An Implicit Compressible SPH Solver for Snow Simulation

CHRISTOPH GISSLER, University of Freiburg, Germany and FIFTY2 Technology GmbH, Germany
ANDREAS HENNE, FIFTY2 Technology GmbH, Germany
STEFAN BAND, University of Freiburg, Germany
ANDREAS PEER, FIFTY2 Technology GmbH, Germany
MATTHIAS TESCHNER, University of Freiburg, Germany

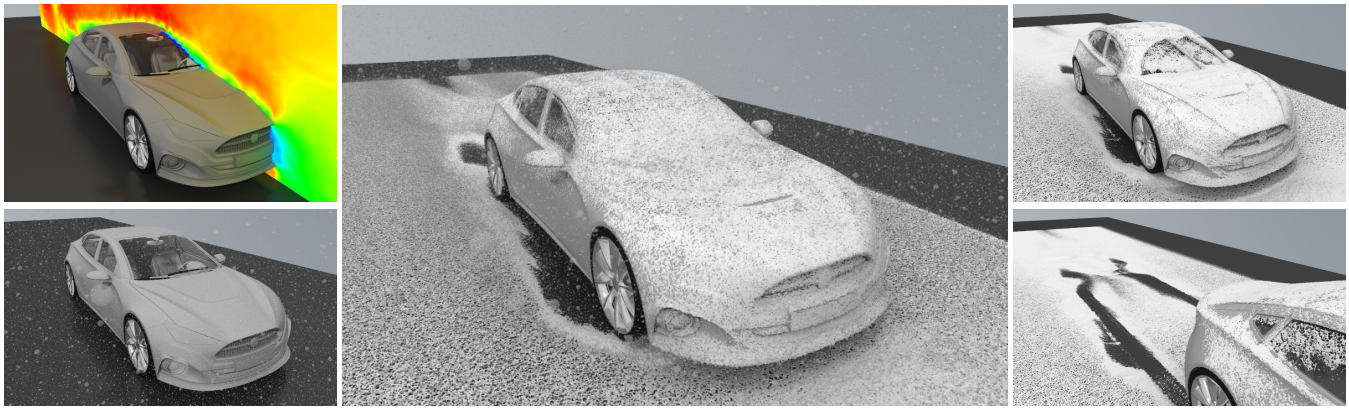


Fig. 1. Snow fall onto a car is simulated with single snow flakes modeled as single particles. The particles are coupled to a precomputed airflow field (visualized in the top left). The snow accumulates and is wiped away by moving windshield wipers. The car drives away in the end, compressing the snow below its tires.

Snow is a complex material. It resists elastic normal and shear deformations, while some deformations are plastic. Snow can deform and break. It can be significantly compressed and gets harder under compression. Existing snow solvers produce impressive results. E.g., hybrid Lagrangian/Eulerian techniques have been used to capture all material properties of snow. The auxiliary grid, however, makes it challenging to handle small volumes. In particular, snow fall and accumulation on surfaces have not been demonstrated with these solvers yet. Existing particle-based snow solvers, on the other hand, can naturally handle small snow volumes. However, existing solutions consider simplified material properties. In particular, shear deformation and the hardening effect are typically omitted.

We present a novel Lagrangian snow approach based on Smoothed Particle Hydrodynamics (SPH). Snow is modeled as an elastoplastic continuous material that captures all above-mentioned effects. The compression of snow is handled by a novel compressible pressure solver, where the typically employed state equation is replaced by an implicit formulation. Acceleration due to shear stress is computed using a second implicit formulation. The linear solvers of the two implicit formulations for accelerations due to shear and normal stress are realized with matrix-free implementations. Using implicit formulations and solving them with matrix-free solvers allows to couple the snow to other phases and is beneficial to the stability and the time step size, i.e., performance of the approach. Solid boundaries are represented with particles and a novel implicit formulation is used to handle friction at solid boundaries. We show that our approach can simulate accumulation, deformation, breaking, compression and hardening of snow. Furthermore, we demonstrate two-way coupling with rigid bodies, interaction with incompressible and highly viscous fluids and phase change from fluid to snow.

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

Additional Key Words and Phrases: Implicit solver, physically-based animation, smoothed particle hydrodynamics, snow, two-way coupling

ACM Reference Format:
Christoph Gissler, Andreas Henne, Stefan Band, Andreas Peer, and Matthias Teschner. 2020. An Implicit Compressible SPH Solver for Snow Simulation. *ACM Trans. Graph.* 39, 4, Article 36 (July 2020), 16 pages. https://doi.org/10.1145/3386569.3392431

1 INTRODUCTION

Smoothed Particle Hydrodynamics (SPH) is mostly known in the computer graphics community for the simulation of fluids, e.g., as described by Müller et al. [2003]; Ihmsen et al. [2014b]; Bender and Koschier [2017]. However, it has also been used to simulate a varied collection of different materials such as viscous fluids (e.g., as shown by Takahashi et al. [2015]; Peer et al. [2015]; Weiler et al. [2018]), elastic solids (e.g., as shown by Desbrun and Gascuel [1996]; Solenthaler et al. [2007]; Becker et al. [2009]; Peer et al. [2018]), dynamic rigid body objects (e.g., as proposed by Gissler et al. [2019]) or ferrofluids (e.g., as presented by Huang et al. [2019]). While SPH covers this wide range of materials, fully-featured snow simulations with SPH are uncommon.

Authors' addresses: C. Gissler, S. Band, and M. Teschner, Georges-Köhler-Allee 52, 79110 Freiburg im Breisgau, Germany; emails: {gisslerc, bands, teschner}@informatik.uni-freiburg.de; A. Henne, and A. Peer, Tullastraße 80, 79108 Freiburg im Breisgau, Germany; emails: {andreas.henne, andreas.peer}@fifty2.eu.

Instead, there are a lot of other approaches that each focus on capturing different types of snow effects. Height fields and signed distance fields are typically used to model snow accumulation. For example, Feldman and O'Brien [2002] use height fields and additionally take wind effects into account when computing the accumulated snow on geometry. Stomakhin et al. [2013] use the material point method (MPM) to simulate snow as a continuous material using an elastoplatic model. Their hybrid Lagrangian/Eulerian method is able to reproduce different types of snow dynamics but they do not show snow fall and accumulation. Lagrangian particle-based approaches simulate snow for example as fluids (e.g., as shown by Takahashi and Fujishiro [2012]) but since they employ a viscosity model and thus their snow simulation only reacts to shear rate instead of shear, they are unable to realistically capture complex physical snow behaviors for which a full elastoplastic snow model is needed. We present a novel snow solver based on SPH and show that we can reproduce various effects which were previously only shown by multiple different approaches. In Fig. 1, a simulation can be seen where we simulate snow fall on a car. The snow is coupled to a precomputed wind simulation and accumulates on the car. The wipers of the car move, which illustrates the interaction with complex and moving geometry. Since we use an elastoplastic model similar to the one by Stomakhin et al. [2013], we are also able to simulate different types of snow and the snow is able to compress, deform and break. We also show the coupling of snow with (viscous) fluids and dynamic rigid body objects.

Contributions.
We propose *a novel fully-fledged elastoplastic SPH approach for snow.* The proposed approach extends the range of materials that can be handled within the SPH formalism by realizing a variant of the constitutive model of Stomakhin et al. [2013]. As an essential component of our approach we propose *a novel implicit compressible SPH pressure solver.* This solver allows us to handle snow compression while providing the performance and stability advantages of an implicit formulation. The combination of this novel pressure solver with an implicit linear elastic solver inspired by Peer et al. [2018] enables the simulation of a wide range of snow dynamics and the coupling of snow with other phases. In addition to the novel handling of the elastic deformation, we propose to improve the typically employed differential update of the plastic deformation by incorporating additional information from the current particle configuration. Our boundary handling bases on particle representations which particularly simplifies the interaction of snow with complex geometries. In this context, we propose *a novel direct computation of a viscosity-based boundary friction which solves a limitation discussed in Peer et al. [2018].* Within the scope of the analyses of our SPH snow approach, we present *novel experiments showing the two-way coupling of snow with fluids and rigid bodies, snow fall and snow accumulation.*

2 RELATED WORK

Early snow-related computer graphics work mostly focused on snow accumulation on geometries. Nishita et al. [1997] employ metaballs to allow users to manually define snow accumulation on surfaces. Fearing [2000] computes the sky occlusion of areas by shooting particles upwards to automatically cover geometries with snow. They combine this with a stability criterion to get realistic snow accumulation behavior. Using height fields to represent snow is also a common method. Sumner et al. [1999] for example use a height field representation to model foot imprints in snow. Haglund et al. [2002] use height fields for the real-time simulation of snow accumulation. Reynolds et al. [2015] also use height fields for representing snow accumulation in real-time while respecting occlusion by other geometries. Feldman and O'Brien [2002] and Wang et al. [2006] couple snow fall simulations with wind simulations while also representing the accumulated snow with height fields. Festenberg and Gumhold [2009] use a statistical model for snow distribution which they later extended with a diffusion approach to model snow bridges and overhangs [2011]. Cordonnier et al. [2018] recently proposed to use multiple height fields to model different types of layered snow to simulate snow-covered landscapes and avalanches. Hinks and Museth [2009] use a level-set approach instead of height fields to more accurately simulate wind-driven snow buildup.

In recent years, more work is published that focuses on simulating the snow dynamics as a continuous material using a constitutive model. These simulation methods typically build on physical elastoplastic snow models, e.g., on the one presented by Meschke et al. [1996]. Most prominently, Stomakhin et al. [2013] introduced the material point method (MPM) in the computer graphics community to simulate snow. Stomakhin et al. [2014] further extended the MPM method to handle phase changes. There has been additional work extending the use and performance of MPM, e.g., by Tampubolon et al. [2017]; Wretborn et al. [2017]; Gao et al. [2018]; Fang et al. [2019]; Wang et al. [2019]. Other authors building upon the MPM method also show snow simulations, e.g., Gast et al. [2015]; Fang et al. [2018]; Hu et al. [2019b]. Recently, Han et al. [2019] showed frictional contact for MPM simulations, amongst other things between snow and hair. An elastoplastic snow model is able to reproduce realistic snow behavior as shown by Gaume et al. [2018, 2019] who predict snow avalanches using an adapted MPM snow solver based on the work of Stomakhin et al. [2013].

Apart from MPM, there are also other discretization methods that are used to simulate the complete snow dynamics. Wong and Fu [2015] use the Discrete Element Method (DEM) in combination with springs between the snow particles for an interactive simulation. Mukai et al. [2017] use an extended DEM method to simulate splitting and sliding snow on a roof. Position-based dynamics (PBD) is used by Dagenais et al. [2016] to simulate snow behavior by modeling snow particles as a granular material on top of a base level-set representation. Takahashi and Fujishiro [2012] approximate the behavior of snow by modeling it as a fluid using SPH. An extensive overview of SPH fluids is given by Monaghan [2012], Ihmsen et al. [2014b] and Koschier et al. [2019]. In a later work, they replaced SPH with the Fluid-Implicit-Particle (FLIP) method and added a durability measure to model compressibility. Abdelrazek et al. [2014] model snow as a fluid by employing the Bingham viscosity model in order to simulate snow avalanches. Goswami et al. [2019] use particles to model snow behavior in real-time on the GPU, but compute simple inter-particle forces to avoid evaluating an SPH interpolation kernel.

We simulate snow as an elastoplastic material and there exists extensive work with regard to elastic and elastoplastic materials in

the computer graphics community which is too large to be covered here in full extend. While Desbrun and Gascuel [1996] presented an early SPH-based method to simulate deformable bodies, there has also been a lot of work focusing on discretization methods (e.g., as proposed by Gerszewski et al. [2009] or Wojtan et al. [2009]), on fracturing (e.g., as shown by Hahn and Wojtan [2015]; Jones et al. [2016a]; Wolper et al. [2019]), or on other aspects, e.g., real-time simulation as for example recently shown by Brandt et al. [2018] or on plastic deformation without simulating elasticity as presented by Jones et al. [2016b]. While outdated, the report by Nealen et al. [2006] gives a good overview of different methods. Similarly, coupling between different phases is a popular research topic in the simulation community since it allows to reproduce interesting phenomena as shown by Losasso et al. [2006]. While it is out of scope for us to cover all the work, recent advances were made by Akbay et al. [2018], Brandt et al. [2019] and Gissler et al. [2019].

