrate that our method is suitable for use in complex, multi-physics scenarios through several experiments, examples of which can be found in Figure 1.

2 RELATED WORK

There exists a broad range of approaches to simulate deformable objects in computer animation. Mass-spring systems are perhaps the most simple models and were introduced in the early days of physically based animation. They can be used to simulate deformable materials like cloth [Bridson et al. 2002; Provot 1995] but also volumetric solids [Teschner et al. 2004]. Even today they are favored for their ease of implementation and potentially low computational cost. However, in general, mass-spring systems are not motivated by a continuum mechanics perspective which makes it harder to obtain behavior that resembles common material models. Similar characteristics are shared by shape matching [Müller et al. 2005] and other position based dynamics approaches [Müller et al. 2006]. They were successfully used to build unified frameworks to simulate rigid bodies, deformables and fluids, as presented for example by Macklin et al. [2014]. For an overview of these methods we refer to the survey of Bender et al. [2017].

For our use case we will focus on methods that are directly derived from continuum mechanics theory. One of the most wide-spread approaches in this category is FEM. For a general overview covering FEM we refer to the tutorial by Sifakis and Barbic [2012] or the survey by Nealen et al. [2006]. This publication in particular builds upon previous work by Kugelstadt et al. [2018] who presented a fast operator-splitting method for the simulation of corotated linear elastic deformables with FEM. As mentioned earlier, one core contribution of this work is to adapt this approach to a particle-based discretization using SPH.

SPH. Smoothed particle hydrodynamics is a popular particle-based method from the fields of computational physics and fluid mechanics [Ganzenmüller 2015; Monaghan 2012] adopted by the graphics community originally for materials with large inelastic deformations [Desbrun and Gascuel 1996]. A vast amount of research in the field of computer animation proposes extensions or improvements for different aspects of SPH fluid simulations. This includes works on implicit pressure solvers [Bender and Koschier 2017; Cornelis et al. 2019; Ihmsen et al. 2014; Weiler et al. 2016], and implicit viscosity formulations [Peer et al. 2015; Weiler et al. 2018]. Considering the great results obtained with SPH for fluids, it was a natural step to experiment with unified simulators that support other materials and phases. Notable contributions in this direction, in particular for robust interaction of SPH fluids and rigid bodies are the works by Akinci et al. [2012], Akbay et

al. [2018] and especially the recent work on strong coupling by Gissler et al. [2019]. For a general overview of SPH we refer to the tutorial by Koschier et al. [2019].

In the context of deformable solids, Solenthaler et al. [2007] proposed to use SPH to evaluate the deformation gradient to model a linear elastic material. As the standard SPH formulation is not 1^{st}-order consistent [Bonet and Lok 1999], the resulting deformation gradients were not rotationally invariant. This resulted in artifacts, specifically forces counteracting the rotations. Becker et al. [2009] instead proposed to use shape matching to determine the deformation gradient and employed corotated linear elasticity combined with explicit time integration. Instead of shape matching, Peer et al. [2018] again proposed an entirely SPH based formulation that addressed the known issues. They resorted to a kernel gradient correction proposed by Bonet and Lok [1999] which ensures that the SPH gradients are 1^{st}-order consistent. Peer et al. also use a corotated formulation combined with implicit time integration allowing for large time steps. An implicit SPH pressure solver enforces incompressibility and allows coupling with fluids. Instead of relying entirely on SPH, Abu Rumman et al. [2019] developed a method for coupling PBD deformables with SPH fluids. They introduced a free-surface formulation that does not explicitly depend on a surface or volumetric particle sampling of the deformables. This is relatively fast and interactive simulations can be achieved but also shares the downsides of PBD deformables, that it is difficult to model common elastic materials. Further works that couple SPH with other methods for the simulation of deformables include the papers of Dagenais et al. [2012] who used a predictor-corrector approach with shape matching and Huber et al. [2015] who presented a coupling method for cloth with SPH supporting wetting and no-slip boundary conditions. Recently, Gissler et al. [2020] proposed a solver for the simulation of compressible elastoplastic snow. They introduced a compressible SPH pressure solver which is coupled with a linear implicit elasticity solver inspired by the work of Peer et al. but adapted for plasticity.

Alternative approaches. In recent years, the material point method (MPM) gained popularity in the graphics community for multi-material simulations due to its versatility as demonstrated for example by Stomakhin et al. by simulating snow and phase change of materials [Stomakhin et al. 2014]. MPM is a hybrid Eulerian and Lagrangian method that transfers quantities back and forth between a particle and a grid representation. Early problems with large dissipation or instabilities due to the transfer were addressed by the APIC method by Jiang et al. [2015a]. A large amount of publications explores MPM for various uses cases with multi-material or multi-phase simulations for example frictional contact [Han et al. 2019], liquid-solid coupling [Fei et al. 2019, 2018, 2017] and strong nonlinear coupling without MPM-typical sticking [Fang et al. 2020]. With SPH we are instead using a purely Lagrangian approach which also avoids problems like artificial plasticity that might arise when relying on a hybrid method without special care.

Inspired from computational physics, models based on peridynamics theory were also introduced to the field of computer animation. At its core, peridynamics is a continuum based method using global integral equations which can be discretized using particles. It is especially well suited for simulating fracturing [Chen et al. 2018; Levine et al. 2014] but can also be used for elastoplastic, viscous and granular materials [He et al. 2018].

Another category of methods are particle-based formulations using MLS interpolation. First introduced in the field of physically based animation by Müller et al. [2004], a typical MLS-based method uses least squares to fit polynomials of arbitrary order to a quantity in a particle's neighborhood. Due to the possible higher order of the polynomials this allows more accurate evaluations than in standard SPH formulations. MLS was proposed for elastic deformables [Keiser et al. 2005], plasticity [Gerszewski et al. 2009; Jones et al. 2014; Zhou et al. 2013], fracturing [Pauly et al. 2005], unified volumetric, shell and rod simulation [Martin et al. 2010] and multi-phase simulation [Chen

et al. 2020]. However, the higher-order approaches increase complexity, can be significantly more expensive and without special care, degenerate particle configurations can also introduce problems. Overall, the SPH approach with the kernel gradient correction we use is quite similar to a 1^{st}-order MLS interpolation. Therefore, we believe that our approach can also be adapted to be used in an MLS-based simulator. For now, our goal was to couple a fast simulation of deformables with other recent advances specifically from the SPH community.

3 ELASTIC MODEL

This section introduces the continuum formulation of deformable solids and an SPH discretization of the corresponding forces. In Section 4 we discuss the problem of zero-energy modes and introduce a novel implicit method for zero-energy mode suppression. Our efficient time discretization using operator splitting is introduced in Section 5. Finally, we discuss solid-fluid coupling, our particle sampling method and mesh skinning in Sections 6, 7 and 8.

3.1 Continuum Formulation

To simulate deformable objects we will employ the *total Lagrangian formulation* of SPH. This means the equations of motion and the constitutive laws are described using the positions in the undeformed rest pose, the so called reference coordinates $\mathbf{X}$. The deformed positions $\mathbf{x}$ of a body at time t can be found with the deformation mapping $\mathbf{x} = \phi(\mathbf{X}, t)$ (see [Sifakis and Barbic 2012] for more details). Elastic energies are usually expressed in terms of the deformation gradient $\mathbf{F}$ which can be found by taking the derivative of the deformation mapping w.r.t. the reference positions $\mathbf{X}$:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}. \tag{1}$$

In the following we use the corotated linear constitutive model with the elastic energy density [Sifakis and Barbic 2012]

$$\Psi = \mu \parallel \mathbf{F} - \mathbf{R} \parallel_F^2 + \frac{\lambda}{2} \mathrm{tr}(\mathbf{R}^T\mathbf{F} - \mathbb{1})^2, \tag{2}$$

where $\mathbf{R}$ is the rotational part of the deformation gradient which can be computed using a polar decomposition, $\mathbb{1}$ is the identity matrix, $\| \cdot \|_F$ denotes the Frobenius norm, and μ and λ are the Lamé parameters. The first term results in an elastic response to stretching and compression in the individual spatial directions and the second term models resistance to volume changes of the material. The total deformation energy of a body can be found by computing the integral of the energy density over the rest-pose domain Ω of the body

$$E = \int_\Omega \Psi\left(\mathbf{F}(\mathbf{X})\right) d\mathbf{X}. \tag{3}$$

3.2 SPH Discretization

In order to evaluate the elastic energy and forces numerically, we apply a spatial discretization using SPH. To do so, we sample the undeformed body with n equally sized particles (for details see Section 7) which store the values of the field quantities at their respective positions. This allows us to interpolate an arbitrary field quantity $A(\mathbf{X})$ at position $\mathbf{X}_i$ as

$$A(\mathbf{X}_i) \approx \sum_{j \in \mathcal{N}_i^0} V_j A(\mathbf{X}_j) W(\parallel \mathbf{X}_i - \mathbf{X}_j \parallel, h), \tag{4}$$

where W is a radially symmetric kernel function with compact support, h is the smoothing length of the kernel and $\mathcal{N}_i^0$ is the set of particles which are inside the support radius of the kernel when it is centered at $\mathbf{X}_i$. The superscript 0 in $\mathcal{N}_i^0$ indicates that we take the rest-pose neighbors. The set of

neighbors in the current pose will be denoted as $\mathcal{N}_i$. Assuming all particles have the same mass, V_j is the rest-pose volume of the particle with index j which is computed as [Solenthaler et al. 2007]

$$V_i = \frac{1}{\sum_{j \in \mathcal{N}_i^0} W(\| \mathbf{X}_i - \mathbf{X}_j \|, h)}. \tag{5}$$

