Conformation Constraints for Efficient Viscoelastic Fluid Simulation

HÉCTOR BARREIRO, URJC Madrid
IGNACIO GARCÍA-FERNÁNDEZ, Universitat de Valencia
IVÁN ALDUÁN, Next Limit
MIGUEL A. OTADUY, URJC Madrid

Fig. 1. A complex multiphysics simulation involving viscoelastic fluids, rigid bodies, and deformable bodies. We simulate whipped cream and strawberry syrup efficiently using our novel viscoelasticity model based on conformation constraints. The complete scene consists of 150,000 particles and runs at 1.13 seconds per frame.

The simulation of high viscoelasticity poses important computational challenges. One is the difficulty to robustly measure strain and its derivatives in a medium without permanent structure. Another is the high stiffness of the governing differential equations. Solutions that tackle these challenges exist, but they are computationally slow. We propose a constraint-based model of viscoelasticity that enables efficient simulation of highly viscous and viscoelastic phenomena. Our model reformulates, in a constraint-based fashion, a constitutive model of viscoelasticity for polymeric fluids, which defines simple governing equations for a conformation tensor. The model can represent a diverse palette of materials, spanning elastoplastic, highly viscous, and inviscid liquid behaviors. In addition, we have designed a constrained dynamics solver that extends the position-based dynamics method to handle efficiently both position-based and velocity-based constraints. We show results that range from interactive simulation of viscoelastic effects to large-scale simulation of high viscosity with competitive performance.

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

Additional Key Words and Phrases: Position Based Dynamics, Positon Based Fluids, Viscoelastic Fluid, High Viscosity

ACM Reference format:
Héctor Barreiro, Ignacio García-Fernández, Iván Alduán, and Miguel A. Otaduy. 2017. Conformation Constraints for Efficient Viscoelastic Fluid Simulation. *ACM Trans. Graph.* 36, 6, Article 221 (November 2017), 11 pages. https://doi.org/10.1145/3130800.3130854

This work is supported by Ministerio de Economía y Competitividad of Spain, under Contrato Torres Quevedo PTQ-15-07401 and Proyecto Retos TIN2015-70799-R.
Author's addresses: hector.barreiro@urjc.es; ignacio.garcia@uv.es; ivan.alduan@nextlimit.com; miguel.otaduy@urjc.es .

0730-0301/2017/11-ART221 $15.00
https://doi.org/10.1145/3130800.3130854

1 INTRODUCTION

Many real-world substances and materials exhibit a viscous fluid or viscoelastic behavior. As a result, the simulation of viscoelastic fluids has attracted important attention in computer graphics [Bargteil et al. 2007; Batty and Bridson 2008; Carlson et al. 2002; Goktekin et al. 2004; Larionov et al. 2017; O'Brien et al. 2002; Paiva et al. 2009; Peer et al. 2015; Ram et al. 2015; Takahashi et al. 2015; Terzopoulos and Fleischer 1988; Wojtan and Turk 2008; Yue et al. 2015]. Simulation of high viscosity is a computationally challenging problem, since it requires implicit formulations in order to robustly solve the numerically stiff differential equations. In addition, due to the difficulty in computing the strain of a fluid, numerical drift turns into perceptible loss of viscoelasticity.

A common alternative to the solution of stiff equations is to model stiff properties as constraints, thereby effectively removing degrees of freedom from the simulation. Constrained dynamics formulations have proven successful for the simulation, among others, of articulated bodies [Baraff 1996], contact [Baraff 1989; Kaufman et al. 2008], deformation limits [Wang et al. 2010], inextensibility [Goldenthal et al. 2007], volume preservation for solids [Irving et al. 2007], fluid incompressibility [Foster and Metaxas 1996; Macklin and Müller 2013; Solenthaler and Pajarola 2009], or even generic dynamic deformations [Bender et al. 2014; Müller et al. 2006; Stam 2009].