For our approach, we discretize the snow with SPH particles and compute accelerations based on an elastoplastic constitutive model. Using SPH in contrast to MPM simplifies the boundary handling and the simulation of single falling particles that accumulate over time. Furthermore, compared to previous approaches, we employ a combination of two implicit solvers to replicate the elastic behavior of the snow. First, we use a novel implicit compressible equation-of-state pressure solver. Compressible SPH-based fluid solvers usually explicitly model the fluid using a state equation, e.g., as proposed by Becker and Teschner [2007]. A notable exception is the fluid solver proposed by Weiler et al. [2016], which builds upon the Projective Dynamics method proposed by Bouaziz et al. [2014]. Weiler et al. use a linear state equation evaluated with SPH to formulate the constraints required by the implicit Projective Dynamics method. However, since their approach is based on the Projective Dynamics framework, it is more challenging to combine their solver with existing incompressible fluids that are typically simulated using iterative pressure solvers, e.g., as proposed by Solenthaler and Pajarola [2009]; Ihmsen et al. [2014a]; Bender and Koschier [2017]. For our solver, we combine the performance advantage of having an iterative implicit pressure solver, the possibility to couple it with existing incompressible solvers, and with the ability to model compression. As a second solver, we use a linear elasticity solver. The linearization of our elastic solver is guided by the work of Peer et al. [2018], who apply their solver for the simulation deformable objects with SPH. Compared to their approach, we simulate an elastoplastic material. Furthermore, we do not rely on the initial particle configuration and we adapted the discretization to satisfy our requirements for snow. In contrast to previous methods that focus on specific aspects of snow, our approach and the combined use of the two solvers allows us to simulate a wider range of effects, including snow fall and accumulation, phase change, phase interactions and compression, deformation and breaking.

3 METHOD

We base the physical behavior of our snow solver on the constitutive model proposed by Stomakhin et al. [2013]. Accordingly, we model the snow as a continuous elastoplastic material. The snow counteracts elastic deformations while plastic deformations are permanent and influence its stiffness. An acceleration resulting from the elastic deformation tries to return the snow into a rest configuration which is dependent on the previous plastic deformation of the snow. Accordingly, to simulate the snow behavior, we compute this elastic response and additionally update the plastic deformation over time.

In the following, we first give an overview of the complete algorithm for a single simulation step in Subsection 3.1. We then detail in Subsection 3.2 how we compute the acceleration due to elastic deformation. For this, we employ two implicit solvers. First, we propose a novel compressible pressure solver that iteratively computes accelerations counteracting the volume change of the snow. Secondly, we use an implicit linear solver that counteracts shear. Afterward, in Subsection 3.3, we explain how we update the plastic deformation of the snow and how the compression of the snow influences its stiffness and shear modulus. In Subsection 3.4, we describe our boundary handling which includes an implicit boundary friction formulation. Finally, we summarize and discuss our proposed method in Subsection 3.5.

3.1 Overview

A single simulation step consists of multiple successive computations. We give a brief overview of all the steps in this section before describing each step in detail in Subsections 3.2 to 3.4. Algorithm 1 shows the order of the steps. In a first loop over all particles (Lines 1 to 5), we compute and store variables for each particle that are required in later steps. This includes the current rest density of a particle (Line 2) which, e.g., indicates the compression of the snow and is used to model the plastic hardening, i.e., the increase of stiffness of the snow when compressed. In Line 3, the correction matrix $\mathbf{L}_i$ is computed which is used for an improved evaluation of the SPH kernel gradient (cf. Eq. (16)). Additionally, we evaluate accelerations like gravity which are included in $\mathbf{a}_i^{\text{other},t}$ (Line 4) and the friction-based acceleration $\mathbf{a}_i^{\text{friction},t}$ at the boundary (Line 5).

Algorithm 1 A single simulation step of our proposed SPH-based snow solver.

1: **foreach** particle i **do**
2: compute $\rho_{0,i}^t$ ▷ see Subsection 3.3.2
3: compute $\mathbf{L}_i$ ▷ see Eq. (15)
4: compute $\mathbf{a}_i^{\text{other},t}$ ▷ e.g., gravity and adhesion
5: compute $\mathbf{a}_i^{\text{friction},t}$ ▷ using Eq. (24)
6: SOLVE for $\mathbf{a}_i^{\lambda}$ ▷ see Subsection 3.2.1
7: SOLVE for $\mathbf{a}_i^{G}$ ▷ see Subsection 3.2.2
8: **foreach** particle i **do**
9: integrate $\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^t + \Delta t(\mathbf{a}_i^{\text{other},t} + \mathbf{a}_i^{\text{friction},t} + \mathbf{a}_i^{\lambda} + \mathbf{a}_i^{G})$
10: **foreach** particle i **do**
11: integrate $\mathbf{F}_{E,i}$ ▷ see Subsection 3.3.1
12: **foreach** particle i **do**
13: integrate $\mathbf{x}_i^{t+\Delta t} = \mathbf{x}_i^t + \Delta t \mathbf{v}_i^{t+\Delta t}$

Then, in Lines 6 and 7, we compute the acceleration of the snow due to elastic deformation. For this, we use two separate solvers

which simplifies the integration of our approach with existing SPH solvers for fluids and rigid bodies. We detail the motivation for using these two solvers in Subsection 3.2. In principle, we model the elastic response of the snow using the Lamé parameters λ and G. Each of the solver computes an acceleration corresponding to one of the Lamé parameters. First, a pressure solver computes $\mathbf{a}_i^\lambda$, reacting to compression of the snow. Then, we use a linear elasticity solver to compute $\mathbf{a}_i^G$ to counteract shear.

Finally, we integrate all particles. For this, we first integrate the velocities of each particles with the previously computed accelerations as shown in Lines 8 and 9. Then, we update and store the deformation gradient (Lines 10 and 11) of each particle. The separation of the overall deformation into elastic and plastic parts also happens in this step. Finally, we update the particle positions (Lines 12 and 13).

3.2 Elastic deformation

In this section, we describe how we compute an acceleration to react on the elastic deformation of each snow particle using our proposed snow solver. There are multiple requirements that our snow solver needs to satisfy. First, we want to integrate it into an existing SPH framework and want to simulate interactions between different phases. Secondly, snow gets less compressible the more it already is compressed until it is nearly incompressible. Furthermore, snow not only counteracts compression but also shear strain. For an elastic material, the Lamé parameters λ and G can be used to define the stress-strain relationship, i.e., the Cauchy stress $\boldsymbol{\sigma}$ is computed as $\boldsymbol{\sigma} = 2G\boldsymbol{\epsilon} + \lambda tr(\boldsymbol{\epsilon})\mathbb{1}$, where $\boldsymbol{\epsilon}$ is the strain and $\mathbb{1}$ is the identity matrix. However, instead of computing the stress and resulting acceleration in a single step, we compute the accelerations of the snow using two implicit solvers to satisfy the above-mentioned requirements.

The first solver iteratively computes a pressure which we use to compute the acceleration $\mathbf{a}^\lambda$ that counteracts compression of the snow depending on the Lamé parameter λ. This allows us to integrate our snow solver into an existing SPH framework where the other phases are also based on computing pressure. SPH-based compressible solvers that compute the pressure values explicitly using a state equation require that the time step needs to be severely restricted to keep the simulation stable, thus harming the performance. Alternatively, iterative pressure solvers that model incompressible materials are common and allow large time steps. In the following, we present a novel implicit compressible pressure solver that combines the compressibility of a state equation solver with the performance and stability of an iterative pressure solver and thus allows us to model the compressible snow. Our derivation of the system of equations of the solver is using a state equation and is motivated by the derivations used for the Implicit Incompressible SPH (IISPH) approaches by Ihmsen et al. [2014a]; Band et al. [2018a]. This leads us to an iterative matrix-free compressible pressure solver, which enables the integration with existing iterative pressure solvers.

The second solver counteracts the deformations that are not yet captured, e.g., shear-based deformations, by computing the acceleration $\mathbf{a}^G$ based on the Lamé parameter G. For this, we employ an implicit solver based on a linear hyperelastic material model. For the derivation of this solver, we adopt the linearization concept proposed by Peer et al. [2018]. We describe the derivation of the linear system, the discretization using SPH and its matrix-free implementation in Subsection 3.2.2.

3.2.1 Implicit equation-of-state solver. We derive an implicit formulation for an equation of state which enables us to solve the pressure of a compressible material. We use this pressure to compute an acceleration to counteract compression of the snow. In the following, we first show the theoretical derivation of the linear system of equations. We then give implementation details, e.g., how we discretize the equations with SPH and how we solve the system such that it integrates into existing pressure solvers.

Theory.
In general, we follow the basic steps outlined in the work by Band et al. [2018a] to derive our system of equations. Note that we omit the particle index i in the following for better readability. We start with the mass-conservation law:

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} + \rho\nabla\cdot\mathbf{v} = 0\,, \tag{1}$$

where $\mathbf{v}$ is the velocity and ρ the density. We compute the velocity at the next time $t+\Delta t$ as

$$\mathbf{v}^{t+\Delta t} = \mathbf{v}^* - \Delta t\frac{1}{\rho^{t+\Delta t}}\nabla p^{t+\Delta t} \tag{2}$$

where p is pressure and $\mathbf{v}^*$ is the predicted velocity based on known accelerations: $\mathbf{v}^* = \mathbf{v}^t + \Delta t(\mathbf{a}^{\mathrm{other},t} + \mathbf{a}^{\mathrm{friction},t})$. $\mathbf{a}^{\mathrm{friction},t}$ is a friction-based acceleration near the boundary which is described in more detail in Subsection 3.4. $\mathbf{a}^{\mathrm{other},t}$ includes accelerations like gravity. Note that for a better clarity in the derivation we use $\mathbf{v}^{t+\Delta t}$ in Eq. (2) although the final velocity used in the integration step also includes the acceleration from the second implicit solver which we detail in Subsection 3.2.2. By combining a backward-difference discretization at time $t+\Delta t$ of Eq. (1) with Eq. (2) and approximating $\rho^{t+\Delta t}\nabla\cdot(\frac{1}{\rho^{t+\Delta t}}\nabla p^{t+\Delta t}) = \nabla^2 p^{t+\Delta t}$, we get a system of equations with unknowns $\rho^{t+\Delta t}$ and $p^{t+\Delta t}$: $\rho^{t+\Delta t} - \Delta t^2\nabla^2 p^{t+\Delta t} = \rho^t - \Delta t\rho^{t+\Delta t}\nabla\cdot\mathbf{v}^*$. To only have the two unknowns on the left-hand side, we approximate the term on the right-hand side as $\rho^* = \rho^t - \Delta t\rho^t\nabla\cdot\mathbf{v}^*$ which finally leads to:

$$\rho^{t+\Delta t} - \Delta t^2\nabla^2 p^{t+\Delta t} = \rho^*\,. \tag{3}$$

For an incompressible material, we would now constrain the density at the next time $\rho^{t+\Delta t}$ to the rest density ρ_0. However, snow is compressible and we therefore need to find another way to replace the unknown density $\rho^{t+\Delta t}$. Accordingly, we use an equation of state to establish a linear relationship between $\rho^{t+\Delta t}$ and the unknown pressure $p^{t+\Delta t}$:

$$p^{t+\Delta t} = \lambda^{t+\Delta t}\left(\frac{\rho^{t+\Delta t}}{\rho_0^{t+\Delta t}} - 1\right). \tag{4}$$

$\lambda^{t+\Delta t}$ indicates the stiffness of the material which is based on the volume change, e.g., compression, of the snow. We simplify the equation by approximating $\rho_0^{t+\Delta t}$ with ρ_0^t and $\lambda^{t+\Delta t}$ with λ^t. We detail the computation of ρ_0^t and λ^t in Subsection 3.3.2. By using Eq. (4) to replace $\rho^{t+\Delta t}$ in Eq. (3) and further transformations, we

get the following linear equation with the only unknown being the pressure at the next time $p^{t+\Delta t}$:

$$-\frac{\rho_0^t}{\lambda^t} p^{t+\Delta t} + \Delta t^2 \nabla^2 p^{t+\Delta t} = \rho_0^t - \rho^* . \tag{5}$$

Note that Eq. (5) is not a pressure Poisson equation (PPE). However, in the limit of λ^t equaling infinity, Eq. (5) then is a PPE, e.g. as shown by Band et al. [2018a]. Stomakhin et al. [2014] and Kwatra et al. [2009] use a similar equation on a grid which they derive from the pressure evolution equation. By applying Eq. (5) to every snow particle i in our simulation, we get a linear system of equations which we solve to compute the pressure. The pressure gradient results in the acceleration $\mathbf{a}^\lambda = -\frac{1}{\rho}\nabla p$ that counteracts compression.