In the following we will use a shorter notation for the function arguments $A_i = A(\mathbf{X}_i)$ and $W_{ij} = W(\| \mathbf{X}_i - \mathbf{X}_j \|, h)$. Note that A_i and W_{ij} always mean that we evaluate the function using the rest-pose positions.

Derivatives can be computed by applying the differential operator to the kernel function such that the gradient becomes

$$\nabla A_i = \frac{\partial A_i}{\partial \mathbf{X}_i} \approx \sum_{j \in \mathcal{N}_i^0} V_j A_j \nabla W_{ij}. \tag{6}$$

With this approximation we can compute the deformation gradient of a particle with index i by plugging in the components of the deformed positions $\mathbf{x}$ for A. However, it was pointed out by Bonet and Lok [1999] that this approximation leads to erroneous deformation gradients because it is not 1^{st}-order consistent, i.e. it cannot guarantee that linear functions are reproduced exactly. Peer et al. [2018] demonstrated that this leads to artifacts so that the bodies cannot rotate. This problem can be overcome by constructing a 1^{st}-order consistent discretization of the SPH gradient operator [Bonet and Lok 1999]:

$$\nabla A_i \approx \sum_{j \in \mathcal{N}_i^0} V_j (A_j - A_i) \mathbf{L}_i \nabla W_{ij} \tag{7}$$

with the 3×3 *kernel correction matrix*

$$\mathbf{L}_i = \left(\sum_{j \in \mathcal{N}_i^0} V_j \nabla W_{ij} \otimes \mathbf{X}_{ji} \right)^{-1}, \tag{8}$$

where $\mathbf{X}_{ji} = \mathbf{X}_j - \mathbf{X}_i$ and $\otimes$ denotes the dyadic product of two vectors $\mathbf{a} \otimes \mathbf{b} = \mathbf{a}\mathbf{b}^T$. The derivation of the matrix can be found in the supplemental document. For the Lagrangian SPH formulation it only depends on the rest-pose positions such that it can be precomputed and stored at the beginning of the simulation.

The deformation gradient of particle i can be found by plugging in the components of the positions of the deformed state for A [Peer et al. 2018]:

$$\mathbf{F}_i = \sum_{j \in \mathcal{N}_i^0} V_j \mathbf{x}_{ji} \otimes \mathbf{L}_i \nabla W_{ij}, \tag{9}$$

where we introduced the short notation $\mathbf{x}_{ji} = \mathbf{x}_j - \mathbf{x}_i$. Note that in contrast to the common SPH formulation for fluids we are summing over the particle neighborhoods from the rest-pose configuration and the volume V_j and the corrected kernel gradient are also evaluated in the rest-pose.

The integral in the elastic energy (3) can be split up into integrals over the individual particle volumes and these can be approximated by applying a simple one point quadrature resulting in

$$\begin{aligned} E &= \sum_i \int_{V_i} \Psi\left(\phi(\mathbf{X})\right) d\mathbf{X} \\ &\approx \sum_i \mu_i V_i \parallel \mathbf{F}_i - \mathbf{R}_i \parallel_F^2 + \sum_i \frac{\lambda_i V_i}{2} \mathrm{tr}(\mathbf{R}_i^T \mathbf{F}_i - \mathbb{1})^2 \\ &= E^{\mathrm{s}} + E^{\mathrm{v}}. \end{aligned} \tag{10}$$

Here we introduced the notation E^{s} for the stretching part and E^{v} for the part of the elastic energy related to volume conservation.

3.3 Stretching Forces

In order to compute the stretching forces we rewrite the stretching energy E^{s} from Eq. (10) of one particle with index i as [Kugelstadt et al. 2018]

$$E_i^{\mathrm{s}}(\mathbf{x}) = \mu_i V_i \parallel \mathrm{vec}(\mathbf{F}_i) - \mathrm{vec}(\mathbf{R}_i) \parallel^2, \tag{11}$$

where we vectorize the 3×3 matrices by concatenating the columns into 9d vectors. Now the deformation gradient can be expressed as a matrix-vector product

$$\mathbf{D}_i \mathbf{x} = \mathrm{vec}(\mathbf{F}_i) = \sum_{j \in \mathcal{N}_i^0} V_j \begin{pmatrix} \mathbf{x}_{ji}(\mathbf{L}_i \nabla W_{ij})_1 \\ \mathbf{x}_{ji}(\mathbf{L}_i \nabla W_{ij})_2 \\ \mathbf{x}_{ji}(\mathbf{L}_i \nabla W_{ij})_3 \end{pmatrix}, \tag{12}$$

where $(\cdot)_k$ denotes the k-th component of a vector. More details about the computation of matrix $\mathbf{D}_i$ can be found in the supplemental document.

With this notation we can write the sum for the total stretching energy as a matrix-vector product as well

$$E^{\mathrm{s}}(\mathbf{x}) = \sum_i \mu_i V_i \parallel \mathbf{D}_i \mathbf{x} - \mathrm{vec}(\mathbf{R}_i) \parallel^2 = \parallel \mathbf{K}^{1/2}(\mathbf{D}\mathbf{x} - \mathbf{r}) \parallel^2, \tag{13}$$

where the matrix $\mathbf{D} \in \mathbb{R}^{9n \times 3n}$ contains all matrices $\mathbf{D}_i$ stacked on top of each other, $\mathbf{r}$ contains all vectorized rotations concatenated into one vector, and $\mathbf{K} = \mathrm{diag}(\mu_1 V_1 \mathbb{1}_{9\times 9}, \cdots, \mu_n V_n \mathbb{1}_{9\times 9})$. Now, we can compute the forces as

$$\mathbf{f}^{\mathrm{s}} = -\frac{\partial E^{\mathrm{s}}}{\partial \mathbf{x}} = -2\mathbf{D}^T \mathbf{K} \mathbf{D}\mathbf{x} + 2\mathbf{D}^T \mathbf{K}\mathbf{r}. \tag{14}$$

Because the matrix $\mathbf{D}$ only depends on the rest-pose positions it will be constant during the simulation (see supplemental document).

3.4 Volume Conserving Forces

Finally, we have to compute the volume conserving forces by taking the negative gradient of E^{v} w.r.t. $\mathbf{x}_i$:

$$\mathbf{f}_i^{\mathrm{v}} = \sum_j V_j \frac{\partial \Psi_j^{\mathrm{v}}}{\partial \mathbf{x}_i} = V_i \frac{\partial \Psi_i^{\mathrm{v}}}{\partial \mathbf{F}_i} : \frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_i} + \sum_{j \in \mathcal{N}_i^0} V_j \frac{\partial \Psi_j^{\mathrm{v}}}{\partial \mathbf{F}_j} : \frac{\partial \mathbf{F}_j}{\partial \mathbf{x}_i}, \tag{15}$$

where Ψ_i^{v} denotes the volume conserving part of the energy density in Eq. (2). The derivative of the energy density w.r.t. the $\mathbf{F}$ is known as the first Piola-Kirchhoff stress tensor [Sifakis and Barbic

2012]

$$\mathbf{P}_i = \frac{\partial \Psi_i^{\mathrm{v}}}{\partial \mathbf{F}_i} = \lambda_i \mathrm{tr}(\mathbf{R}_i^T \mathbf{F}_i - \mathbb{1})\mathbf{R}_i. \tag{16}$$

The derivatives of the deformation gradient result in the third-order tensors

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_i} = -\sum_{j \in \mathcal{N}_i^0} V_j \mathbb{1} \otimes \mathbf{L}_i \nabla W_{ij}, \qquad \frac{\partial \mathbf{F}_j}{\partial \mathbf{x}_i} = V_i \mathbb{1} \otimes \mathbf{L}_j \nabla W_{ji}. \tag{17}$$