In this paper, we propose a constraint-based solution for fluid viscoelasticity. Our solution is inspired by a constitutive model of polymeric fluids (i.e., fluids where elastic polymers are dissolved), which supports a large range of viscoelasticity behaviors under one common formulation [Deshpande et al. 2010]. We describe the state of the fluid using a conformation tensor, which is evolved in time as a function of the ratio between elastic and viscous forces. By enforcing implicit, velocity-based constraints on the conformation tensor, as described in Section 3, we achieve high viscoelasticity.

Using just two intuitive physically based parameters, the artist may choose from a palette of materials that range between elastoplastic, highly viscous fluid, and inviscid liquid.

With our constraint-based viscoelasticity model, simulation efficiency is determined by the choice of constrained dynamics solver. In Section 4, we propose a *doubly constrained* position-based dynamics (DC-PBD) solver, which inherits the robustness under constraint nonlinearity and the per-iteration efficiency of the original PBD method [Müller et al. 2006], but exhibits improved stability under velocity-based constraints, such as those in our viscoelasticity formulation, especially with large time steps. As a corollary, while others have also accounted for viscosity in PBD solvers [Alduán et al. 2016; Macklin and Müller 2013; Takahashi et al. 2016, 2014], our method is derived from a constitutive model, hence it allows physics-based parameterization. Its range of behaviors is also superior, comparable to the one obtained with methods that also compute viscoelastic stress from a discretization of constitutive models, albeit at a much lower computational cost. In Section 5, we describe in detail the integration of our viscoelasticity formulation in a position-based fluids (PBF) model [Macklin and Müller 2013].

We show results that range from interactive simulation of viscoelastic effects (Fig. 10) to large-scale simulation of high viscosity with competitive performance (Fig. 11). The materials exhibit the classic buckling and coiling effects produced by viscoelasticity (Fig. 9), and we also show how our method can be integrated seamlessly in rich multiphysics scenarios (Fig. 1).

2 RELATED WORK

The modeling and simulation of complex fluid effects has been a topic of research in computer graphics for many years. The dynamics of complex fluids, those that exhibit high viscosity or nonlinear strain-stress relationships, have caught the interest of several researchers, as many interesting effects and behaviors are unique to these types of fluids.

The first attempts to model such effects relied on grid-based discretizations [Terzopoulos and Fleischer 1988]. Carlson et al. [2002] simulated highly viscous fluid materials and melting effects using an implicit viscosity formulation over a Marker-and-Cell (MAC) approach. Goktekin et al. [2004] extended the Navier-Stokes equations incorporating elastic and plastic terms to simulate viscoelastic fluids over a level-set discretization. Batty and Bridson [2008] designed an accurate method for the simulation of characteristic effects in free-surface viscous fluids, such as buckling and coiling. They emphasized the formulation of correct boundary conditions, outlined a variational formulation of viscosity, and designed an efficient solver. Wojtan and Turk [2008] enabled the preservation of thin viscoelastic features in finite element simulations. Recently, Larionov et al. [2017] have proposed an implicit Stokes solver to simulate highly viscous Newtonian fluids also using a grid-based discretization. The focus of these methods has been to enable complex effects, without particular attention to computational performance.

Some works have designed solutions optimized for the simulation of specific viscosity effects. Bergou et al. [2010] and Batty et al. [2012] focused on the simulation of viscous threads and thin layers, respectively. Remeshing strategies helped them preserve thin surfaces and reduce the simulation cost. Zhu et al. [2015] simulated viscous effects on features of different dimensions, all handled in a uniform manner. They achieved high-quality simulation of very thin features for non-Newtonian fluids.

The Material Point Method (MPM) has recently gained attention for the simulation of various effects and materials, including viscoelasticity. It can be regarded as a particle-based method, although it uses a background grid for certain computations. Stomakhin et al. [2013] first applied it to snow simulation, and later they extended it to phase changes and high viscosity [Stomakhin et al. 2014]. Ram et al. [2015] used the MPM formalism to simulate viscoelastic materials, while Yue et al. [2015] applied it for the simulation of foam. More recently, this method has been applied to the simulation of sand [Daviet and Bertails-Descoubes 2016; Klár et al. 2016] and the interaction between sand and water [Tampubolon et al. 2017]. While MPM has gained popularity due to its high-quality results, it is computationally intensive.