Implementation.
We solve Eq. (5) with a matrix-free relaxed Jacobi solver. By employing a relaxed Jacobi solver, we can interleave each iteration of our compressible pressure solver with the iterations of other iterative pressure solvers, e.g., with the solvers by Ihmsen et al. [2014a]; Band et al. [2018a]; Gissler et al. [2019]. This is in contrast to treating the solvers of the separate phases as black boxes, e.g., as done by Akbay et al. [2018]. Accordingly, on the right-hand side of Eq. (5), we need to compute a divergence of the predicted velocity encoded in ρ^*. On the left-hand side, we need to compute the Laplacian of the pressure. Following the idea presented by Ihmsen et al. [2014a], we discretize the Laplacian by two subsequent first-order derivatives: First, we compute the gradient of the pressure and then we compute the divergence of this pressure gradient. Accordingly, we need to discretize the divergence of a vector field on both sides of Eq. (5). We show the SPH discretization in the following using a generic vector field $\mathbf{v}$:

$$\nabla \cdot \mathbf{v}_i = \sum_k V_k \left(\mathbf{v}_k - \mathbf{v}_i\right) \cdot \nabla W_{ik} , \tag{6}$$

where k includes neighbors of all phases and boundary neighbors and ∇W is the SPH kernel gradient. $V_k = \frac{m_k}{\rho_k}$ is the volume of the respective particles, where m_k denotes the constant mass of a particle. See Subsection 3.4 for more information on the volume of a boundary particle. We discretize the pressure gradient as

$$\nabla p_i = \sum_j \left(p_j + p_i\right) V_j \nabla W_{ij} + \psi p_i \sum_b V_b \nabla W_{ib} , \tag{7}$$

where j are fluid and snow neighbors and b are boundary neighbors. ψ is a parameter which we set to 1.5 inspired by the work of Akinci et al. [2012]. Since we use a relaxed Jacobi solver, we need to compute the diagonal element a_{ii} of the system matrix. This is done as:

$$\begin{aligned} a_{ii} = &-\frac{\rho_0^t}{\lambda^t} - \Delta t^2 \sum_j V_i V_j \left\| \nabla W_{ij} \right\|^2 \\ &-\Delta t^2 \left(\sum_j V_j \nabla W_{ij} + \psi \sum_b V_b \nabla W_{ib} \right) \cdot \sum_k V_k \nabla W_{ik} , \end{aligned} \tag{8}$$

where j are fluid and snow neighbors, b are boundary neighbors and k are all three.

Equations (6) to (8) are used by the relaxed Jacobi solver of which we show an overview in Algorithm 2. The pressure is updated in each iteration l using the relaxation factor $\omega = 0.5$ and we iterate until the error is lower than a user-defined threshold, typically 0.1 % density deviation. Note that we do not clamp the pressure in contrast to what is typically done for fluid solvers, e.g., as discussed by Ihmsen et al. [2014a]. Accordingly, we allow positive and negative pressure values for our snow particles.

Algorithm 2 Solver steps of the proposed implicit compressible pressure solver.

procedure PREPARE
  foreach particle i **do**
    compute $\rho_i^* = \rho_i^t - \Delta t \rho_i^t \nabla \cdot \mathbf{v}_i^*$ ▷ using Eq. (6)
    compute a_{ii} ▷ see Eq. (8)
procedure SOLVE
  while not converged **do**
    foreach particle i **do**
      compute ∇p_i^l ▷ see Eq. (7)
    foreach particle i **do**
      compute $(\mathbf{A}p^l)_i$ ▷ LHS of Eq. (5)
      $p_i^{l+1} = p_i^l + \frac{\omega}{a_{ii}} \left(\rho_0^t - \rho^* - (\mathbf{A}p^l)_i\right)$

3.2.2 Shear. Apart from the accelerations counteracting a volume change, there are also large shear-based accelerations acting inside of the snow. We describe the computation of these accelerations in the following and start by giving a theoretical derivation of the implicit system that we solve to compute these accelerations. For readability, we omit the particle index i in the theory section. This system is based on a hyperelastic material model and inspired by the work of Peer et al. [2018]. We then give implementation details regarding the discretization with SPH and solving of the system.

Theory.
Since we use a hyperelastic material model, the Cauchy stress is computed as $\boldsymbol{\sigma} = 2G\boldsymbol{\epsilon} + \lambda tr\left(\boldsymbol{\epsilon}\right)\mathbb{1}$. We already computed the second term on the right-hand side and the resulting acceleration is included in $\mathbf{a}^\lambda$ as described in the previous section. Accordingly, we compute $\boldsymbol{\sigma}^{t+\Delta t}$ as

$$\boldsymbol{\sigma}^{t+\Delta t} = 2G^t \boldsymbol{\epsilon}^{t+\Delta t} . \tag{9}$$

Note that $\boldsymbol{\sigma}^{t+\Delta t}$ as computed in Eq. (9) contains a response to shear and volume deformations. Only considering the shear, this would need to read $2G^t(\boldsymbol{\epsilon}^{t+\Delta t} - \frac{1}{3}tr(\boldsymbol{\epsilon}^{t+\Delta t}))$. We discuss the reason for using the formulation shown in Eq. (9) in Subsection 3.5. We explain the computation of the time- and particle-based shear modulus G^t which we use as an approximation of $G^{t+\Delta t}$ in Subsection 3.3.2. $\boldsymbol{\epsilon}^{t+\Delta t}$ is the strain at time $t + \Delta t$ which encodes the deformation of the material.

To build a linear implicit system, we start with the computation of the velocity at time $t + \Delta t$ based on the divergence of the Cauchy stress as

$$\mathbf{v}^{t+\Delta t} = \mathbf{v}^{**} + \Delta t \frac{1}{\rho^t} \nabla \cdot \boldsymbol{\sigma}^{t+\Delta t} , \tag{10}$$

where the predicted velocity $\mathbf{v}^{**}$ includes all known accelerations as $\mathbf{v}^{**} = \mathbf{v}^t + \Delta t(\mathbf{a}^{\text{other},t} + \mathbf{a}^{\text{friction},t} + \mathbf{a}^\lambda)$.

Following Eq. (9), $\boldsymbol{\sigma}^{t+\Delta t}$ in Eq. (10) depends on the elastic deformation of the snow at time $t+\Delta t$. In turn, the future elastic deformation depends on the unknown velocity $\mathbf{v}^{t+\Delta t}$ since the velocities determine how the particles move relative to each other. By following these successive dependencies, we can transform Eq. (10) into the desired system of equations. Accordingly, we continue by defining how we compute the strain $\boldsymbol{\epsilon}^{t+\Delta t}$. Since we use a corotational model (cf. Subsection 3.3.1), we can follow the work of Peer et al. [2018] and use the infinitesimal strain tensor to keep our model linear:

$$\boldsymbol{\epsilon}^{t+\Delta t} = \frac{1}{2}\left(\mathbf{F}_E^{t+\Delta t} + \left(\mathbf{F}_E^{t+\Delta t}\right)^\top\right) - \mathbb{1}\,, \tag{11}$$

where $\mathbf{F}_E^{t+\Delta t}$ is the elastic deformation gradient at time $t+\Delta t$. The deformation gradient encodes how the deformation changes spatially and is based on the displacement of the material over time. Accordingly, we can compute $\mathbf{F}_E^{t+\Delta t}$ based on the current elastic deformation gradient $\mathbf{F}_E^t$ and the change of displacement $\Delta t\nabla\mathbf{v}^{t+\Delta t}$ as

$$\mathbf{F}_E^{t+\Delta t} = \mathbf{F}_E^t + \Delta t\left(\nabla\mathbf{v}^{t+\Delta t}\right)\mathbf{F}_E^t\,. \tag{12}$$

The differential update is frequently used in MPM, e.g., by Stomakhin et al. [2013, 2014]; Hu et al. [2018] and we adopt this form. In the beginning, the deformation gradient is initialized with the identity matrix. Note that Eq. (12) is only an approximation of the steps we use to integrate $\mathbf{F}_E$ over time where we additionally use a rotation extraction and a splitting of plastic and elastic deformations. We use this approximation here to achieve a linear dependence of the deformation gradient on the velocity $\mathbf{v}^{t+\Delta t}$. The details for the full integration of $\mathbf{F}_E$ are shown in Subsection 3.3.1. By using Eqs. (9), (11) and (12) in Eq. (10) we get a linear system with unknown velocities $\mathbf{v}^{t+\Delta t}$:

$$\begin{aligned}\mathbf{v}^{t+\Delta t} - \frac{\Delta t^2}{\rho^t}\nabla\cdot\left(G^t\left(\left(\nabla\mathbf{v}^{t+\Delta t}\right)\mathbf{F}_E^t + \left(\left(\nabla\mathbf{v}^{t+\Delta t}\right)\mathbf{F}_E^t\right)^\top\right)\right)\\ = \mathbf{v}^{**} + \frac{\Delta t}{\rho^t}\nabla\cdot\left(G^t\left(\mathbf{F}_E^t + \left(\mathbf{F}_E^t\right)^\top - 2\mathbb{1}\right)\right)\,,\end{aligned} \tag{13}$$

where we moved all known terms to the right-hand side. In the following, we explain how we discretize Eq. (13) using SPH and solve the system of equations to get an acceleration $\mathbf{a}^G$.

Implementation.
Instead of solving for the velocity $\mathbf{v}_i^{t+\Delta t}$, we directly solve for an acceleration $\mathbf{a}_i^G$ in our implementation. By transforming Eq. (13) we get

$$\begin{aligned}\mathbf{a}_i^G - \frac{\Delta t}{\rho_i^t}\nabla\cdot\left(G_i^t\left(\left(\nabla\mathbf{a}_i^G\right)\mathbf{F}_{E,i}^t + \left(\left(\nabla\mathbf{a}_i^G\right)\mathbf{F}_{E,i}^t\right)^\top\right)\right)\\ = \frac{1}{\rho_i^t}\nabla\cdot\left(2G_i^t\left(\frac{1}{2}\left(\mathbf{F}_{E,i}^{**} + \left(\mathbf{F}_{E,i}^{**}\right)^\top\right) - \mathbb{1}\right)\right)\,,\end{aligned} \tag{14}$$

where $\mathbf{F}_{E,i}^{**} = \mathbf{F}_{E,i}^t + \Delta t\nabla\mathbf{v}_i^{**}\mathbf{F}_{E,i}^t$ is the predicted deformation gradient due to the predicted velocities $\mathbf{v}_i^{**}$. As outlined by Peer et al. [2018], the linear system shown in Eq. (14) can be solved in a matrix-free way. We need to discretize the gradient and divergence computations in Eq. (14) for which we employ SPH as detailed in the following.

Gradient discretization.
The gradient of a vector fields needs to be computed on both sides of Eq. (14). On the left-hand side, the gradient of the acceleration $\mathbf{a}^G$ is needed while on the right-hand side, the gradient of $\mathbf{v}^{**}$ is required. In the following, we show how we compute this gradient with SPH. We use the same discretization on both sides of Eq. (14) and therefore just show the discretization for a generic velocity $\mathbf{v}$. In order to capture rotational motion of the velocity field in the velocity gradient $\nabla\mathbf{v}_i$ more accurately, we employ a first-order consistent kernel gradient $\tilde{\nabla}W$ as proposed by Peer et al. [2018]. Accordingly, we follow Bonet and Lok [1999] and compute a correction matrix for the standard SPH gradient ∇W:

$$\mathbf{L}_i = \left(\sum_j V_j\nabla W_{ij}\otimes\mathbf{x}_{ji}\right)^{-1}\,, \tag{15}$$

where j are snow and boundary neighbors of particle i and $\mathbf{x}_{ji} = \mathbf{x}_j - \mathbf{x}_i$. If the particles are in a coplanar or colinear configuration, the matrix cannot be inverted and we instead compute the Moore-Penrose pseudoinverse as proposed by Peer et al. [2018]. $\mathbf{L}_i$ from Eq. (15) can be used to compute the corrected kernel gradient as

$$\tilde{\nabla}W_{ij} = \mathbf{L}_i\nabla W_{ij}\,. \tag{16}$$

Computing the velocity gradient using the corrected kernel gradient improves the capturing of rotational movement. However, we noticed that it often leads to an overestimation of the volume change. To alleviate this problem, we combine the volume change from a velocity gradient $\nabla\mathbf{v}_i^{'}$ which we compute using the normal kernel gradient with rotational and shear changes from a velocity gradient $\tilde{\nabla}\mathbf{v}_i$ computed using the corrected kernel gradient. We start by calculating uncorrected snow-based and boundary-based velocity gradients as:

$$\begin{aligned}\nabla\mathbf{v}_{i,s}^{'} &= \sum_j\left(\mathbf{v}_j - \mathbf{v}_i\right)\otimes V_j\nabla W_{ij}\,,\\ \nabla\mathbf{v}_{i,b}^{'} &= \sum_b\left(\mathbf{v}_b - \mathbf{v}_i\right)\otimes V_b\nabla W_{ib}\,,\end{aligned} \tag{17}$$

where j are neighboring snow particles and b are neighboring boundary particles. Due to the definition of the corrected kernel gradient (cf. Eq. (16)), the corrected velocity gradient can now simply be computed as $\tilde{\nabla}\mathbf{v}_i = \nabla\mathbf{v}_{i,s}^{'}\mathbf{L}_i^\top + \frac{1}{3}tr(\nabla\mathbf{v}_{i,b}^{'}\mathbf{L}_i^\top)\mathbb{1}$. Note that we only consider the volume change with regard to the boundary. This is motivated by the fact that we compute frictional forces separately as shown in Subsection 3.4.