Using the relation $\mathbf{A} : \mathbb{1} \otimes \mathbf{b} = \mathbf{A}\mathbf{b}$ for a 3×3 matrix $\mathbf{A}$ and a 3d vector $\mathbf{b}$ results in the forces [Ganzenmüller 2015]

$$\mathbf{f}_i^{\mathrm{v}} = \sum_{j \in \mathcal{N}_i^0} V_i V_j (\mathbf{P}_i \mathbf{L}_i \nabla W_{ij} - \mathbf{P}_j \mathbf{L}_j \nabla W_{ji}). \tag{18}$$

4 ZERO-ENERGY MODES

The one point quadrature approximation used in Eq. (10) has the advantages that it is simple and it leads to very efficient computations. However, it can cause visual artifacts due to the so called zero-energy modes [Ganzenmüller 2015], a similar problem to the hourglass modes known in FEM simulations. Using only one quadrature point for each particle neighborhood means that we take only one constant deformation gradient to describe the deformation of the particle neighborhood. The 9 components of $\mathbf{F}_i$ can only account for linear deformation. But the actual neighborhood usually consists of 30-40 particles such that it can in principle take on more complicated non-linear deformations. In practice this happens, e.g. due to strong deformations which are induced by boundary conditions or collisions. Since the non-linear part of the deformation is not captured by the deformation gradient, it does not contribute to the elastic energy (hence the name zero-energy modes) and prevents that the particles return into their rest-pose.

One way to avoid zero-energy modes is to use more quadrature points. However, it is not trivial to find out where the additional quadrature points have to be placed when the particle sampling in the rest pose is irregular as pointed out by Ganzenmüller [2015]. Moreover, the additional quadrature points drastically increase the computation costs.

4.1 Implicit Zero-Energy Mode Suppression

An alternative solution is to compute additional forces in order to suppress the zero-energy modes as suggested by Ganzenmüller [2015]. Inspired by his method we derive additional penalty forces to counteract the zero-energy modes without the need for additional quadrature points. In contrast to Ganzenmüller's explicit method which is only conditionally stable, we propose an implicit approach.

Our goal is to penalize the non-linear part of the deformations so that we obtain deformations which can be accurately captured by the deformation gradients of the particles. The non-linear part can be found by subtracting the total deformation from the linear deformation. We consider the line segment which is formed by two particles with indices i and j. To apply the linear deformation we multiply its rest pose positions with the deformation gradient while the non-linear deformation is given by the relative position in the current pose. This results in the error

$$\mathcal{E}_{ij} = \mathbf{F}_i \mathbf{X}_{ij} - \mathbf{x}_{ij}. \tag{19}$$

We define a quadratic energy to penalize this error for each particle by computing the SPH average of $\| \mathcal{E}_{ij} \|^2$ using Eq. (4). To obtain the total energy we take the sum over all particles

$$E^{\text{ze}} = \frac{\alpha}{2} \sum_i \mu_i V_i \sum_{j \in \mathcal{N}_i^0} V_j \frac{\| \mathcal{E}_{ij} \|^2}{\| \mathbf{X}_{ij} \|^2} W_{ij}, \tag{20}$$

where α is a user defined stiffness parameter. We divide $\mathcal{E}_{ij}$ by $\| \mathbf{X}_{ij} \|$ so that it becomes a dimensionless strain and we multiply with $\mu_i V_i$ so that the correction forces are proportional to the elastic forces and the α parameter becomes independent of the material stiffness.

We remark that our method is not just the result of direct implicit integration of the force proposed by Ganzenmüller. In order to conserve angular momentum, Ganzenmüller argues that it was necessary to project the error $\mathcal{E}_{ij}$ onto the line between the two particles. Since we instead define a *translation-invariant and rotation-invariant penalty energy* based on $\mathcal{E}_{ij}$, we obtain a force expression that does not require projection of the error. Because of Noether's theorem the rotational invariance means that the penalty forces derived from this energy will preserve angular momentum. Linear momentum is preserved as well since the energy is translation invariant. This can be easily seen because only position differences $\mathbf{x}_{ij}$ are considered which are invariant to translations of the particles.

To see that the energy is rotation invariant, consider the case when the current positions $\mathbf{x}_i$ of all particles are rotated with the rotation matrix $\mathbf{R}$. Then the deformation gradient becomes $\mathbf{RF}_i$ as we see in Eq. (9) and the error becomes $\mathbf{R}\mathcal{E}_{ij}$ for all i and j. Because we take the squared norm of the error to obtain the energy in Eq. (20) and the norm is rotation invariant the energy is rotation invariant as well.

4.2 Zero-Energy Mode Forces

To derive the forces for the zero-energy mode suppression we also rewrite the corresponding energy as a matrix-vector product. First, we rewrite the error vectors $\mathcal{E}_{ij}$ from Eq. (19) as a product of a constant matrix with the positions $\mathbf{x}$:

$$\mathcal{E}_{ij} = \sum_{k \in \mathcal{N}_i^0} V_k \mathbf{x}_{ki} (\mathbf{L}_i \nabla W_{ik})^T \mathbf{X}_{ij} - \mathbf{x}_{ij} = \mathbf{H}_{ij} \mathbf{x}. \tag{21}$$

Details about the computation of the matrix $\mathbf{H}_{ij}$ can be found in the supplemental document.

With this notation we can rewrite the sum over all neighbors in Eq. (20) as a matrix vector product

$$E^{\text{ze}} = \frac{1}{2} \sum_i \sum_{j \in \mathcal{N}_i^0} \mathbf{x}^T \mathbf{H}_{ij}^T \tilde{\mathbf{K}}_{ij} \mathbf{H}_{ij} \mathbf{x} = \frac{1}{2} \sum_i \mathbf{x}^T \mathbf{H}_i^T \tilde{\mathbf{K}}_i \mathbf{H}_i \mathbf{x}, \tag{22}$$

by concatenating the $\mathbf{H}_{ij}$ matrices for all neighbors into one matrix $\mathbf{H}_i = (\mathbf{H}_{i1}^T, ..., \mathbf{H}_{in_i}^T)^T$ where n_i is the number of neighbors of particle i. The constant factors are encapsulated in the diagonal matrices $\tilde{\mathbf{K}}_{ij} = \alpha \mu_i V_i V_j W_{ij} / \| \mathbf{X}_{ij} \|^2 \, \mathbb{1}$ which are combined into $\tilde{\mathbf{K}}_i = \text{diag}(\tilde{\mathbf{K}}_{i1}, ..., \tilde{\mathbf{K}}_{in_i})$. The same can be done with the sum over all particles i so that we get

$$E^{\text{ze}}(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{H}^T \tilde{\mathbf{K}} \mathbf{H} \mathbf{x}, \tag{23}$$

with $\mathbf{H} = (\mathbf{H}_1^T, ..., \mathbf{H}_n^T)^T$ and $\tilde{\mathbf{K}} = \text{diag}(\tilde{\mathbf{K}}_1, ..., \tilde{\mathbf{K}}_n)$. The forces are computed as the negative gradient of the energy:

$$\mathbf{f}^{\text{ze}} = -\frac{\partial E_{ZE}}{\partial \mathbf{x}} = -\mathbf{H}^T \tilde{\mathbf{K}} \mathbf{H} \mathbf{x}. \tag{24}$$

Note that the matrices $\mathbf{H}$ and $\tilde{\mathbf{K}}$ only depend on the rest-pose positions and the stiffness parameters such that they are constant during the simulation (see supplemental document).

5 TIME DISCRETIZATION

The elastic energy $E = E^{\mathrm{s}} + E^{\mathrm{v}} + E^{\mathrm{ze}}$ consists of three terms: stretching E^{s}, volume conservation E^{v} and the zero-energy mode penalty E^{ze}. We make the commonly used assumption that the rotation is computed at the beginning of each time step and kept constant during the step. Then, all energy terms are quadratic in $\mathbf{x}$ such that we will obtain forces that are linear in the positions and result in the following linear system per backwards Euler step:

$$\mathbf{M}\Delta\mathbf{v} = \Delta t \left(\mathbf{f}^{\mathrm{ext}} + \mathbf{f}^{\mathrm{s}}(\mathbf{x}^{n+1}) + \mathbf{f}^{\mathrm{v}}(\mathbf{x}^{n+1}) + \mathbf{f}^{\mathrm{ze}}(\mathbf{x}^{n+1})\right), \tag{25}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^{n} + \Delta t(\mathbf{v}^{n} + \Delta\mathbf{v}). \tag{26}$$

Here, $\mathbf{M} = \mathrm{diag}(m_1\mathbb{1}, m_2\mathbb{1}, ..., m_n\mathbb{1})$ is a diagonal mass matrix, $\mathbf{v}$ contains the velocities, $\mathbf{x}$ the positions and $\mathbf{f}^{\mathrm{ext}}$ the external forces of all particles, Δt is the time step size, and the superscript n / $n+1$ denotes the current / next time step. Solving for the velocity changes $\Delta\mathbf{v}$ is convenient because for static particles they are 0. This means that some particles can be fixed by removing the corresponding equations from the linear system.