Particle-based discretizations offer the ability to simulate highly-deformable materials and thin features with good computational performance. The smoothed particle hydrodynamics (SPH) discretization is one example. Solenthaler et al. [2007] proposed a unified model to simulate melting and solidification effects. They incorporated an elastic force term based on a strain measure, inspired by the previous work of Müller et al. [2004]. Paiva et al. [2006; 2009] used an XSPH velocity correction to simulate non-Newtonian fluids, which has been extended by Andrade et al. [2015] to reproduce buckling in Newtonian viscous fluids. Chang et al. [2009], on the other hand, used an SPH discretization of the elastic strain tensor to simulate viscoelastic behavior. He et al. [2012] used the SPH approximation to solve the Poisson equation locally and simulate moderately viscous fluids. Granular media can be considered a special type of non-Newtonian fluid. Lenaerts et al. [2009] extended the work of Solenthaler et al. [2007] to simulate granular behavior, and Alduan and Otaduy [2011] proposed strain-rate-based models for granular friction and cohesion effects.

Recently, several authors have designed implicit SPH solvers for the simulation of highly viscous fluids. Bender and Koshier [2017] propose a divergence-free SPH solver that also supports viscous materials. Takahashi et al. [2015] enforce incompressibility in simulations with high viscosity by solving for pressure implicitly. Peer et al. [2015] simulate high viscosity by canceling shear rate in a least squares manner. For each particle, they set a goal velocity gradient that cancels the local shear rate, and they solve for the velocity field that best matches the goal velocity gradients. Even though their method computes the velocity field very efficiently, we propose an even more efficient and versatile solution, which handles larger time steps thanks to a PBD-style constrained dynamics solver, and supports more diverse viscoelastic behaviors. Peer et al. [2017] have later extended their method to improve vorticity handling.

The addition of elastic effects to viscosity simulations requires in principle knowledge of undeformed configurations to simplify the computation of deformation metrics. However, maintaining this knowledge becomes complicated under plastic flow. Clavet et al. [2005] used a spring-based approach to apply elastic forces in a particle-based fluid simulation, with a strategy to create and remove elastic links between particles as the particle neighborhoods evolve.

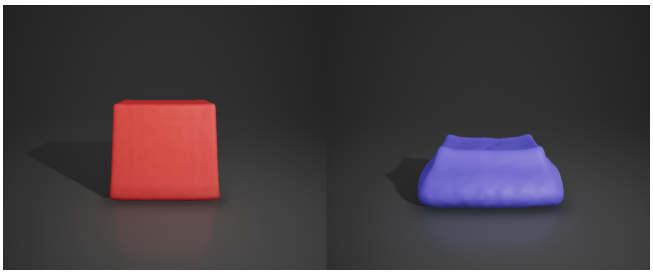


Fig. 2. We drop a block of highly viscous material on the ground using our method (left) and the viscous PBF method of Takahashi et al. [2014] (right). Their method models viscosity using length constraints, which fail to prevent drift without suffering unwanted elastic oscillations.

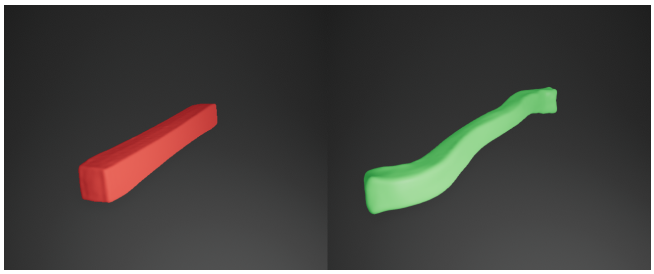


Fig. 3. A beam of highly viscous material, rotating around the vertical axis in absence of external forces, using our method (left) and a PBF simulation with a viscosity method based on that of Peer et al. [2015] (right). The method of Peer et al. constrains velocities successfully, but cannot remove position drift in a PBF simulation.