To get the final velocity gradient $\nabla\mathbf{v}_i$, we take the volume change of the uncorrected velocity gradient $\mathbf{V}_i^{'} = \frac{1}{3}tr(\nabla\mathbf{v}_{i,s}^{'} + \nabla\mathbf{v}_{i,b}^{'})\mathbb{1}$ and combine it with the rotational part $\tilde{\mathbf{R}}_i = \frac{1}{2}(\tilde{\nabla}\mathbf{v}_i - (\tilde{\nabla}\mathbf{v}_i)^\top)$ and shear part $\tilde{\mathbf{S}}_i = \frac{1}{2}(\tilde{\nabla}\mathbf{v}_i + (\tilde{\nabla}\mathbf{v}_i)^\top) - \frac{1}{3}tr(\tilde{\nabla}\mathbf{v}_i)\mathbb{1}$ of the corrected velocity gradient:

$$\nabla\mathbf{v}_i = \tilde{\mathbf{R}}_i + \mathbf{V}_i^{'} + \tilde{\mathbf{S}}_i\,. \tag{18}$$

Divergence discretization.
After having computed the gradient as shown in the previous section, the remaining spatial derivative in Eq. (14) is the divergence of a vector field. Although there are some transformations in Eq. (14), this is essentially the divergence of a stress tensor. Accordingly, in

the following, we show how we compute $\nabla \cdot \boldsymbol{\sigma}_i$ with SPH for which we also use the corrected kernel gradient:

$$\nabla \cdot \boldsymbol{\sigma}_i = \left(\sum_j \boldsymbol{\sigma}_j \left(-V_j \tilde{\nabla} W_{ji} \right) + \boldsymbol{\sigma}_i V_j \tilde{\nabla} W_{ij} + \boldsymbol{\sigma}^{\mathrm{b}}_{b,i} \sum_b V_b \tilde{\nabla} W_{ib} \right), \tag{19}$$

where j are neighboring snow particles, b are neighboring boundary particles and $\boldsymbol{\sigma}^{\mathrm{b}}_{b,i}$ is a mirrored Cauchy stress tensor which we compute as $\boldsymbol{\sigma}^{\mathrm{b}}_{b,i} = \frac{1}{3} tr\left(\boldsymbol{\sigma}_i\right) \mathbb{1}$.

Solver.

We solve the linear system described in Eq. (14) using a Bi-CGSTAB solver as proposed by van der Vorst [1992], since the system is not symmetric and we did not investigate a potential symmetrization. The solver requires us to compute the right-hand side and the matrix-vector product for a generic basis vector on the left-hand side. Both can be implemented in a matrix-free way which is shown in Algorithm 3.

Algorithm 3 Computation steps for the Bi-CGSTAB solver.

procedure RIGHT-HAND-SIDE
  foreach particle i **do**
    compute $\nabla \mathbf{v}_i^{**}$ ▷ using Eqs. (15), (17) and (18)
  foreach particle i **do**
    $\frac{1}{\rho^t} \nabla \cdot \left(2G_i^t \left(\frac{1}{2} \left(\mathbf{F}^{**}_{E,i} + \left(\mathbf{F}^{**}_{E,i} \right)^\top \right) - \mathbb{1} \right) \right)$ ▷ RHS of Eq. (14)
procedure LEFT-HAND-SIDE(basis vector $\mathbf{b}$)
  foreach particle i **do**
    compute $\nabla \mathbf{b}_i$ ▷ using Eqs. (15), (17) and (18)
  foreach particle i **do**
    $\mathbf{b}_i - \frac{\Delta t}{\rho_i^t} \nabla \cdot \left(G_i^t \left(\left(\nabla \mathbf{b}_i \right) \mathbf{F}^t_{E,i} + \left(\left(\nabla \mathbf{b}_i \right) \mathbf{F}^t_{E,i} \right)^\top \right) \right)$ ▷ LHS of Eq. (14)

3.3 Plastic deformation

As described in Subsection 3.2, our snow model computes an acceleration for each particle to counteract the elastic deformation of the snow. Additionally, the snow deforms plastically. This has two effects: First, the rest configuration of the snow changes over time. This is modeled by imposing a maximum elastic deformation following the idea of Stomakhin et al. [2013]. Deformations larger than this maximum are regarded as a permanent plastic deformation. This is explained in more detail in Subsection 3.3.1. The second aspect is that the snow gets harder when it is compressed, i.e., the elastic Lamè parameters λ and G used in the previous section change depending on the compression of a snow particle. This hardening of the snow is described in Subsection 3.3.2.

3.3.1 Maximum elastic deformation. Instead of evaluating a yield function using the Cauchy stress and projecting the elastic deformation gradient on the surface of this yield function, we adopt the simpler approach used by Stomakhin et al. [2013] and change the rest configuration of the snow by only allowing a maximum elastic deformation. Deformations larger than this limit are considered permanent. Following the proposal of Stomakhin et al. [2013], we clamp the elastic deformation gradient $\mathbf{F}_{E,i}$ while integrating it. In general, we update $\mathbf{F}_{E,i}$ with a differential update in the integration step in each time step. We first update the velocities from $\mathbf{v}_i^t$ to $\mathbf{v}_i^{t+\Delta t}$ using the previously computed accelerations. Then, using the velocity gradient $\nabla \mathbf{v}_i^{t+\Delta t}$, we compute an intermediate elastic deformation $\mathbf{F}^{'}_{E,i} = \mathbf{F}^t_{E,i} + \Delta t \nabla \mathbf{v}_i^{t+\Delta t} \mathbf{F}^t_{E,i}$.

We determine if the deformation was too large by using a singular value decomposition (SVD) of $\mathbf{F}^{'}_{E,i} = \mathbf{U}_i \Sigma^{'}_i \mathbf{V}_i^\top$ and clamping the singular values with $\Sigma_n = \mathrm{clamp}(\Sigma^{'}_n, [1 - \theta_c, 1 + \theta_s])$ for $1 \leq n \leq 3$ where $\Sigma^{'}_n$ and Σ_n are the singular values of $\Sigma^{'}_i$ and the new matrix Σ_i respectively. θ_c and θ_s are user-defined parameters. Additionally, we want to only keep the shear part of $\mathbf{F}^{'}_{E,i}$ and remove the rotational part. Since we already have the SVD, we can just reconstruct the clamped elastic deformation gradient as $\mathbf{F}^{t+\Delta t}_{E,i} = \mathbf{V}_i \Sigma_i \mathbf{V_i}^\top$ to leave out the rotational part $\mathbf{U}_i \mathbf{V}_i^\top$.

3.3.2 Hardening. Snow gets harder when it is compressed. We model this behavior by first estimating the current compression of the snow based on the current particle configuration. For this, we compute the current density of each particle i using SPH:

$$\rho_i^t = \sum_k m_k W_{ik} \tag{20}$$

where k are particle neighbors and the kernel is evaluated based on the current positions of all particles at time t. ρ_i^t is the current density of particle i. This includes the compression which is due to a permanent plastic deformation of the snow and the compression due to the elastic deformation of the snow. For the computation of the hardening effect of the snow, we only want to consider the plastic deformation. This means we need to compute the rest density of particle i at the current time: $\rho^t_{0,i}$. Based on ρ_i^t, we can compute this rest density by removing the elastic compression from it. Since we know the elastic deformation gradient and since the determinant of the deformation gradient indicates the volume change, we compute $\rho^t_{0,i}$ as:

$$\rho^t_{0,i} = \rho_i^t \left| det\left(\mathbf{F}^t_{E,i} \right) \right|. \tag{21}$$

Note that we use the absolute value of the determinant of the elastic deformation gradient due to the fact that the elastic deformation gradient may be inverted.

Using the current rest density $\rho^t_{0,i}$, we then compute the Lamé parameters to model the hardening following the work of Stomakhin et al. [2013]:

$$G_i^t = \frac{E}{2\left(1+\nu\right)} e^{\xi \frac{\rho^t_{0,i} - \rho_0}{\rho^t_{0,i}}}, \tag{22}$$

$$\lambda_i^t = \frac{E\nu}{\left(1+\nu\right)\left(1-2\nu\right)} e^{\xi \frac{\rho^t_{0,i} - \rho_0}{\rho^t_{0,i}}}. \tag{23}$$

The Young modulus E, Poisson's ratio ν and hardening coefficient ξ can be set by the user to influence the snow behavior.

3.4 Boundary Handling

Our boundary handling is purely particle-based and builds on the approaches proposed by Solenthaler and Pajarola [2008] and Akinci et al. [2012] by sampling geometries with boundary particles b. To sample triangle meshes with particles, we use an algorithm similar to the one presented by Bell et al. [2005]. We then compute a volume for each boundary particle as $V_b = \phi \frac{1}{\sum_j W_{bj}}$ where j are boundary neighbors of b and h is the particle spacing. ϕ is a scaling factor which we set to 0.8 motivated by Band et al. [2018a]. In contrast to ψ used in Eq. (7) which only influences the pressure force at the boundary, ϕ influences all computations which consider boundary particles. For the implicit equation-of-state solver presented in Subsection 3.2.1, we integrate the boundary particles in Eqs. (6) to (8). For the elastic solver proposed in Subsection 3.2.2, the boundary particles are considered in Eqs. (15), (17) and (19).

Additionally, we consider the boundary in the computation of accelerations that contribute to the predicted velocity $\mathbf{v}^*$. This includes an acceleration due to adhesion and the frictional acceleration $\mathbf{a}^{\text{friction},t}$ which we both detail in the following.

Adhesion.

Since we do not clamp the pressure to be positive, we have adhesive behavior at the boundary if the pressure of a snow particle is negative. To be able to additionally influence the strength of the adhesive behavior, we implemented an adhesion force acting between snow and boundary particles based on the inter-particle force proposed by Akinci et al. [2013].

Friction.

Snow often sticks to boundaries which may require large friction-based accelerations for snow particles near the boundary. We compute this frictional behavior using an SPH viscosity formulation. To achieve a sticking behavior, a high friction value needs to be chosen which would limit the time step severely when employing an explicit viscosity model. Using an implicit model allows to use larger time steps while remaining stable and thus improves performance. We compute the friction acceleration based on the implicit viscosity formulation proposed by Weiler et al. [2018]. In their approach, Weiler et al. solve a linear system to compute the viscosity-based accelerations. In contrast to their work, we use a different discretization for the computation of the Laplacian. Weiler et al. compute the Laplacian as proposed by Monaghan [2005]. However, this discretization does not result in any force when two particles move tangentially with respect to each other as detailed by Band et al. [2018b]. Instead, we use the discretization of the Laplacian as presented by Morris et al. [1997]. Interestingly, since we only compute friction between snow and boundary and not in between snow particles, we can solve this implicit system separately for each particle and thus do not need to run a computationally expensive iterative solver. The intuition behind this approach is that in our linear system of equations, the system matrix has non-zero entries only on the diagonal. Accordingly, we compute the friction acceleration by dividing the right-hand side of the viscosity system by the respective diagonal element d_{ii}:

$$\mathbf{a}_i^{\text{friction},t} = \frac{1}{d_{ii}\Delta t}\left(\mathbf{v}_i^t + \Delta t \mathbf{a}_i^{\text{other},t} - \Delta t \nu_b \sum_b V_b \frac{\mathbf{x}_{ib}\cdot\nabla W_{ib}}{||\mathbf{x}_{ib}||^2 + 0.01h^2}\mathbf{v}_b\right), \tag{24}$$

where the friction coefficient ν_b is a user-defined parameter that allows to change the strength of the friction effect. The diagonal elements d_{ii} are computed as:

$$d_{ii} = 1 - \Delta t \nu_b \sum_b V_b \frac{\mathbf{x}_{ib}\cdot\nabla W_{ib}}{||\mathbf{x}_{ib}||^2 + 0.01h^2}. \tag{25}$$

3.5 Discussion

We presented a novel snow solver that combines two implicit solvers to compute the material behavior. By using SPH as a discretization method and coupling an implicit compressible pressure solver with a hyperelastic material solver and introducing an implicit boundary friction, we are able to simulate a wide range of scenarios as we will show in Section 4.