5.1 Operator Splitting

In the context of corotated FEM Kugelstadt et al. [2018] observed that the stretching term has a constant stiffness matrix when the Lamé parameters are constant. The stiffness matrix of the volume conserving term changes in every time step because of the changing rotation matrices. This is also true when we use our SPH discretization since the stretching force in Eq. (14) linearly depends on $\mathbf{x}$ while the volume conserving force in Eq. (18) also depends on the rotation matrix. Our zero-energy mode term also has a constant stiffness matrix (cf. Eq.(24)). This observation can be used to speed up the simulation by applying operator splitting and solving the terms with the constant system matrix with a direct solver by using a precomputed Cholesky factorization. The remaining volume conservation term is solved with a conjugate gradient (CG) solver. Our results show that this approach is much faster than solving the whole system with CG.

First, we solve the system without considering the volume conservation term to determine the velocity updates $\Delta\mathbf{v}$

$$\mathbf{M}\Delta\mathbf{v} = \Delta t \left(\mathbf{f}^{\mathrm{ext}} + \mathbf{f}^{\mathrm{s}}(\tilde{\mathbf{x}} + \Delta t\Delta\mathbf{v}) + \mathbf{f}^{\mathrm{ze}}(\tilde{\mathbf{x}} + \Delta t\Delta\mathbf{v})\right), \tag{27}$$

which are used to find the intermediate velocities $\mathbf{v}^* = \mathbf{v}^n + \Delta\mathbf{v}$. Here $\tilde{\mathbf{x}} = \mathbf{x}^n + \Delta t\mathbf{v}^n$. In a next step we solve the linear system for the volume conserving term to obtain the new velocities $\mathbf{v}^{n+1} = \mathbf{v}^* + \Delta\mathbf{v}^*$:

$$\mathbf{M}\Delta\mathbf{v}^* = \Delta t\mathbf{f}^{\mathrm{v}}(\mathbf{x}^* + \Delta t\Delta\mathbf{v}^*), \tag{28}$$

with $\mathbf{x}^* = \mathbf{x}^n + \Delta t\mathbf{v}^*$. Note that before solving the second linear system (28) we recompute the required rotation matrices using the updated positions $\mathbf{x}^*$ in order to consider the changes due to the solve of the first system (27) (cf. Algorithm 1).

Stretching and Zero-Energy Mode Term. Plugging the forces into Eq. (27) yields the linear system

$$\left(\mathbf{M} + 2\Delta t^2\mathbf{D}^T\mathbf{K}\mathbf{D} + \Delta t^2\mathbf{H}^T\tilde{\mathbf{K}}\mathbf{H}\right)\Delta\mathbf{v} = \Delta t\left(\mathbf{f}^{\mathrm{ext}} - 2\mathbf{D}^T\mathbf{K}(\mathbf{D}\tilde{\mathbf{x}} - \mathbf{r}) - \mathbf{H}^T\tilde{\mathbf{K}}\mathbf{H}\tilde{\mathbf{x}}\right). \tag{29}$$

Here we see that the system matrix is constant as long as the time step size, the Lamé parameters and the parameter α do not change. This means that we can precompute its Cholesky factorization so that we can solve the linear system very efficiently at runtime with linear complexity in the number of non-zeros in the triangular factor by using backward and forward substitution. Note that

the components x, y and z are independent of each other. Hence, the system can be solved in parallel per component and per body. Because the matrix $\mathbf{H}$ has the same sparsity pattern as $\mathbf{D}$, adding the zero-energy mode suppression does not add non zeroes to the Cholesky factorization, and therefore it does not negatively influence the time to solve the linear system. We will discuss later how we can use adaptive time stepping even though a fixed time step is used in the precomputed factorization.

Volume Conservation Term. In order to construct the linear system for the volume conservation term, we split the volume conserving force in Eq. (18) into a constant term $\mathbf{f}_i^c$ and a linear term $\mathbf{f}_i^l(\mathbf{x})$ so that $\mathbf{f}_i^v(\mathbf{x}) = \mathbf{f}_i^c + \mathbf{f}_i^l(\mathbf{x})$:

$$\mathbf{f}_i^c = \sum_{j \in \mathcal{N}_i^0} V_i V_j (3\lambda_i \mathbf{R}_i \mathbf{L}_i \nabla W_{ij} - 3\lambda_j \mathbf{R}_j \mathbf{L}_j \nabla W_{ji}),$$

$$\mathbf{f}_i^l = \sum_{j \in \mathcal{N}_i^0} V_i V_j (\lambda_i \mathrm{tr}(\mathbf{R}_i^T \mathbf{F}_i) \mathbf{R}_i \mathbf{L}_i \nabla W_{ij} - \lambda_j \mathrm{tr}(\mathbf{R}_j^T \mathbf{F}_j) \mathbf{R}_j \mathbf{L}_j \nabla W_{ji}).$$

The force $\mathbf{f}_i^c$ is constant since we keep the rotation constant during the step. The linear system for backwards Euler from Eq. (28) becomes

$$\mathbf{M}\Delta\mathbf{v}^* - \Delta t \mathbf{f}^l(\Delta t \Delta \mathbf{v}^*) = \Delta t(\mathbf{f}^l(\mathbf{x}^*) + \mathbf{f}^c). \tag{30}$$

The linear part of the forces depends on the rotation matrices so that the system matrix changes in every time step. Hence, we cannot use a precomputed factorization. Because it is very expensive to compute the Cholesky factorization at runtime we will solve this system with an iterative CG solver. We can avoid building the full system matrix because CG only requires products of the matrix with a given vector. These can be computed efficiently by evaluating the linear part of the forces. To do so we iterate twice over all particles and their neighbors. In the first loop we compute the stress tensors for all particles and in the second one we evaluate the forces. To speed up the convergence we use the velocity increments $\Delta\mathbf{v}^*$ from the last time step as an initial guess to warm-start our matrix-free CG solver.

6 COLLISIONS AND SOLID-FLUID COUPLING

We integrate the solid solver into an SPH fluid simulation framework as shown in Algorithm 1. For our experiments we used the DFSPH method [Bender and Koschier 2017] but it would work in the same way for other methods like IISPH [Ihmsen et al. 2014] or position-based fluids [Macklin and Müller 2013]. We have two sets of particles, one for the fluid and one for the solids. The coupling is achieved by including the solid particles into the pressure solver which enforces a constant particle density and a divergence-free velocity field. This prevents fluid particles from entering the solids and accounts for collisions and self-collisions of the solids. We first apply external and elastic forces as well as other non-pressure forces and perform the pressure solve at the end of the time step such that we have a collision free state after the step.

It is common practice that SPH fluid solvers use adaptive time steps based on a CFL condition to prevent artifacts due to collocated particles. However, the system matrix of the solid solver in Eq. (29) depends on the time step so that we cannot change Δt in every simulation step because computing the Cholesky factorization is expensive. Therefore, we use different time steps for the solid and the fluid which was proposed by Peer et al. [2018] and is similar to the asynchronous SPH approach of Reinhardt et al. [2017]. While the step size Δt^{el} of the elasticity solver stays constant, the time step Δt of the pressure solver is determined using the CFL condition $\Delta t = 0.4d/\|\mathbf{v}\|_\infty$, where d is the particle diameter. To implement this we transform the velocity changes determined by the elasticity solver into accelerations and then interpolate the final velocities using the time

Algorithm 1 Coupled solid-fluid solver

1: perform neighborhood search
2: apply non-pressure forces to get $\mathbf{v}^n$
3: $\tilde{\mathbf{x}} := \mathbf{x}^n + \Delta t^{\text{el}} \mathbf{v}^n$
4: compute rotations $\mathbf{R}_i$ using $\tilde{\mathbf{x}}$
5: Solve Eq. (29) for $\Delta\mathbf{v}$ ▷ solve stretching and ZE terms using precomp. Cholesky fact.
6: $\mathbf{v}^* := \mathbf{v}^n + \Delta\mathbf{v}$
7: $\mathbf{x}^* := \mathbf{x}^n + \Delta t^{\text{el}} \mathbf{v}^*$
8: compute rotations $\mathbf{R}_i$ using $\mathbf{x}^*$
9: Solve Eq. (30) for $\Delta\mathbf{v}^*$ ▷ solve volume conservation term using matrix-free CG
10: $\mathbf{v}^{**} := \mathbf{v} + \Delta\mathbf{v}^*$
11: $\mathbf{a} = (\mathbf{v}^{**} - \mathbf{v}^n)/\Delta t^{\text{el}}$
12: $\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \mathbf{a}$
13: pressure solve using DFSPH

step size Δt (see Algorithm 1, lines 11 & 12). Figure 1 shows complex experiments which were performed with this time stepping scheme.