Gerszewski et al. [2009] introduced instead an approach to measure deformation that does not require an explicit undeformed configuration in a Lagrangian plastic flow model. Our viscoelasticity model also avoids the explicit definition of undeformed configurations, thanks to a conformation tensor whose time evolution is reduced to the solution of a first-order system. Nevertheless, our formulation reproduces a larger palette of viscoelastic materials.

Another particle-based alternative for the simulation of fluids is the PBF model [Macklin and Müller 2013]. It builds on a SPH discretization of fluid quantities, formulates incompressibility as density constraints, and solves for particle positions directly using the PBD constrained dynamics solver [Müller et al. 2006]. Takahashi et al. [2014] have proposed the treatment of viscosity in the context of PBF. They define particle links similar to those of Clavet et al. [2005] to describe the local material structure, addressing also elasticity and volume conservation [Takahashi et al. 2016]. We have tested the ability of their model to enforce high viscosity, and we have found that it fails to prevent drift without suffering unwanted elastic oscillations, as shown in Fig. 2 and the accompanying video. We have also considered integrating and implicit particle-based viscosity formulation within PBF, in particular the approach of Peer et al. [2015]. However, this combination suffers from excessive drift, as shown in Fig. 3, because constraints are applied on velocities, but not on positions directly.

3 CONSTRAINT-BASED VISCOELASTICITY

In this section, we introduce our constraint-based model of viscoelasticity. We first describe a constitutive model of viscoelasticity in polymeric fluids, which is the stress-based counterpart for our constraint-based formulation. Then, we describe the derivation of implicit conformation constraints acting on fluid velocities, as well as the parameters of our model.

3.1 Constitutive Model of Polymer Conformation

In polymeric fluids, the dissolved polymer endows the fluid with viscoelastic properties. Due to friction between the fluid and the polymer, the elasticity of the polymer is transmitted to the fluid, producing the overall viscoelastic behavior [Bird et al. 1977]. This behavior can be represented using a constitutive model that relates viscoelastic stress to the change in a polymer conformation tensor $\mathbf{Q}$ [Deshpande et al. 2010].

The polymer conformation model defines a reference value $\bar{\mathbf{Q}} = \mathbf{I}$ for the conformation tensor at rest state, and reduces its time evolution to the solution of a first-order system with relaxation time constant $\tau = \frac{b}{k}$, which amounts to the ratio between the viscous friction b between the polymer and the fluid, and the stiffness k of the polymer. The first-order time evolution of the conformation tensor is defined as:

$$\frac{\mathcal{D}\mathbf{Q}}{\mathcal{D}t} = -\frac{1}{\tau}\left(\mathbf{Q} - \bar{\mathbf{Q}}\right). \tag{1}$$

In this equation, the time evolution of the conformation tensor is expressed using the *upper convected derivative* $\frac{\mathcal{D}\mathbf{Q}}{\mathcal{D}t}$, which is a derivative that takes into account local fluid translation and rotation. It is defined as:

$$\frac{\mathcal{D}\mathbf{Q}}{\mathcal{D}t} = \frac{D\mathbf{Q}}{Dt} - \mathbf{Q}\,\nabla\mathbf{u} - (\nabla\mathbf{u})^T\,\mathbf{Q}, \tag{2}$$

where $\frac{D\mathbf{Q}}{Dt}$ is the standard convective derivative and $\mathbf{u}$ is the fluid velocity.