The two combined solvers each compute a separate part of the total elastic force of a snow particle. Each solver is essentially responsible for one term of the Cauchy stress $\boldsymbol{\sigma} = 2G\boldsymbol{\epsilon} + \lambda tr(\boldsymbol{\epsilon})$. This split allows us to compute the second term using a novel compressible pressure solver such that it can be interleaved with existing incompressible pressure solvers. While we use two solvers, this does not correspond to a complete split into a volume-based part and a shear-based part as stated in Subsection 3.2.2. This is due to the fact that Eq. (9) still encodes a volume change in the strain. Completely splitting the volume and shear reactions would mean we need to use $K = \lambda + \frac{2G}{3}$ as stiffness in the compressible pressure solver and use $\boldsymbol{\sigma} = 2G(\boldsymbol{\epsilon} - \frac{1}{3}tr(\boldsymbol{\epsilon}))$ instead of Eq. (9). However, volume and shear changes are actually dependent on each other and we noticed that either smaller time steps or multiple alternating runs of the two solvers were needed to achieve highly shear-resistant snow when using this formulation. Using the separation as proposed in the previous sections allows us to reproduce a wide range of snow behavior while keeping a good solver performance and enabling the coupling with other phases.

It is also possible to solve the snow dynamics by only using the solver described in Subsection 3.2.2. For this, Eq. (9) would need to be replaced with $\boldsymbol{\sigma}^{t+\Delta t} = 2G^t\boldsymbol{\epsilon}^{t+\Delta t} + \lambda^t tr(\boldsymbol{\epsilon}^{t+\Delta t})\mathbb{1}$. While this brings a performance benefit, it makes coupling with other phases more challenging. To compare our proposed approach to computing compression and shear with a single solver, we simulated the scene shown in Fig. 11 with both approaches. While the results vary slightly with the same parameters, it is possible to achieve a similar visual appearance by slightly adapting them. Accordingly, we chose to favor versatility (i.e., being able to couple our snow with other materials) over performance. We additionally discuss the results of the comparison simulations in Subsection 4.2.4.

4 RESULTS

In this section we demonstrate the capabilities of our snow solver by simulating various experiments and showcases. We first show the friction effects and the result of changing snow parameters on the

compression of the snow in Subsection 4.1. We also show a scene where we couple the Young modulus to on an additional temperature simulation. In Subsection 4.2, we investigate the properties of our snow solver in more detail. For example, we show the behavior for single particles, show interactions with animated geometry, simulate snow fall and accumulation, and compare our approach to an MPM-based snow simulation. In Subsection 4.3, we present the coupling of the snow solver with other phases. This includes the coupling of single particles to a precomputed wind simulation. We further demonstrate two-way coupling with rigid bodies. We also show the interaction of snow and incompressible fluids and phase transitions from fluid to snow and the interaction of snow with highly viscous fluid. An overview over all scenes is given in Table 1. We rendered all scenes using PreonLab by FIFTY2 Technology GmbH [2020].

Table 1. Overview of used particle spacing, maximum number of particles and average simulation statistics for the presented scenes.

Scene		particle number		average			
		spacing	of particles	time-step	iter. $\mathbf{a}^\lambda$	iter. $\mathbf{a}^G$	time per frame
Snowballs	Fig. 2	2 mm	122k	0.2 ms	2.7	35.1	28.7 s
Compression							
20 kPa	Fig. 3	3 cm	110k	1.5 ms	5.1	1.0	1.98 s
140 kPa	Fig. 3	3 cm	110k	1.5 ms	4.76	1.17	1.87 s
IISPH	Fig. 3	3 cm	110k	1.5 ms	8.93	1.0	1.91 s
Tire	Fig. 6	1.5 mm	5.5M	0.15 ms	3.2	4.2	2.6 min
Oven bunnies	Fig. 7	1 mm	878k	0.1 ms	1.9	16.9	3.7 min
Single particles	Fig. 8	5 mm	2028	0.5 ms	3.2	2.0	0.49 s
Snow angel	Fig. 9	1 cm	9.42M	1 ms	4.8	13.7	10 min
City street	Fig. 10	8 cm	5.07M	3.1 ms	3.1	27.2	3.24 min
Armadillo							
split 1	Fig. 11a	2.6 cm	5.37M	1.6 ms	5.1	17.4	4.5 min
split 2	Fig. 11b	2.6 cm	5.37M	1.6 ms	5.28	7.79	2.6 min
combined 1	Fig. 11c	2.6 cm	5.37M	1.6 ms		16.86	3.5 min
combined 2	Fig. 11d	2.6 cm	5.37M	1.6 ms		4.54	1.6 min
Rel. to MPM	Fig. 12b	5 mm	1350	0.5 ms	3.65	11.31	0.13 s
Density comp.	Fig. 13	5 mm	1171	0.1 ms	2.64	7.05	0.14 s
Car snowing	Fig. 1	5 mm	5.04M	0.1 ms	3.6	1.9	8.7 min
Sled race	Fig. 14	1.5 cm	6.63M	1.2 ms	4.7	20.1	8.0 min
Submarine	Fig. 15	0.25 m	6.52M	5 ms	14.4	16.1	28.1 s
Ice machine	Fig. 16	1 mm	953k	0.1 ms	3.0	9.4	3 min

We simulated the experiments on a 16-core 3.1 GHz Intel Xeon E5-2687W workstation and by default used a framerate of 50 frames s^{-1}. We use a cubic spline kernel and a variant of the approach proposed by Band et al. [2020] for the neighborhood search. If not indicated otherwise, we use a snow density of 400 kg m^{-3}. Furthermore, by default, we set $E = 140$ kPa, $\nu = 0.2$, $\xi = 10$, $\theta_c = 0.025$ and $\theta_s = 0.0075$.

4.1 Parameters

4.1.1 Snowballs. We show in a simple scene how different boundary friction coefficients ν_b influence the interaction of snow with the boundary. For this, we drop three snowballs on inclined planes with friction coefficients of 0, 1 and 10. As seen in Fig. 2 and the accompanying video, a friction coefficient of 0 leads to a slip boundary condition while higher values converge to a no-slip behavior. This scene is simulated with a frame rate of 250 frames s^{-1}.

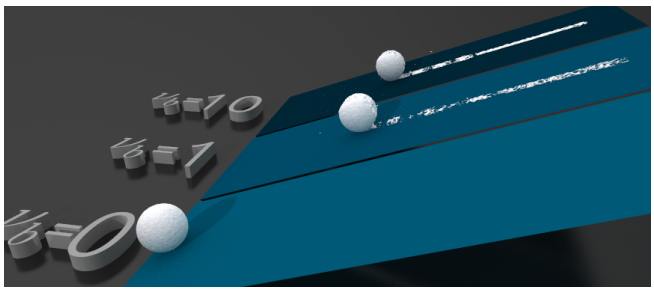


Fig. 2. Snowballs rolling down a inclined planes with different friction coefficients.

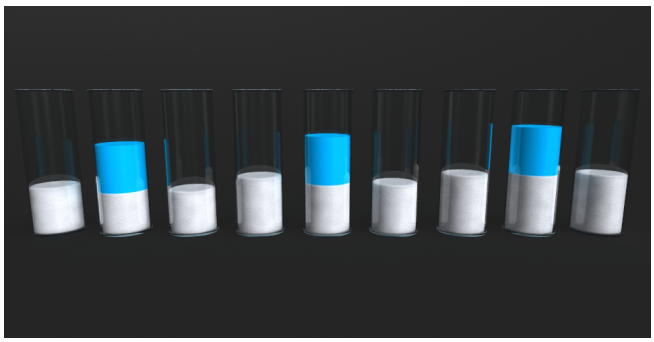


Fig. 3. Compression test with three different settings and three cylinders per setting. The three cylinders on the left show snow with $E = 20$ kPa, the three cylinders in the middle contain snow with $E = 140$ kPa and the three cylinders on the right use an IISPH implementation instead of our compressible pressure solver.

4.1.2 Compression. Figure 3 shows a test scene with snow of varying compressibility in a cylinder in three different cases: under gravity, under an additional load, and after removing the additional load. The level of compression is adjusted by different values for the Young modulus E, which is set to 20 kPa ($\lambda^0 \approx 5.5$ kPa) and 140 kPa ($\lambda^0 \approx 38.8$ kPa). In the third example, the Young modulus is also set to 140 kPa but we replaced the compressible solver with an incompressible pressure solver implementation based on IISPH as proposed by Ihmsen et al. [2014a]. When we use the incompressible SPH variant, the snow height is the same in all three cases. When using the proposed compressible pressure solver, the snow compresses under gravity. It compresses more under the additional load and this compression does not change after removing the additional load. Detailed statistics regarding average number of neighbors and performance of the solvers are shown in Figs. 4 and 5. The average number of particle neighbors increases as the snow is compressed for the $E = 20$ kPa and $E = 140$ kPa cases while it stays constant

when using the incompressible solver. Similarly, memory requirements increases and performance of the solver decreases under load. While we use a variant of the neighborhood search proposed by Band et al. [2020], we do not employ their proposed compression scheme for the neighbor lists. However, it might be interesting to integrate this in the future considering that snow particles have significantly more neighbors when compressed.

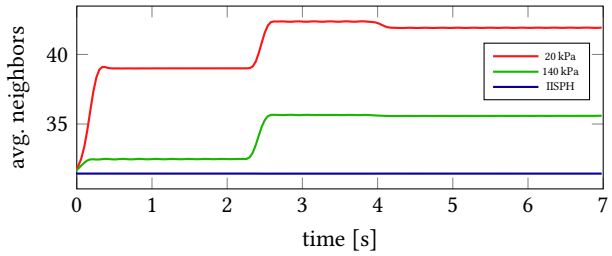


Fig. 4. Average number of neighbors in the compression scene.

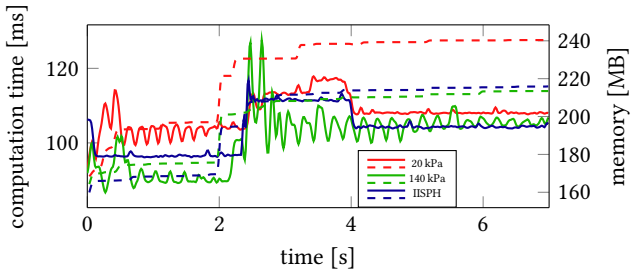


Fig. 5. Performance and memory consumption in the compression scene. Solid lines indicate computation time per simulation step while dashed lines indicate memory consumption.

4.1.3 Tire. As a more realistic experiment to show the effects of varying the boundary friction coefficient ν_b, we simulate a tire rolling through snow. Figure 6 shows close-ups of the two settings, one with $\nu_b = 0$ and one with $\nu_b = 1$. It can be seen that the amount of snow sticking to the tire surface varies drastically and that it is possible to recreate the typical tire profile when rolling through snow. This scene is simulated with a frame rate of 250 frames s^{-1}.

4.1.4 Oven bunnies. We added a particle-based thermodynamics simulation (cf., Brookshaw [1994]; Weiler et al. [2018]) to our solver which enables us to couple snow parameters to temperatures separately for each particle. To illustrate the possible effect, we simulate multiple snow bunnies being put into an oven where they are slowly being heated as shown in Fig. 7. To model the decreasing stiffness of the snow when the temperature increases, we mapped the starting temperature of −100 °C to a Young modulus of 600 kPa and let it decrease to a value of 20 kPa for a temperature of 0 °C.

(a) $\nu_b = 0$ (b) $\nu_b = 1$

Fig. 6. A tire rolls through snow. Depending on the boundary friction coefficient ν_b, different amounts of snow stick to the surface of the tire.

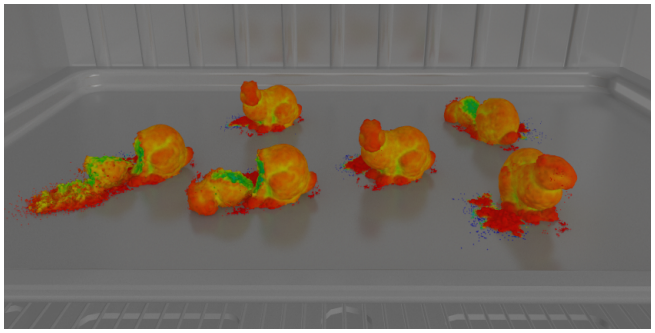


Fig. 7. Snow bunnies are heated inside an oven. The Young modulus of the snow particles varies depending on their temperature. The color of the snow indicates the temperature with blue being −100 °C and red being 0 °C.