7 SAMPLING

In order to simulate elastic solids with SPH we need to sample the solid volume with particles. We assume that the solids are given as closed surface meshes or as signed distance fields (SDF). When they are given as meshes we convert them into SDFs using the methods described by Koschier et al. [2017]. Then we create a regular sampling by placing a regular grid inside the bounding box of the object. The grid spacing is equal to the particle radius r. For each grid node which is inside the solid geometry we create a particle with radius r. The inside or outside test can be easily performed by querying the SDF.

A regular sampling can be used to simulate solids, however, it can lead to some problems. First, fine surface details of the input geometry are not well represented by this type of sampling, especially when the sampling is coarse. Moreover, fine features of the geometry may lead to co-linear or co-planar particle neighborhoods, which are problematic because their kernel correction matrices in Eq. (8) become singular and cannot be inverted.

In the following we propose a novel sampling technique that significantly alleviates these problems. It is inspired by the blue noise sampling method presented by Jiang et al. [2015b] which takes an initially dense regular sampling and refines it using an explicit SPH pressure solver. In our work we use an implicit pressure solver for the sampling to avoid instabilities. The pressure solve makes sure that we get a particle distribution which is beneficial for the pressure solver that is used in the actual simulation. Jiang et al. [2015b] apply correction and cohesion forces to make the surface particles coherent with the interior particles. In our work we use a different cohesion force which aims at obtaining a homogeneous particle distribution. Moreover, we add an additional adhesion force which pulls particles to the surface to get a better representation of fine features of the geometry. Finally, this gives us better results than the method of Jiang et al. [2015b] as we show in Section 9.

Starting with a dense regular sampling we refine it iteratively by applying the following position changes $\Delta\mathbf{x}_i$:

$$\Delta\mathbf{x}_i = \Delta\mathbf{x}_i^{\text{pressure}} + \Delta\mathbf{x}_i^{\text{cohesion}} + \Delta\mathbf{x}_i^{\text{adhesion}}. \tag{31}$$

The pressure term enforces a constant particle density ρ_i which is computed as

$$\rho_i = \sum_{j \in \mathcal{N}_i} m W(\mathbf{x}_{ij}), \tag{32}$$

where m is the mass which is equal for all particles. The displacements are computed using the constant density solve of the DFSPH method [Bender and Koschier 2017]

$$\Delta \mathbf{x}_i^{\text{pressure}} = -\sum_{j \in \mathcal{N}_i} m_j \left(\frac{\kappa_i}{\rho_i} + \frac{\kappa_j}{\rho_j} \right) \nabla W(\mathbf{x}_{ij}), \tag{33}$$

where

$$\kappa_i = \frac{\rho_i (\rho_i - \rho_0)}{\| \sum_{j \in \mathcal{N}_i} m_j \nabla W(\mathbf{x}_{ij}) \|^2 + \sum_{j \in \mathcal{N}_i} \| m_j \nabla W(\mathbf{x}_{ij}) \|^2}. \tag{34}$$

The cohesion displacement is similar to a spring force which enforces that the particles in each neighborhood have the same distance of one particle diameter d to each other

$$\Delta \mathbf{x}_i^{\text{cohesion}} = -\beta \sum_{j \in \mathcal{N}_i} V_j \left(\|\mathbf{x}_i - \mathbf{x}_j\| - d \right) \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} W(\mathbf{x}_{ij}), \tag{35}$$

where β is the cohesion coefficient. This displacement improves the particle distribution especially at the free surface, where the pressure solver cannot find an optimal solution due to the particle deficiency problem.

If a particle $\mathbf{x}_i$ is closer to the surface than the support radius, we compute the closest point on the surface $\mathbf{x}_s$ and add an attractive displacement

$$\Delta \mathbf{x}_i^{\text{adhesion}} = -\gamma V_i \left(\mathbf{x}_i - \mathbf{x}_s \right) W(\mathbf{x}_i - \mathbf{x}_s), \qquad \mathbf{x}_s = \mathbf{x}_i - \Phi(\mathbf{x}_i) \frac{\nabla \Phi(\mathbf{x}_i)}{\|\nabla \Phi(\mathbf{x}_i)\|}, \tag{36}$$

where γ is the adhesion coefficient and $\Phi(\mathbf{x}_i)$ is the SDF with the convention that distances inside of the solid are negative. The adhesion pulls particles which are close to the surface towards the surface such that they align with fine features of the geometry.

In each iteration the positions are updated using the total displacements $\Delta \mathbf{x}$

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \eta \frac{0.4d}{\|\Delta \mathbf{x}\|_\infty} \Delta \mathbf{x}, \tag{37}$$

where η is the CFL factor for which we use 0.25 as default value. After each iteration we apply collision response displacements which prevent the particles from leaving the solid volume. Using the SDF we project particles which are outside of the volume or closer to the surface than the particle radius back such that the particles are exactly touching the surface from inside.

When the position change is small enough, we can stop the iterative process. Our SPH refinement results in a high quality particle distribution which considerably reduces the occurrence of degenerate configurations and particles with low numbers of neighbors.

8 MESH SKINNING

For rendering we want to skin a high resolution surface mesh to the particles of the deformable solid. This can be performed during the simulation or as a post-processing step. In a preprocessing step we determine the neighboring SPH solid particles for each vertex of the visualization mesh. We also precompute the Shepard filter factors s_k for each vertex $\mathbf{x}_k$ of the visualization mesh

$$s_k = \frac{1}{\sum_{j \in \mathcal{N}_k^0} V_j^0 W(\mathbf{X}_{kj})}. \tag{38}$$

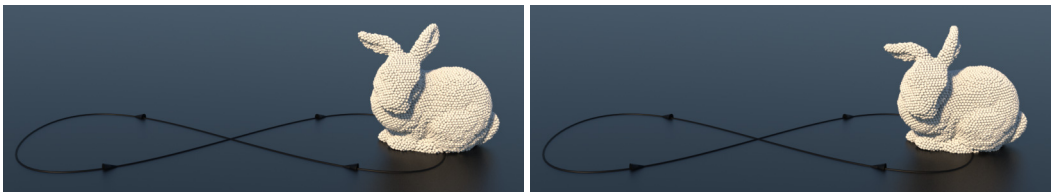


Fig. 2. Comparison between our method (left) and the method by Peer et al. (right) in an experiment where the bottom particles of a sampled bunny follow a prescribed motion. The material behavior obtained with both methods is comparable.

Similar to the kernel correction matrix they are needed to obtain a 0^{th}-order consistent SPH interpolation of the function itself instead of the gradient [Reinhardt et al. 2019]. For every frame of the simulation we can interpolate the vertex positions of the simulation mesh from the deformed SPH particles as

$$\mathbf{x}_k = s_k \sum_{j \in \mathcal{N}_k^0} V_j \left(\mathbf{F}_j \mathbf{X}_{ij} + \mathbf{x}_j \right) W(\mathbf{X}_{ij}). \tag{39}$$

This interpolation can be evaluated very fast and it leads to much better visual results than surface reconstructions with marching cubes.

9 RESULTS

In this section we discuss experiments to validate our method and to compare it with existing solutions. All contributions of this publication were implemented in the open-source SPH framework SPlisHSPlasH [Bender et al. 2021]. The solid-fluid solver relies on the already implemented *Divergence-Free SPH* method [Bender and Koschier 2017] as the pressure solver, *Volume Maps* [Bender et al. 2020] for the boundary handling, a *Micropolar* method [Bender et al. 2019] to counteract numerical damping in the fluid, and the *Analytic Polar Decomposition* method by Kugelstadt [2018] to extract the rotations from the deformation gradient.

We used Eigen [Guennebaud et al. 2021] for all the required linear algebra functionalities, including the Cholesky factorization and the matrix-free CG solver. Important bottlenecks in all the methods have been optimized using single precision AVX2 instructions whenever possible.