From (1) and (2), on a Lagrangian setting, the conformation tensor rate can be computed as:

$$\frac{D\mathbf{Q}}{Dt} = \mathbf{Q}\,\nabla\mathbf{u} + (\nabla\mathbf{u})^T\,\mathbf{Q} - \frac{1}{\tau}\left(\mathbf{Q} - \bar{\mathbf{Q}}\right). \tag{3}$$

The constitutive model of polymeric fluids is complete with the definition of the viscoelastic stress σ as:

$$\sigma = k\,c\,s\,\left(\mathbf{Q} - \bar{\mathbf{Q}}\right), \tag{4}$$

where s is a scale factor that depends on the geometry and structure of the polymer, c is the polymer concentration, and k is the polymer's stiffness, as mentioned above. This stiffness is typically a function of temperature.

By plugging the viscoelastic stress into the equation of fluid momentum conservation, we would obtain an upper convected Maxwell model for a polymeric fluid [Deshpande et al. 2010]. Instead, we formulate viscoelasticity as a constraint on the conformation tensor, as we show next.

Fig. 4. Viscoelastic cubes are dropped on the ground. By varying two physics-based parameters, the conformation relaxation time constant τ and the compliance α, we achieve materials that bounce elastically (yellow), appear highly viscous (magenta), or splash as an inviscid liquid (cyan).

3.2 Implicit Conformation Constraint

To model high viscoelasticity, we propose a constraint that preserves the rest-state value of the conformation tensor, i.e.,

$$\mathbf{C}(\mathbf{Q}) = \mathbf{Q} - \bar{\mathbf{Q}} = \mathbf{Q} - \mathbf{I} = \mathbf{0}. \tag{5}$$

On each simulation step, we wish to enforce this constraint implicitly, i.e., on the simulation state at the end of the time step. We do this by combining the constraint equation (5) with implicit integration of the conformation tensor rate (3). We denote with a superscript 0 variables at the beginning of the time step, e.g., $\mathbf{Q}^0$ is the conformation tensor at the beginning of the time step. Then, the conformation tensor at the end of the time step can be obtained through implicit Euler integration of (3), by solving:

$$\frac{\mathbf{Q} - \mathbf{Q}^0}{\Delta t} = \mathbf{Q}\,\nabla\mathbf{u} + (\nabla\mathbf{u})^T\,\mathbf{Q} - \frac{1}{\tau}\,\left(\mathbf{Q} - \bar{\mathbf{Q}}\right). \tag{6}$$

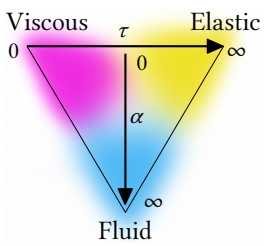
Plugging the tensor constraint (5) in this equation, we obtain an implicit conformation constraint on fluid velocities:

$$\mathbf{C}(\mathbf{u}) = \mathbf{Q}^0 - \bar{\mathbf{Q}} + \Delta t\,\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right) = \mathbf{0}. \tag{7}$$

Our simulation loop for viscoelastic fluids proceeds on each time step by computing a dynamic update subject to the implicit velocity-based constraint (7). For this purpose, we use the constrained dynamics solver proposed in Section 4. At the end of each time step, we update the conformation tensor by solving for $\mathbf{Q}$ in (6). In the particular case when $\tau \to 0$, we simply set the tensor to be $\mathbf{Q} = \mathbf{I}$.

The relaxation time τ affects the viscoelasticity behavior of the fluid. As $\tau \to 0$, the internal elastic forces of the polymer dominate over friction forces with the fluid ($k \gg b$), and the polymer recovers quickly its structure. In the viscoelastic fluid simulation, the conformation tensor $\mathbf{Q}$ barely changes over time, and the conformation constraint (7) becomes effectively a null-strain-rate constraint. The fluid appears highly viscous. Conversely, as $\tau \to \infty$, the friction forces of the polymer with the fluid dominate over its internal elastic forces ($b \gg k$), and the polymer fails to recover its structure. In the viscoelastic fluid simulation, the conformation tensor $\mathbf{Q}$ varies over time as a result of the strain rate according to (3), and the conformation constraint (7) acts on fluid velocities to remove the existing conformation change. The fluid appears elastic.