4.2 Snow solver

4.2.1 Single particles. As with all discretization approaches, our SPH snow solver does not compute meaningful strains or stresses for single samples with incomplete neighborhood. Without any neighboring sample points, all involved SPH derivatives are evaluated to zero. Nevertheless, single particles, i.e. small snow volumes, move plausibly. Air interaction is handled with the drag force of Gissler et al. [2017]. Solid boundaries are plausibly handled as contributions from missing neighbors are accounted for by the boundary handling of Akinci et al. [2012]. Finally, the advection is straightforward in a purely Lagrangian setting.

If single particles approach each other or larger volumes, the pressure solver starts working and preserves particle volumes. Also, the SPH derivatives start to give more meaningful values. When

particles finally reach a complete neighborhood, their behavior adheres to the constitutive model.

As in many other SPH solvers, e.g. in fluids, single particles behave plausibly, but with limited material properties. Nevertheless, solid boundaries and the interaction with air can be handled. Although many SPH sums compute erroneous values for isolated samples, the sums can always be computed and do not cause stability issues. The transition of single particles to particle clusters with complete neighborhood also does not cause stability issues as illustrated, e.g., in Fig. 8, but also in the snow fall scenes in Fig. 1, Fig. 10 and Fig. 13. Particles with complete neighborhood follow the constitutive model.

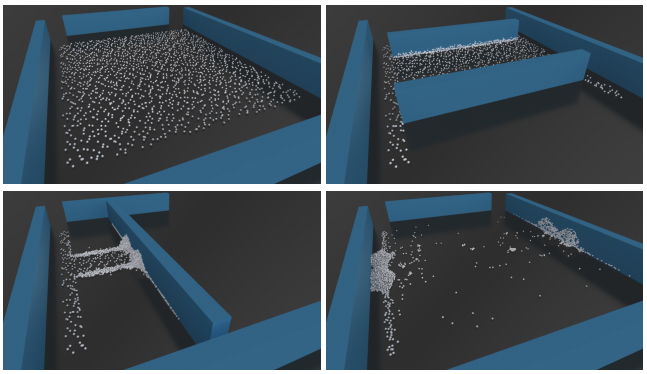


Fig. 8. Demonstration that our solver produces plausible behavior for particles with few neighbors. Initially, all the particles are separated before being formed into snow piles by moving boundaries.

4.2.2 Snow angel. To show the practicality of our approach, we simulate a scene where the snow interacts with an animated mesh. Figure 9 shows an image of the scene where a man walks through snow and then falls on his back to make a snow angel with his arms. The interaction with the animated mesh demonstrates the wide range of motion that our snow solver can interact with and the plausible results that are produced.

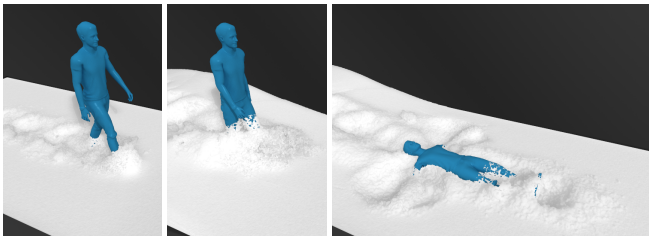


Fig. 9. A man is walking through snow before falling on his back and making a snow angel with his arms.

4.2.3 City street. In this scene, we demonstrate snow accumulation through snow fall on a city street. Then, as shown in Fig. 10, a truck with a snow plow shield clears the street of the city. The snow has a density of 200 $\mathrm{kg\,m^{-3}}$ and overall we simulate the scene for more than 1.5 min of physical time.

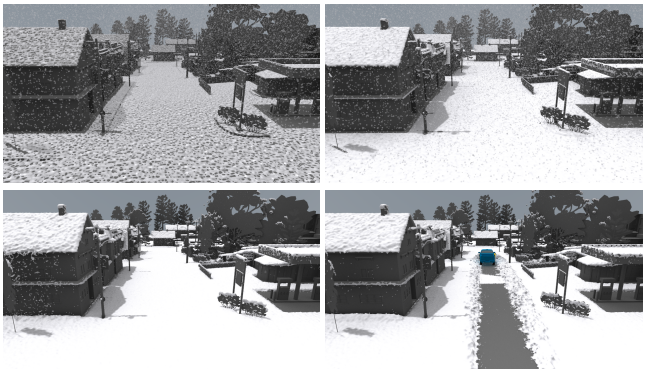
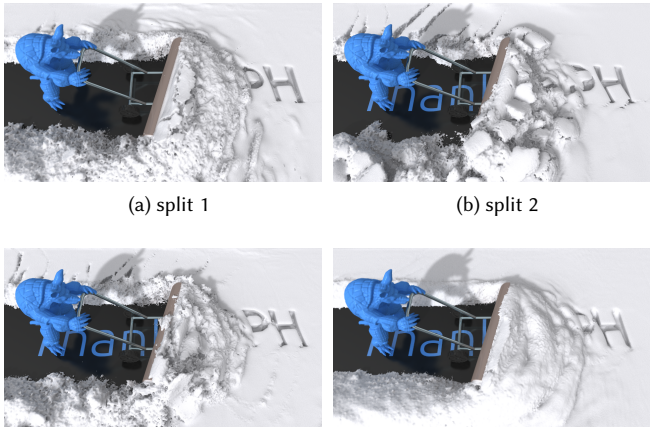


Fig. 10. Snow is falling on a city street before a snow plow truck clears it. Overall, more than 1.5 min of physical time is simulated with up to 2.57M particles.

4.2.4 Armadillo. We simulate the Armadillo pushing a snow shovel to clear snow as seen in Fig. 11. The snow deforms, compresses, fractures and gets harder when compressed.



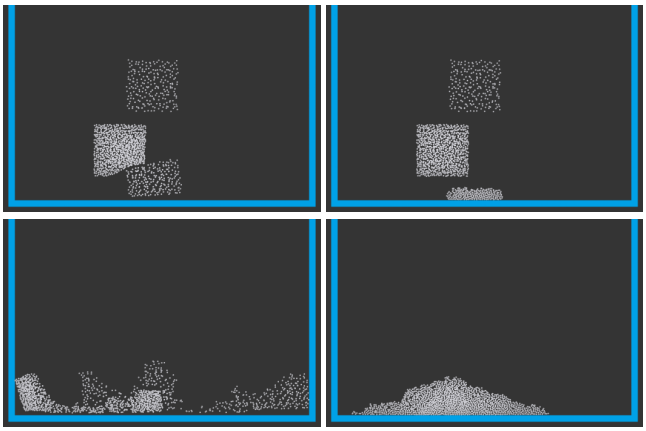
(a) split 1 (b) split 2

(c) combined 1 (d) combined 2

Fig. 11. The Armadillo clears snow using a snow shovel. As discussed in Subsection 4.2.4, we simulate this scene with different parameters and solver variants. (a) and (b) use our approach where two solvers are used to compute the acceleration due to elastic deformation (referred to as *split*). (c) and (d) are examples where only the elastic solver is used to compute the acceleration (referred to as *combined*).

Following the discussion in Subsection 3.5, it is possible to use a single solver to compute the acceleration due to elastic deformation instead of using our proposed split-force approach if coupling with other phases is not required. To evaluate this alternative approach, we simulate the scene shown in Fig. 11 with the combined-force approach where we only use the linear elastic solver. When using the same parameters ($\theta_c = 0.025$, $\theta_s = 0.0075$), the visual result varies slightly as can be seen in the accompanying video and when comparing Fig. 11a and Fig. 11c. However, we show that by slight

adaption of the parameters ($\theta_c = 0.019$, $\theta_s = 0.0075$ for *split 2* and $\theta_c = 0.025$, $\theta_s = 0.0045$ for *combined 2*), the snow behavior can be tuned to be more similar.



(a) (b)

Fig. 12. Loosely and densely sampled snow with Taichi MPM (a) and our SPH approach (b). The loosely and densely sampled MPM particles move globally according to the snow model, but the loosely sampled MPM particles do not pack as in SPH.

4.2.5 Relation to MPM Snow. Our paper bases on the MPM snow solver of Stomakhin et al. [2013]. One major contribution by Stomakhin et al. [2013] is a constitutive model that covers a practically relevant range of snow behavior with an intuitive parameter control. The other major contribution is the discretization of the respective equations with MPM which has advanced the state-of-the-art in MPM in general.

In contrast, we contribute to the state-of-the-art in SPH by solving SPH-specific challenges in the realization of the same material behavior governed by Stomakhin et al.'s [2013] model. We discuss a novel implicit compressible SPH pressure formulation and a novel boundary handling. The relevance of SPH as a promising alternative to MPM is shown in novel scenarios that have not been presented with previously used discretization concepts. In particular, we show snow fall and snow accumulation.

We are hesitant to postulate general benefits or drawbacks of SPH compared to MPM or other discretization concepts. Instead, we see a currently alleged drawback of a concept as a challenge that will be resolved by future research. E.g., Zhu and Bridson [2005] mention considerable computational expenses of MPM which have been significantly reduced by Stomakhin et al. [2013]. Stomakhin et al. [2013], on the other hand, state that SPH has issues in the handling of volume preservation, stiffness, and plasticity. While this was true back then, the proposed SPH formulations improve the situation for SPH. SPH can be used for Stomakhin et al.'s [2013] model and we even extend the range of scenarios that can be handled. To continue that thought, we are rather confident that future MPM research will find solutions for, e.g., snow fall and accumulation. Finally, we are also convinced that MPM and SPH will be used for snow simulations that go beyond the state presented in this paper, as both discretization concepts are subject to continuous improvements.

Nevertheless, the relation between SPH and MPM can be discussed. While the snow behavior is the same for larger volumes in both concepts, single particles seem to require different processing in case of snow fall and accumulation. This is illustrated in a scenario with loosely and densely sampled particles in Fig. 12. The SPH particles naturally accumulate to a larger snow volume with our approach, but the Taichi MPM particles [Hu et al. 2019a] based on the work of Hu et al. [2018] behave differently. As a whole, the loosely and densely sampled MPM particles move globally according to the snow model, but the loosely sampled MPM particles do not pack as in SPH.

We speculate that the difference is due to implementation aspects. Before discussing these aspects, we briefly summarize the concept of Taichi MPM.

Taichi MPM.
Taichi MPM works with two sample sets, Lagrangian samples, i.e. particles, and Eulerian samples, i.e. grid cells. It starts with velocities at particles which are interpolated at grid points. Deformation gradient, strain and stress are considered at particles [Stomakhin et al. 2013, 2014; Hu et al. 2018]. The resulting momentum change, however, is computed at grid points from adjacent particles within the support of a kernel function. The grid velocities are further processed, e.g. boundary handling, and the final grid velocities are used to update the deformation gradients at particles. Here, adjacent grid points within the kernel support of a particle are used. Finally, grid velocities are interpolated at particle positions, particles are advected, and the velocities of the advected particles are interpolated at grid points. This results in updated velocities at both sample sets and an updated deformation gradient at the particles.

Velocity divergence vs. density invariance.
There are two interesting differences in the Taichi MPM implementation compared to the proposed SPH realization. The first one is related to the differential update of the deformation gradient with the velocity gradient which suffers from drift. In particular, the volume estimation is affected by the differential update from the velocity divergence. Although the differential update of the deformation gradient is considered in both settings, Taichi MPM and SPH, it is less frequently applied in our approach. In Taichi MPM, compression is exclusively deduced from the velocity divergence. Also, the hardening, i.e. the update of the Lamé parameters, is deduced from the determinant of the deformation gradient in Taichi MPM without a notion of the actual volume. In contrast, our SPH approach considers the actual particle volume in the hardening process, i.e. in the computation of the Lamé parameters. Further, our pressure solver considers the actually predicted compression which avoids the volume drift. As there is no notion of the actual compression in Taichi MPM, approaching particles might decelerate too early or the hardening might be too strong, if a negative velocity divergence is detected. This is not a conceptual issue. As discussed, e.g., for SPH PPE solvers by Cornelis et al. [2019], the density invariance, the velocity divergence or even combinations can be used in the pressure computation and in the update of the Lamé parameters. Thus, we speculate that the hardening and compression handling

in MPM could also employ the actual density deviation instead of exclusively relying on the velocity divergence.