While other methods can be parallelized across particles, this is not the case for the Cholesky solve in our method since it requires backward and forward substitution of the corresponding triangular matrices. Despite those operations being hard to parallelize beyond using SIMD instructions, we can still use parallelization to great extent by solving the linear systems concurrently per body and per dimension (x, y and z), since they are independent of each other.

Comparison with Peer et al. [2018]. We simulated a moving elastic bunny that follows a prescribed trajectory and compare our method to the method by Peer et al. (see Figure 2). Unless otherwise specified, the bunny is discretized with 32k particles of radius 0.0127 m, using a Young's modulus of 1 MPa, a Poisson's ratio of 0.33 and 2 ms time step size. All tests related to this experiment were run on an Intel Core i7-7700K Processor with 4 physical cores and 8 threads at 4.20GHz. The experiments always use all available cores.

In Table 1 we can see the benchmark results with varying particle sampling resolution. The times correspond to the average runtime in milliseconds required per time step. We observe that our method is superior in terms of performance to the method proposed by Peer et al. for all the resolutions tested. The substantial speedup factor is a direct consequence of using a precomputed

Particles	Ours	Peer et al.	Speedup	nnzs $\times 10^6$	Memory [MB]
4208	6.9	144	20.9x	2.52	19.28
8208	19.2	362	18.9x	7.59	58.01
16193	61.5	895	14.9x	27.80	212.16
31749	179	2102	11.7x	86.41	659.34
64055	507	5994	11.8x	284.88	2173.74

Table 1. Comparison of the moving bunny simulation runtimes and memory requirements with varying number of particles. The table shows the average computation times per time step (in ms), the speedup factors, the number of non zeros of the lower triangular matrix resulting from the Cholesky factorization and the space needed to store it.

factorization instead of an iterative solver for the most time consuming step of the simulation: the implicit solve of the stretching forces and the zero-energy mode correction. However, storing the Cholesky factorization entails larger memory requirements than methods that use a pure matrix-free iterative solver. The fact that the memory requirements scale superlinearly with the number of particles is due to a higher fill-in of the triangular matrix obtained from the Cholesky decomposition, which is also the reason behind the decreasing speedup factor. In any case, the runtime gap is still very large, ranging from a speedup of 20.9x for the coarsest model to 11.8x for the highest detailed one. Also, it is important to remark that only one factorization per unique object has to be stored since the decomposition is computed at rest pose and it is invariant to rigid translations and rotations.

For the sake of the comparison we followed the common practice in SPH of using a compact support radius that results in approximately 30 neighbors per particle. However, we observed that elastic solids can also be simulated with less neighbors. By using the closest 10 neighbors per particle instead of the full neighborhood, we reduced our memory requirements by a factor of 2.5, while obtaining visually identical results. Naturally, the runtime also improved, by a factor of 2, but so does the method by Peer et al., therefore we do not include a comparison table for that case.

Table 1a shows the benchmark results for the moving bunny simulation with varying Young's modulus. There is a strong correlation between increasing material stiffness and the speedup factor with respect to the method by Peer et al. This is can also be attributed to our direct Cholesky solve which performs a fixed amount of operations. An iterative linear solver instead needs more iterations to converge when the material stiffness is increased due to a larger condition number of the system. Table 1b contains the results for the moving bunny simulation using different time step sizes, which shows the same trend observed in the experiment with varying Young's modulus. When an iterative solver is used, employing larger time steps will require more iterations due to poorer conditioning and a worse initial guess. Neither of those issues are relevant when a direct solver is used. These two experiments highlight an important property of our solver, which is that it has a very consistent runtime without sacrificing accuracy, in contrast to methods which use iterative solvers.

Zero-energy mode correction. To showcase the importance of our implicit zero-energy mode correction, we carried out an experiment where the particles of a walrus model are randomized to induce an initial deformed state (see Figure 3). We used 50k particles with a radius of 0.025 m, a Young's modulus of 2.5 MPa, a Poisson's ratio of 0.33, a time step size of 1 ms, $\alpha = 1$ in the case of zero-energy mode correction and $\alpha = 0$ in the case without correction. By employing the zero-energy mode

(a) Varying values of Young's modulus

E [MPa]	**Ours**	**Peer et al.**	**Speedup**
0.1	123	805	6.5x
0.5	155	1583	10.2x
1.0	179	2102	11.7x
5.0	255	3878	15.2x
10.0	338	4930	14.6x

(b) Varying time step sizes

Δt [ms]	**Ours**	**Peer et al.**	**Speedup**
1	134	1152	8.6x
2	179	2102	11.7x
5	301	4629	15.4x
10	479	8712	18.2x
15	433	12894	29.8x

Table 2. Comparison of the moving bunny simulation runtimes (in ms).

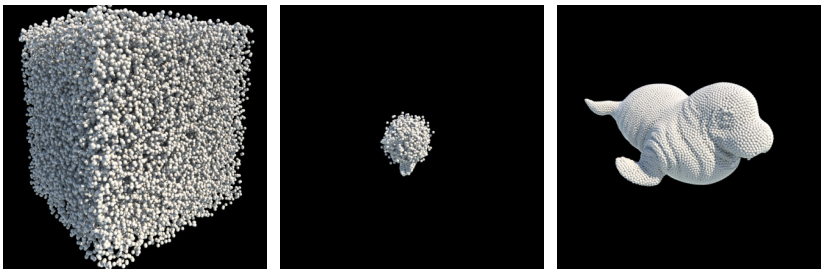


Fig. 3. Randomized particles experiment. From left to right: Initial deformed configuration, no zero-energy mode control, zero-energy mode control. The original shape is recovered using our zero-energy mode correction.

correction, the original shape can be recovered even after such strong deformation. With correction enabled, the elasticity solve required an average of 75.9 ms per time step. This measurement and all subsequent measurements were obtained using an Intel Core i9-9900KF processor with 8 physical cores and 16 threads.

Our experience is that for modest values of α, the impact the zero-energy mode correction has on the dynamics has little bearing on the visual results. We illustrate this with a cantilever beam experiment in the supplemental video. Four cantilever beams attached to a wall swing under gravity, with $\alpha = 0$, $\alpha = 0.01$, $\alpha = 0.1$ and $\alpha = 1$, respectively. Although, we can observe a small change in the swing frequency for increasing α, the effect is subtle.

To further demonstrate the added stability provided by our implicit zero-energy mode correction, we compare it to the explicit method proposed by Ganzenmüller [2015] and also used by Peer et al. [2018]. For this, we used a twisted beam test (see Figure 4) which provides both large deformations and high stresses. The beam is composed of 18k particles with radius 0.025 m using a Young's modulus of 10 MPa and a Poisson's ratio of 0.2. We chose the value of α (for our method) and α' (for the method of Ganzenmüller) that gave the best simulation results. The simulation is run with 1 ms time steps. Our simulation required an average of 15.0 ms per time step for the elasticity solve. The elasticity solve using the method of Peer et al. (using the zero-energy mode correction of Ganzenmüller) took on average 58.6 ms per time step. While the explicit method by Ganzenmüller fails to provide results without artifacts in such conditions, our implicit method produces stable simulations with $\alpha = 10$.

Sampling. To showcase our novel sampling method, we first compare it with the method proposed by Jiang et al. [2015b] for two complex models (see Figure 5). For our sampling, we used a cohesion

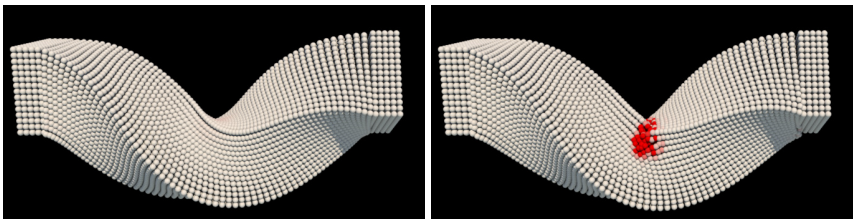
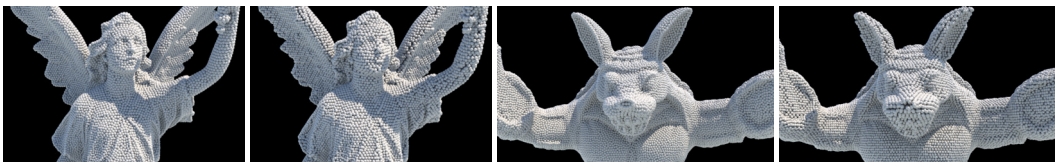
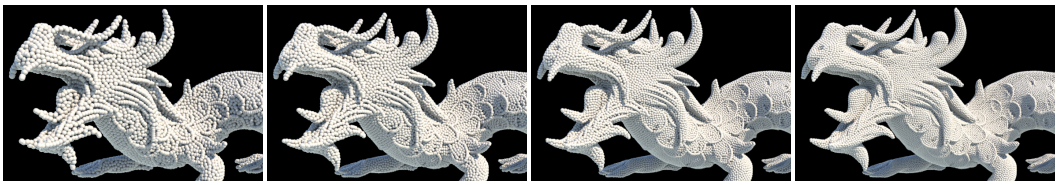


Fig. 4. Twisted beam experiment. Our implicit zero-energy mode correction successfully produces a stable simulation (left) while the explicit handling proposed by Ganzenmüller does not (right). Particles colored in red experience very high velocity.