Two sample sets.
A second interesting aspect is the usage of two staggered sample sets in MPM. Particle velocities are interpolated at grid points within the kernel support which is typically two times the grid cell edge length, e.g. as done by Hu et al. [2019a]. Thus, a 2D particle affects velocities at 3×3 nearby grid points. Also, the (processed) velocities of 3×3 nearby grid points are used to update the deformation gradient at a particle. I.e., whenever two particles in adjacent cells move towards each other, the respective negative velocity divergence is encoded as a compression in the deformation gradient at both particles independent from their actual volume. This aspect is similar to SPH, where particles within the kernel support distance influence each other. Nevertheless, the kernel support in MPM tends to be larger than in SPH. If, e.g., the 2D cell area is four times the size of a 2D particle, the MPM kernel support would be four times the particle diameter, while it is two times the diameter in our SPH implementation.

4.2.6 Density computation. As already discussed in Subsection 4.2.5, it is critical for the snow accumulation behavior how the current compression and resulting hardening of the snow is computed. To further illustrate this, we simulate a simple snow fall scene as shown in Fig. 13. The density of particles falling into the left cylinder is handled as shown in Eq. (21). In contrast, the rest density of the snow on the right-hand side is computed based on the determinant of the plastic deformation gradient which we additionally accumulated as proposed by Stomakhin et al. [2013]. This scene is simulated with a frame rate of 250 frames s^{-1}.

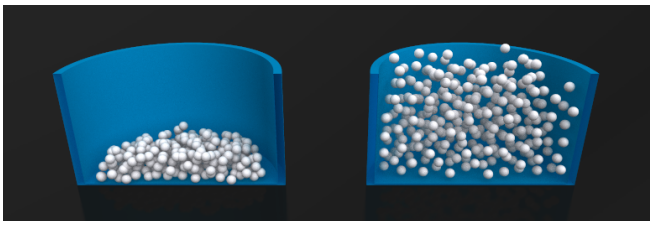


Fig. 13. Falling snow accumulates differently depending on how the current rest density of a particle is computed.

4.3 Coupling

4.3.1 Car snowing. Figure 1 shows a scene in which we simulate snow fall on a car. Similar to the city scene, this scene demonstrates that our solver is able to simulate single particles that accumulate over time. Furthermore, the animated wipers and the car that drives away at the end illustrate the ability of the snow to interact with complex and moving geometries. The scene furthermore contains an air flow and the snow-air interactions are computed based on the drag force model presented by Gissler et al. [2017]. To model the light snow flakes with single particles, we set the snow density to 40 kg m^{-3}.

4.3.2 Sled race. We demonstrate two-way coupling with the SPH-based rigid body solver proposed by Gissler et al. [2019] by simulating sleds that drive down a hill. A frame of the scene is shown in Fig. 14. The sleds are simulated as full rigid bodies with six degrees of freedom.

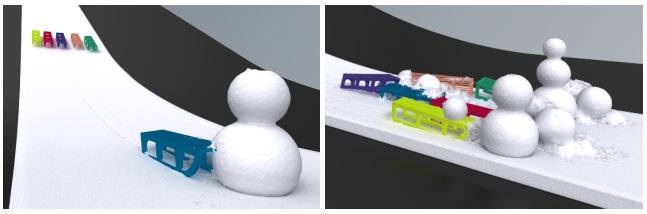


Fig. 14. Sleds are racing down a hill before crashing into snowmen. This scenes demonstrates two-way coupling with rigid bodies.

4.3.3 Submarine. To illustrate the interaction between snow and fluid, we simulate a submarine that breaks through a layer of ice as can be seen in Fig. 15. In this scene, we set the density of the snow to 100 kg m^{-3} and increased the Young modulus to 1 MPa to get a harder snow that behaves more like ice. The water is simulated using the IISPH approach by Ihmsen et al. [2014a].

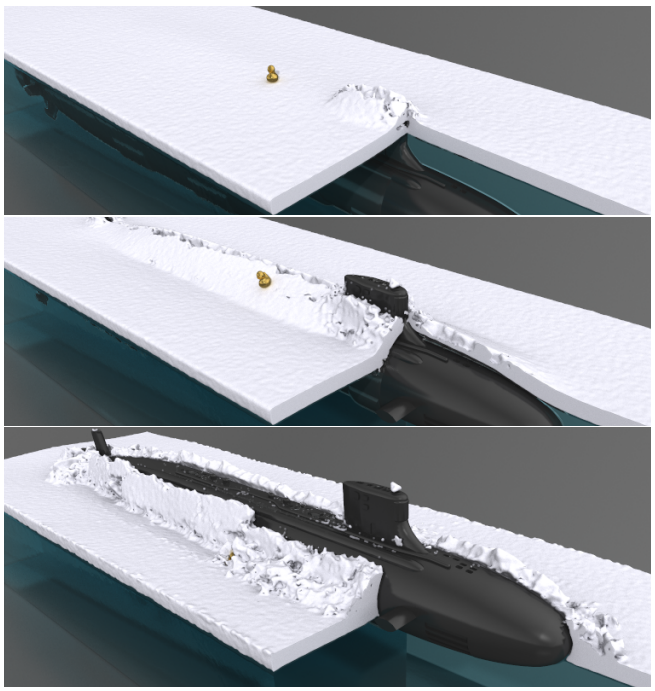


Fig. 15. A submarine breaks through ice which demonstrates the interaction of an incompressible fluid phase with snow.

4.3.4 Ice machine. We demonstrate phase change from viscous fluid to snow and phase interaction between viscous fluid, rigid body objects and snow in the scene shown in Fig. 16. An ice cone,

simulated as rigid body, is filled with a viscous fluid which turns to snow and then is covered with chocolate sauce simulated as viscous fluid.

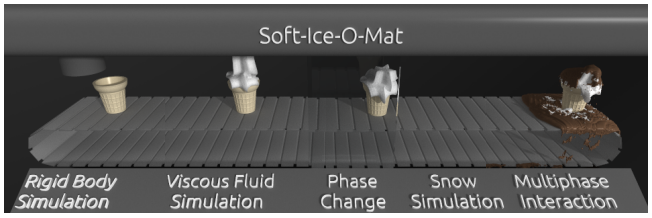


Fig. 16. Ice is filled into a cone and is then covered with chocolate sauce.

5 LIMITATIONS

As mentioned in Subsection 3.5, our proposed splitting of the force computation leads to a decoupling of pressure and shear stress, i.e., our method is not fully implicit. In particular, since we first compute the pressure and then the shear forces as detailed in Algorithm 1, it might be that Eq. (5) is not fully satisfied at the end of a time step. In practice, however, we did not perceive this to be a problem, e.g., as shown in Subsection 4.2.4.

Furthermore, while we compute the snow behavior with two implicit solvers, we update the deformation gradient in the integration step and do not consider the plastic deformation during the implicit solver iterations. While we do not observe any time step related instabilities due to this in our test scenes, it might be that the overall snow behavior depends on the used time step due to this explicit plastic deformation update. It therefore would be interesting to integrate more advanced physical models, e.g., similar to the one proposed by Gaume et al. [2018, 2019], into our snow solver.

We presented an implicit friction formulation at the boundary and additionally use an inter-particle adhesion force. This lets us model different types of boundary interactions for the snow. However, in reality, single snow flakes or ice crystals often bounce away after first impacting on a surface. We cannot parametrize this behavior with our current boundary handling approach. It would be beneficial to be able to control the strength of the bounce using for example a restitution coefficient.

6 CONCLUSION

We presented a novel SPH-based snow solver that combines two implicit solvers. This combination of two solvers allows to simulate snow and its interactions with other phases such as fluids. Using SPH for the discretization simplifies the simulation of snow fall and snow accumulation where single snow flakes are modeled as single particles. Furthermore, we presented an implicit friction formulation at the particle-based boundary. Overall, this enabled us to efficiently simulate a wide range of snow effects including snow fall and accumulation, deformation, breaking, compression and hardening, phase change and phase interactions and boundary interactions with complex, moving and deforming boundaries.

The implicit friction formulation and adhesion at the boundary enables us to show different sticking behavior but we cannot parametrize a bounce effect of single particles as discussed in Section 5. Accordingly, we plan to further work on the boundary handling of snow in the future. We are also interested in trying the implicit friction at the boundary for other use cases, e.g., for the boundary friction of elastic solids based on the approach of Peer et al. [2018] or for SPH-based rigid bodies based on the work of Gissler et al. [2019]. Furthermore, we think our proposed implicit compressible pressure solver can also be used for other applications and scenarios. For example, we plan to employ it for the simulation of compressible gases such as air. We think investigating its features and applying this novel solver to other use cases will inspire future research. Finally, given our coupling with fluids, it would be natural to let snow melt into water. However, we think this requires being able to simulate an SPH fluid with varying particle sizes. We believe that this could be an interesting future research direction, e.g., based on the work of Winchenbach et al. [2017].

ACKNOWLEDGMENTS

We thank the team at FIFTY2 Technology GmbH for their support. In particular, we would like to thank Marc Gißler for implementing the thermodynamics solver, Fabian Meyer for his insights with regard to MPM and Markus Ihmsen for his support. We also want to thank Jennifer Weiche from AVL for her early feedback. We thank the reviewers for their suggestions and proof-reading that helped improve this work.

The walking man is courtesy of Renderpeople. The ice cone by 2ROBOTGUY is licensed under CC BY 3.0. The sled by fragar is licensed under CC BY 3.0. The duck by willie is licensed under CC0 1.0. The truck is courtesy of 1991 Chevy on 3dwarehouse.sketchup.com. The city (upper and lower part) is courtesy of Demilune on 3dwarehouse.sketchup.com. The Stanford Bunny and Armadillo are courtesy of the Stanford University Computer Graphics Laboratory. The gas oven by Francesco Coldesina is licensed under CC BY 4.0. The submarine by ate135 is licensed under CC BY 3.0. The rim is courtesy of BBS Motorsport GmbH. Additional models are either bought or created by us using Blender by the Blender Online Community [2020].

REFERENCES

Ahmed M. Abdelrazek, Ichiro Kimura, and Yasuyuki Shimizu. 2014. Numerical simulation of a small-scale snow avalanche tests using non-Newtonian SPH model. *Transactions of the Japan Society of Civil Engineers* 70, 2 (2014), 681–690.

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

Nadir Akinci, Gizem Akinci, and Matthias Teschner. 2013. Versatile Surface Tension and Adhesion for SPH Fluids. *ACM Transactions on Graphics* 32, 6, Article 182 (2013), 8 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, Article 62 (2012), 8 pages.

Stefan Band, Christoph Gissler, Markus Ihmsen, Jens Cornelis, Andreas Peer, and Matthias Teschner. 2018a. Pressure Boundaries for Implicit Incompressible SPH. *ACM Transactions on Graphics* 37, 2, Article 14 (Feb. 2018), 11 pages.

Stefan Band, Christoph Gissler, Andreas Peer, and Matthias Teschner. 2018b. MLS pressure boundaries for divergence-free and viscous SPH fluids. *Computers & Graphics* 76 (2018), 37–46.

Stefan Band, Christoph Gissler, and Matthias Teschner. 2020. Compressed Neighbour Lists for SPH. *Computer Graphics Forum* 39, 1 (2020), 531–542.

Markus Becker, Markus Ihmsen, and Matthias Teschner. 2009. Corotated SPH for Deformable Solids. In *Proceedings of the Fifth Eurographics Conference on Natural Phenomena* (Munich, Germany) *(NPH '09)*. Eurographics Association, Aire-la-Ville, Switzerland, 27–34.

Markus Becker and Matthias Teschner. 2007. Weakly compressible SPH for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*, Dimitris Metaxas and Jovan Popovic (Eds.). The Eurographics Association, Aire-la-Ville, Switzerland, 209–218.

Nathan Bell, Yizhou Yu, and Peter J. Mucha. 2005. Particle-Based Simulation of Granular Materials. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '05)*. Association for Computing Machinery, New York, NY, USA, 77–86.

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.

Blender Online Community. 2020. Blender. http://www.blender.org.

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 (Nov. 1999), 97–115.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4, Article 154 (July 2014), 11 pages. https://doi.org/10.1145/2601097.2601116

Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-Reduced Projective Dynamics. *ACM Trans. Graph.* 37, 4, Article 80 (July 2018), 13 pages.

Christopher Brandt, Leonardo Scandolo, Elmar Eisemann, and Klaus Hildebrandt. 2019. The Reduced Immersed Method for Real-Time Fluid-Elastic Solid Interaction and Contact Simulation. *ACM Trans. Graph.* 38, 6, Article 191 (Nov. 2019), 16 pages.

L. Brookshaw. 1994. Solving the Heat Diffusion Equation in SPH. *Memorie della Società Astronomia Italiana* 65 (Jan 1994), 1033.