(a) Our method (b) Jiang et al. (c) Our method (d) Jiang et al.

Fig. 5. Comparison between our sampling method and the blue noise sampling method proposed by Jiang et al. [2015b]. Our method generates a higher quality sampling for both the sculpture model with 215k particles and the Standford armadillo model with 240k particles.



(a) 26k particles (b) 52k particles (c) 104k particles (d) 208k particles

Fig. 6. Dragon model discretized with our sampling technique using different amounts of particles.

parameter of $\beta = 4$ and an adhesion parameter of $\gamma = 0.2$. It is immediately apparent that our sampling technique achieves a more densely packed and higher quality particle distribution, especially in complex areas like the armadillo's nose and the sculpture's wings. In the accompanying video we compare our sampling technique with a regular sampling in simulations of an armadillo model with different resolutions. When using regular sampling, artifacts occur at the ears and feet due to degenerate particle configurations. The comparison shows that our method helps to reduce the occurrence of such configurations and particles with low neighbor count. However, while our method works well in practice and often avoids such configurations, there is no guarantee that they can be avoided in every case.

In a second experiment we show our sampling technique for different number of particles with the dragon model (see Figure 6). Even for lower resolutions we can see how the thin features, such as the dragon eye and the scales, are preserved.

Walrus funnel. In this experiment, we pushed eight walrus models through a tight funnel (see Figure 1 left) to demonstrate the stability and robustness of our method. The high pressure is handled by our simulator in several instances: in the interior of the elastic bodies, on the contact area between them and also in the interaction of the elastic bodies with the walls of the funnel and with the water. The runtime of the stretch solver remains constant during the simulation regardless of the strong deformations. Our proposed mesh skinning was used to generate the surface triangle meshes from the particle data, which also proved to work with large deformations.

We used 30k particles for each of the eight walrus models and 1.2M fluid particles, all of them with a radius of 0.025 m. For the material parameters, we used a Young's modulus of 0.4 MPa, a Poisson's ratio of 0.2, a density of 800 kg/m^3 for the elastic objects and a density of 1000 kg/m^3 for the fluid. For this scene the elasticity solve required an average of 310.1 ms per time step, consisting of 231.9 ms for the stretching part and 78.2 ms for the volume part. In comparison, the DFSPH pressure solve required an average of 877.8 ms per time step.

Water park. In our final experiment (see Figure 1 right) we demonstrate the coupling of deformable bodies, rigid bodies and fluids (including a highly viscous fluid) using the SPH framework. In addition, we show that our method is capable of simulating many objects in a single simulation. A total of 10 deformable and 4 rigid objects are emitted sequentially, slide down waterslides and drop down the stairs into a bowl, where they interact with water and a highly viscous fluid. Since there are only 2 unique deformable models in the scene, we also only have to compute and store two unique factorizations. We used a total of 833k particles for the fluid and a total of 252k particles for the deformable objects. We used a Young's modulus of 0.75 MPa and a Poisson's ratio of 0.4 for the elastic objects.

10 CONCLUSION

We have seen that our operator splitting approach to simulating elastic solids with SPH can improve robustness and increase simulation speed for many scenarios. In some experiments, we observed more than 20 times faster simulations compared to state-of-the-art. Even for scenes where our operator splitting scheme is not applicable — for example if non-linear material models are required — our implicit zero-energy mode suppression could benefit researchers developing new SPH discretization methods. Similarly, our sampling algorithm produces high-quality particle samplings for almost any purpose, and is therefore not limited to SPH simulators.

Our method relies on a precomputed sparse matrix factorization. This works very well for a wide range of applications. However, as the number of particles per solid object grows very large, the memory and runtime requirements associated with the direct solver eventually become prohibitive. The point at which this occurs depends strongly on the problem and hardware, but our experiments suggest that $64k$ particles is still a feasible number, and at this point also significantly outperforms the iterative variant. With abundant memory, it is likely that the method remains effective for larger particle counts, but at some point the time and space complexity of the method will effectively place a bound on the number of solid particles per body that can be simulated efficiently. With the intent to scale to larger systems, we wish to investigate a variant of domain decomposition in the future, so that the factorization of the system matrix for each subdomain could be precomputed. In addition, it would be senseful to explore an adaptive SPH discretization for the elastic bodies, so that larger particles could be used in the interior and smaller ones at the surface, thereby reducing the overall particle count needed to attain similar visual fidelity.

Finally, like previous methods that use a pressure solver for solid-solid and fluid-solid coupling [Becker et al. 2009; Peer et al. 2018; Solenthaler et al. 2007], the final elastic response of the solid is inconsistent with respect to Poisson's ratio. Whereas the elastic solve correctly respects

Poisson's ratio, our pressure solver will treat the solid objects as incompressible under compression. Due to pressure clamping, the pressure solver will not counteract expansion, however. One possible remedy might be to adapt an implicit pressure solver that allows for compression for this purpose (e.g., [Gissler et al. 2020; Weiler et al. 2016]).

11 ACKNOWLEDGMENTS

The armadillo, bunny, dragon and lucy models are courtesy of the Stanford Computer Graphics Laboratory. The walrus model is courtesy of Yaroslav/CGTrader. This paper was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number BE 5132/3-2.

REFERENCES

Muzaffer Akbay, Nicholas Nobles, Victor Zordan, and Tamar Shinar. 2018. An Extended Partitioned Method for Conservative Solid-fluid Coupling. *ACM Transactions on Graphics* 37, 4, Article 86 (July 2018), 12 pages.

Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. 2012. Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics* 31, 4 (July 2012), 1–8. https://doi.org/10.1145/2185520.2335413

Markus Becker, Markus Ihmsen, and Matthias Teschner. 2009. Corotated SPH for deformable solids. In *Proceedings of Eurographics Conference on Natural Phenomena*. 27–34.

Jan Bender et al. 2021. SPlisHSPlasH Library. https://github.com/InteractiveComputerGraphics/SPlisHSPlasH.

Jan Bender and Dan Koschier. 2017. Divergence-Free SPH for Incompressible and Viscous Fluids. *IEEE Transactions on Visualization and Computer Graphics* 23, 3 (2017), 1193–1206.

Jan Bender, Dan Koschier, Tassilo Kugelstadt, and Marcel Weiler. 2019. Turbulent Micropolar SPH Fluids with Foam. *IEEE Transactions on Visualization and Computer Graphics* 25, 6 (2019), 2284–2295. https://doi.org/10.1109/TVCG.2018.2832080

Jan Bender, Tassilo Kugelstadt, Marcel Weiler, and Dan Koschier. 2020. Implicit Frictional Boundary Handling for SPH. *IEEE Transactions on Visualization and Computer Graphics* 26, 10 (2020), 2982–2993. https://doi.org/10.1109/TVCG.2020.3004245

Jan Bender, Matthias Müller, and Miles Macklin. 2017. A Survey on Position Based Dynamics, 2017. In *EUROGRAPHICS 2017 Tutorials*. Eurographics Association.

J. Bonet and T.-S. L. Lok. 1999. Variational and momentum preservation aspects of Smooth Particle Hydrodynamic formulations. *Computer Methods in Applied Mechanics and Engineering* 180, 1 (1999), 97 – 115.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics* 21, 3 (2002).

Wei Chen, Fei Zhu, Jing Zhao, Sheng Li, and Guoping Wang. 2018. Peridynamics-Based Fracture Animation for Elastoplastic Solids. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 112–124.

Xiao-Song Chen, Chen-Feng Li, Geng-Chen Cao, Yun-Tao Jiang, and Shi-Min Hu. 2020. A moving least square reproducing kernel particle method for unified multiphase continuum simulation. *ACM Transactions on Graphics* 39, 6 (2020), 1–15.

Jens Cornelis, Jan Bender, Christoph Gissler, Markus Ihmsen, and Matthias Teschner. 2019. An optimized source term formulation for incompressible SPH. *The Visual Computer* 35, 4 (2019), 579–590.

Francois Dagenais, Jonathan Gagnon, and Eric Paquette. 2012. A Prediction-Correction Approach for Stable SPH Fluid Simulation from Liquid to Rigid. In *Computer Graphics International*. 1–10.

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed Particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation*. 61–76.

Yu Fang, Ziyin Qu, Minchen Li, Xinxin Zhang, Yixin Zhu, Mridul Aanjaneya, and Chenfanfu Jiang. 2020. IQ-MPM: an interface quadrature material point method for non-sticky strongly two-way coupled nonlinear solids and fluids. *ACM Transactions on Graphics* 39, 4 (2020), 51–1.

Yun Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2019. A multi-scale model for coupling strands with shear-dependent liquid. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–20.

Yun (Raymond) Fei, Christopher Batty, Eitan Grinspun, and Changxi Zheng. 2018. A Multi-scale Model for Simulating Liquid-fabric Interactions. *ACM Transactions on Graphics* 37, 4, Article 51 (July 2018), 16 pages.

Yun (Raymond) Fei, Henrique Teles Maia, Christopher Batty, Changxi Zheng, and Eitan Grinspun. 2017. A Multi-Scale Model for Simulating Liquid-Hair Interactions. *ACM Transactions on Graphics* 36, 4 (2017). https://doi.org/10.1145/3072959.3073630

Georg C. Ganzenmüller. 2015. An hourglass control algorithm for Lagrangian Smooth Particle Hydrodynamics. *Computer Methods in Applied Mechanics and Engineering* 286 (apr 2015), 87–106.

Dan Gerszewski, Haimasree Bhattacharya, and Adam W Bargteil. 2009. A Point-based Method for Animating Elastoplastic Solids. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Vol. 1. 133–138.

Christoph Gissler, Andreas Henne, Stefan Band, Andreas Peer, and Matthias Teschner. 2020. An Implicit Compressible SPH Solver for Snow Simulation. *ACM Transactions on Graphics* 39, 4 (august 2020), 1–16.

Christoph Gissler, Andreas Peer, Stefan Band, Jan Bender, and Matthias Teschner. 2019. Interlinked SPH Pressure Solvers for Strong Fluid-Rigid Coupling. *ACM Transactions on Graphics* 38, 1 (2019).

Gaël Guennebaud, Benoît Jacob, et al. 2021. Eigen v3. http://eigen.tuxfamily.org.

Xuchen Han, Theodore F. Gast, Qi Guo, Stephanie Wang, Chenfanfu Jiang, and Joseph Teran. 2019. A Hybrid Material Point Method for Frictional Contact with Diverse Materials. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 17 (July 2019), 24 pages.

X. He, H. Wang, and E. Wu. 2018. Projective Peridynamics for Modeling Versatile Elastoplastic Materials. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1.

M. Huber, B. Eberhardt, and D. Weiskopf. 2015. Boundary Handling at Cloth-Fluid Contact. *Computer Graphics Forum* 34, 1 (2015), 14–25.

Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. 2014. Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics* 20, 3 (2014), 426–435.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015a. The Affine Particle-In-Cell Method. *ACM Transactions on Graphics* 34, 4 (July 2015), 51:1–51:10.

Min Jiang, Yahan Zhou, Rui Wang, Richard Southern, and Jian Jun Zhang. 2015b. Blue noise sampling using an SPH-based method. *ACM Transactions on Graphics* 34, 6 (2015), 1–11.

Ben Jones, Stephen Ward, Ashok Jallepalli, Joseph Perenia, and Adam W Bargteil. 2014. Deformation embedding for point-based elastoplastic simulation. *ACM Transactions on Graphics* 33, 2 (2014), 1–9.

R. Keiser, Bart Adams, D. Gasser, P. Bazzi, P. Dutre, and M. Gross. 2005. A unified Lagrangian approach to solid-fluid animation. In *Proceedings of Eurographics/IEEE VGTC Symposium Point-Based Graphics*. IEEE, 125–148.

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. 2019. Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids. In *EUROGRAPHICS 2019 Tutorials*. Eurographics Association.

Dan Koschier, Crispin Deul, Magnus Brand, and Jan Bender. 2017. An hp-Adaptive Discretization Algorithm for Signed Distance Field Generation. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (2017), 2208–2221.

Tassilo Kugelstadt, Dan Koschier, and Jan Bender. 2018. Fast Corotated FEM using Operator Splitting. *Computer Graphics Forum* 37, 8 (2018).

J. A. Levine, A. W. Bargteil, C. Corsi, J. Tessendorf, and R. Geist. 2014. A Peridynamic Perspective on Spring-mass Fracture. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Copenhagen, Denmark) *(SCA '14)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 47–55.

Miles Macklin and Matthias Müller. 2013. Position Based Fluids. *ACM Transactions on Graphics* 32, 4 (2013), 1–5.

Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2014. Unified Particle Physics for Real-Time Applications. *ACM Transactions on Graphics* 33, 4 (2014), 1–12.

Sebastian Martin, Peter Kaufmann, Mario Botsch, Eitan Grinspun, and Markus Gross. 2010. Unified Simulation of Elastic Rods, Shells, and Solids. *ACM Transactions on Graphics* 29, 4 (July 2010), 1.

Joseph J Monaghan. 2012. Smoothed particle hydrodynamics and its diverse applications. *Annual Review of Fluid Mechanics* 44 (2012), 323–346.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2006. Position Based Dynamics. In *Virtual Reality Interactions and Physical Simulations (VRIPHYS '06)*. Eurographics Association, 71–80. https://doi.org/10.2312/PE/vriphys/vriphys06/071-080

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless Deformations Based on Shape Matching. *ACM Transactions on Graphics* 24, 3 (2005), 471–478.

M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. 2004. Point based animation of elastic, plastic and melting objects. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 141.

Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. 2006. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum* 25, 4 (2006), 809–836.

Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J Guibas. 2005. Meshless Animation of Fracturing Solids. *ACM Transactions on Graphics* 24, 3 (2005), 957–964.

Andreas Peer, Christoph Gissler, Stefan Band, and Matthias Teschner. 2018. An Implicit SPH Formulation for Incompressible Linearly Elastic Solids. *Computer Graphics Forum* 37, 6 (2018), 135–148.

A. Peer, M. Ihmsen, J. Cornelis, and M. Teschner. 2015. An Implicit Viscosity Formulation for SPH Fluids. *ACM Transactions on Graphics* 34, 4 (2015), 1–10.

Xavier Provot. 1995. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In *Graphics Interface*. AK Peters, 147–154.

Stefan Reinhardt, Markus Huber, Bernhard Eberhardt, and Daniel Weiskopf. 2017. Fully asynchronous SPH simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2.

Stefan Reinhardt, Tim Krake, Bernhard Eberhardt, and Daniel Weiskopf. 2019. Consistent Shepard Interpolation for SPH-Based Fluid Animation. *ACM Transactions on Graphics* 38, 6, Article 189 (2019), 11 pages.

Nadine Abu Rumman, Prapanch Nair, Patric Müller, Loïc Barthe, and David Vanderhaeghe. 2019. ISPH–PBD: coupled simulation of incompressible fluids and deformable bodies. *The Visual Computer* (2019), 1–18.

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids. In *ACM SIGGRAPH Courses*. 1–50.

Barbara Solenthaler, Jürg Schläfli, and Renato Pajarola. 2007. A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds* 18, 1 (2007), 69–82.

Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, Joseph Teran, and Andrew Selle. 2014. Augmented MPM for phase-change and varied materials. *ACM Transactions on Graphics* 33, 4 (July 2014), 1–11.

Matthias Teschner, Bruno Heidelberger, Matthias Müller, and Markus Gross. 2004. A Versatile and Robust Model for Geometrically Complex Deformable Solids. In *Computer Graphics International (CGI '04)*. IEEE Computer Society, 312–319.

Marcel Weiler, Dan Koschier, and Jan Bender. 2016. Projective Fluids. In *ACM Motion in Games*. 1–6.

Marcel Weiler, Dan Koschier, Magnus Brand, and Jan Bender. 2018. A Physically Consistent Implicit Viscosity Solver for SPH Fluids. *Computer Graphics Forum* 37, 2 (2018).

Yahan Zhou, Zhaoliang Lun, Evangelos Kalogerakis, and Rui Wang. 2013. Implicit Integration for Particle-based Simulation of Elasto-Plastic Solids. *Computer Graphics Forum* 32, 7 (Nov. 2013), 215–223.