G. Cordonnier, P. Ecormier, E. Galin, J. Gain, B. Benes, and M.-P. Cani. 2018. Interactive Generation of Time-evolving, Snow-Covered Landscapes with Avalanches. *Computer Graphics Forum* 37, 2 (2018), 497–509.

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.

François Dagenais, Jonathan Gagnon, and Eric Paquette. 2016. An efficient layered simulation workflow for snow imprints. *The Visual Computer* 32, 6 (June 2016), 881–890.

Mathieu Desbrun and Marie-Paule Gascuel. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation '96*, Vol. 96. Springer, Springer Vienna, Vienna, 61–76.

Yu Fang, Yuanming Hu, Shi-Min Hu, and Chenfanfu Jiang. 2018. A Temporally Adaptive Material Point Method with Regional Time Stepping. *Computer Graphics Forum* 37, 8 (2018), 195–204.

Yu Fang, Minchen Li, Ming Gao, and Chenfanfu Jiang. 2019. Silly Rubber: An Implicit Material Point Method for Simulating Non-Equilibrated Viscoelastic and Elastoplastic Solids. *ACM Trans. Graph.* 38, 4, Article 118 (July 2019), 13 pages.

Paul Fearing. 2000. Computer Modelling of Fallen Snow. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 37–46.

Bryan E. Feldman and James F. O'Brien. 2002. Modeling the Accumulation of Wind-Driven Snow. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications* (San Antonio, Texas) *(SIGGRAPH '02)*. Association for Computing Machinery, New York, NY, USA, 218.

Niels v. Festenberg and Stefan Gumhold. 2009. A Geometric Algorithm for Snow Distribution in Virtual Scenes. In *Eurographics Workshop on Natural Phenomena*, Eric Galin and Jens Schneider (Eds.). The Eurographics Association, Aire-la-Ville, Switzerland, 17–25.

Niels v. Festenberg and Stefan Gumhold. 2011. Diffusion-Based Snow Cover Generation. *Computer Graphics Forum* 30, 6 (2011), 1837–1849.

FIFTY2 Technology GmbH. 2020. PreonLab. https://fifty2.eu/.

Ming Gao, Xinlei Wang, Kui Wu, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. 2018. GPU Optimization of Material Point Methods. *ACM Trans. Graph.* 37, 6, Article 254 (Dec. 2018), 12 pages.

Theodore F. Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M. Teran. 2015. Optimization Integrator for Large Time Steps. *IEEE Transactions on Visualization and Computer Graphics* 21, 10 (Oct. 2015), 1103–1115.

J. Gaume, T. Gast, J. Teran, A. van Herwijnen, and C. Jiang. 2018. Dynamic anticrack propagation in snow. *Nature Communications* 9, 1 (08 2018), 681–690.

Johan Gaume, Alec van Herwijnen, Ted Gast, Joseph Teran, and Chenfanfu Jiang. 2019. Investigating the release and flow of snow avalanches at the slope-scale using a unified model based on the material point method. *Cold Regions Science and Technology* 168 (2019), 102847.

Dan Gerszewski, Haimasree Bhattacharya, and Adam W. Bargteil. 2009. A Point-Based Method for Animating Elastoplastic Solids. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New Orleans, Louisiana) *(SCA '09)*. Association for Computing Machinery, New York, NY, USA, 133–138.

Christoph Gissler, Stefan Band, Andreas Peer, Markus Ihmsen, and Matthias Teschner. 2017. Generalized drag force for particle-based simulations. *Computers & Graphics* 69 (2017), 1–11.

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, Article 5 (2019), 13 pages.

Prashant Goswami, Christian Markowicz, and Ali Hassan. 2019. Real-time Particle-based Snow Simulation on the GPU. In *Eurographics Symposium on Parallel Graphics and Visualization*, Hank Childs and Steffen Frey (Eds.). The Eurographics Association, Aire-la-Ville, Switzerland, 49–57.

Håkan Haglund, Mattias Andersson, and Anders Hast. 2002. Snow Accumulation in Real-time. In *Proceedings from SIGRAD 2002*. Linköping University Electronic Press; Linköpings universitet, Linköping, Sweden, 11–15.

David Hahn and Chris Wojtan. 2015. High-Resolution Brittle Fracture Simulation with Boundary Elements. *ACM Trans. Graph.* 34, 4, Article 151 (July 2015), 12 pages.

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.

Tommy Hinks and Ken Museth. 2009. Wind-driven Snow Buildup Using a Level Set Approach. In *Eurographics Ireland Workshop Series*, Vol. 9. The Eurographics Association, Aire-la-Ville, Switzerland, 19–26.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, and Chenfanfu Jiang. 2019a. Taichi MPM: High-Performance MLS-MPM Solver with Cutting and Coupling (CPIC). https://github.com/yuanming-hu/taichi_mpm.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A Moving Least Squares Material Point Method with Displacement Discontinuity and Two-Way Rigid Body Coupling. *ACM Trans. Graph.* 37, 4, Article 150 (July 2018), 14 pages.

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019b. Taichi: A Language for High-Performance Computation on Spatially Sparse Data Structures. *ACM Transactions on Graphics* 38, 6, Article 201 (Nov. 2019), 16 pages.

Libo Huang, Torsten Hädrich, and Dominik L. Michels. 2019. On the Accurate Large-Scale Simulation of Ferrofluids. *ACM Trans. Graph.* 38, 4, Article 93 (July 2019), 15 pages.

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

Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. 2014b. SPH Fluids in Computer Graphics. In *Eurographics (State of the Art Reports)*. The Eurographics Association, Aire-la-Ville, Switzerland, 21–42.

Ben Jones, April Martin, Joshua A. Levine, Tamar Shinar, and Adam W. Bargteil. 2016a. Ductile Fracture for Clustered Shape Matching. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Redmond, Washington) *(I3D '16)*. Association for Computing Machinery, New York, NY, USA, 65–70.

Ben Jones, Nils Thuerey, Tamar Shinar, and Adam W. Bargteil. 2016b. Example-Based Plastic Deformation of Rigid Bodies. *ACM Trans. Graph.* 35, 4, Article 34 (July 2016), 11 pages.

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*, Wenzel Jakob and Enrico Puppo (Eds.). The Eurographics Association, Aire-la-Ville, Switzerland, 1–41.

Nipun Kwatra, Jonathan Su, Jón T. Grétarsson, and Ronald Fedkiw. 2009. A method for avoiding the acoustic time step restriction in compressible flow. *J. Comput. Phys.* 228, 11 (2009), 4146–4161. https://doi.org/10.1016/j.jcp.2009.02.027

Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. 2006. Multiple Interacting Liquids. In *ACM SIGGRAPH 2006 Papers* (Boston, Massachusetts) *(SIGGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 812–819.

Günther Meschke, Changhong Liu, and Herbert A. Mang. 1996. Large Strain Finite-Element Analysis of Snow. *Journal of Engineering Mechanics* 122, 7 (1996), 591–602.

J. J. Monaghan. 2005. Smoothed particle hydrodynamics. *Reports on Progress in Physics* 68, 8 (July 2005), 1703–1759.

J. J. Monaghan. 2012. Smoothed Particle Hydrodynamics and Its Diverse Applications. *Annual Review of Fluid Mechanics* 44, 1 (2012), 323–346.

Joseph P. Morris, Patrick J. Fox, and Yi Zhu. 1997. Modeling Low Reynolds Number Incompressible Flows Using SPH. *J. Comput. Phys.* 136, 1 (Sept. 1997), 214–226.

Nobuhiko Mukai, Yusuke Eto, and Youngha Chang. 2017. Representation Method of Snow Splitting and Sliding on a Roof. In *5th International Conference on Advances in Engineering and Technology*. Eminent Association of Researchers in Engineering & Technology (EARET), London, UK, 100–103.

Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based Fluid Simulation for Interactive Applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '03)*. Eurographics Association, Aire-la-Ville, Switzerland, 154–159.

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.

Tomoyuki Nishita, Hiroshi Iwasaki, Yoshinori Dobashi, and Eihachiro Nakamae. 1997. A Modeling and Rendering Method for Snow by Using Metaballs. *Computer Graphics Forum* 16, 3 (1997), C357–C364.

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

Andreas Peer, Markus Ihmsen, Jens Cornelis, and Matthias Teschner. 2015. An Implicit Viscosity Formulation for SPH Fluids. *ACM Transactions on Graphics* 34, 4 (2015), 114:1–114:10.

D. T. Reynolds, S. D. Laycock, and A. M. Day. 2015. Real-Time Accumulation of Occlusion-Based Snow. *The Visual Computer* 31, 5 (May 2015), 689–700.

Barbara Solenthaler and Renato Pajarola. 2008. Density Contrast SPH Interfaces. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Aire-la-Ville, Switzerland, 211–218.

Barbara Solenthaler and Renato Pajarola. 2009. Predictive-Corrective Incompressible SPH. *ACM Transactions on Graphics* 28, 3, Article 40 (July 2009), 6 pages.

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, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A Material Point Method for Snow Simulation. *ACM Trans. Graph.* 32, 4, Article 102 (July 2013), 10 pages.

Alexey Stomakhin, Craig Schroeder, Chenfanfu Jiang, Lawrence Chai, Joseph Teran, and Andrew Selle. 2014. Augmented MPM for Phase-Change and Varied Materials. *ACM Trans. Graph.* 33, 4, Article 138 (July 2014), 11 pages.

Robert W. Sumner, James F. O'Brien, and Jessica K. Hodgins. 1999. Animating Sand, Mud, and Snow. *Computer Graphics Forum* 18, 1 (1999), 17–26.

Tetsuya Takahashi, Yoshinori Dobashi, Issei Fujishiro, Tomoyuki Nishita, and Ming C. Lin. 2015. Implicit Formulation for SPH-Based Viscous Fluids. *Comput. Graph. Forum* 34, 2 (May 2015), 493–502.

Tetsuya Takahashi and Issei Fujishiro. 2012. Particle-based Simulation of Snow Trampling Taking Sintering Effect into Account. In *ACM SIGGRAPH 2012 Posters* (Los Angeles, California) *(SIGGRAPH '12)*. Association for Computing Machinery, New York, NY, USA, Article 7, 1 pages.

Andre Pradhana Tampubolon, Theodore Gast, Gergely Klár, Chuyuan Fu, Joseph Teran, Chenfanfu Jiang, and Ken Museth. 2017. Multi-Species Simulation of Porous Sand and Water Mixtures. *ACM Trans. Graph.* 36, 4, Article 105 (July 2017), 11 pages.

H. A. van der Vorst. 1992. Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM J. Sci. Statist. Comput.* 13, 2 (1992), 631–644.

Changbo Wang, Zhangye Wang, Tian Xia, and Qunsheng Peng. 2006. Real-time snowing simulation. *The Visual Computer* 22, 5 (May 2006), 315–323.

Stephanie Wang, Mengyuan Ding, Theodore F. Gast, Leyi Zhu, Steven Gagniere, Chenfanfu Jiang, and Joseph M. Teran. 2019. Simulation and Visualization of Ductile Fracture with the Material Point Method. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 18 (July 2019), 20 pages.

Marcel Weiler, Dan Koschier, and Jan Bender. 2016. Projective Fluids. In *Proceedings of the 9th International Conference on Motion in Games* (Burlingame, California) *(MIG '16)*. Association for Computing Machinery, New York, NY, USA, 79–84. https://doi.org/10.1145/2994258.2994282

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), 145–155.

Rene Winchenbach, Hendrik Hochstetter, and Andreas Kolb. 2017. Infinite Continuous Adaptivity for Incompressible SPH. *ACM Transactions on Graphics* 36, 4, Article 102 (2017), 10 pages.

Chris Wojtan, Nils Thürey, Markus Gross, and Greg Turk. 2009. Deforming Meshes That Split and Merge. *ACM Transactions on Graphics* 28, 3, Article 76 (July 2009), 10 pages.

Joshuah Wolper, Yu Fang, Minchen Li, Jiecong Lu, Ming Gao, and Chenfanfu Jiang. 2019. CD-MPM: Continuum Damage Material Point Methods for Dynamic Fracture Animation. *ACM Trans. Graph.* 38, 4, Article 119 (July 2019), 15 pages.

Sai-Keung Wong and I-Ting Fu. 2015. Hybrid-based snow simulation and snow rendering with shell textures. *Computer Animation and Virtual Worlds* 26, 3-4 (2015), 413–421.

Joel Wretborn, Rickard Armiento, and Ken Museth. 2017. Animation of crack propagation by means of an extended multi-body solver for the material point method. *Computers & Graphics* 69 (2017), 131–139.

Yongning Zhu and Robert Bridson. 2005. Animating Sand as a Fluid. *ACM Trans. Graph.* 24, 3 (July 2005), 965–972.