In the constitutive model of polymer conformation, the viscoelastic stress depends also on a compliance $\alpha = \frac{1}{k\,c\,s}$ according to (4). Later in Section 5, we show how to incorporate this compliance into our constrained dynamics solver as a relaxation coefficient for the conformation constraint (7). The compliance α defines the fluidity of the model. With two physically based parameters, τ and α, we obtain a palette of materials that spans elastoplastic, highly viscous, and inviscid fluids, as shown in the inset. The influence of



ALGORITHM 1: PBD step

Input: Initial state $(\mathbf{x}^0, \mathbf{u}^0)$.
Output: Updated state $(\mathbf{x}, \mathbf{u})$.

Compute constraint-free state
$\mathbf{u}^* \leftarrow \mathbf{u}^0 + \Delta t\, \mathbf{M}^{-1}\, \mathbf{f}(\mathbf{x}^0)$
$\mathbf{x}^* \leftarrow \mathbf{x}^0 + \Delta t\, \mathbf{u}^*$

Project positions
$\mathbf{x} \leftarrow$ project $\mathbf{x}^*$ to $\mathbf{C}^{\mathbf{x}}(\mathbf{x}) = \mathbf{0}$
$\mathbf{u} \leftarrow \frac{\mathbf{x}-\mathbf{x}^0}{\Delta t}$

material parameters on the fluid's behavior is evidenced even on simple scenarios, such as impact (Fig. 4) or liquid rope coiling (Fig. 5). In these examples, material colors are chosen by interpolating the colors in the inset according to the parameter settings. Even on moderately complex scenes (more than 10 000 particles), the diverse materials can be simulated interactively.

4 CONSTRAINED DYNAMICS SOLVER

PBD can be regarded as an integration scheme for constrained dynamics. In this section, we propose a doubly constrained PBD (DC-PDB) solver that handles also velocity-based constraints, such as those in our viscoelasticity model. DC-PBD projects both velocities and positions to the velocity-based constraints, and in this way it improves convergence and stability. We start the section with a summary of the regular PBD solver, and then we describe the differences in our DC-PDB approach.

4.1 PBD Solver

Given a dynamic system with mass $\mathbf{M}$, external forces $\mathbf{f}$, and position-based constraints $\mathbf{C}^{\mathbf{x}}$, the PBD method executes each simulation step as follows. Starting from positions and velocities $(\mathbf{x}^0, \mathbf{u}^0)$, it first computes a constraint-free state $(\mathbf{x}^*, \mathbf{u}^*)$ using symplectic Euler integration. Then, it projects the positions to the constraints, and computes velocities through finite differences between final and initial positions, to obtain the state $(\mathbf{x}, \mathbf{u})$ at the end of the time step. The PBD solver is summarized in Algorithm 1.

In PBD, the constraint projection is typically solved using Fast-Projection Jacobi or Gauss-Seidel iterations. Constraints are linearized after each iteration, and Fast Projection implies that, within each iteration, the projection is computed by minimizing the distance to the result from the previous iteration, not to the initial value [Goldenthal et al. 2007; Hairer et al. 2002].

PBD succeeds to robustly and efficiently model stiff potentials as position-based constraints. Moreover, the recent XPBD extension adds relaxation to the constraint projection in order to model constraint compliance. However, PBD is not naturally designed to handle efficiently velocity-based constraints, such as the viscoelasticity constraint (7). In addition, constraint nonlinearity, such as the one introduced by the SPH kernels used in PBF, may complicate the convergence of Jacobi or Gauss-Seidel solvers.

ALGORITHM 2: DC-PBD step

Input: Initial state $(\mathbf{x}^0, \mathbf{u}^0)$.
Output: Updated state $(\mathbf{x}, \mathbf{u})$.

Compute constraint-free state
$\mathbf{u}^* \leftarrow \mathbf{u}^0 + \Delta t\, \mathbf{M}^{-1}\, \mathbf{f}(\mathbf{x}^0)$
$\mathbf{x}^* \leftarrow \mathbf{x}^0 + \Delta t\, \mathbf{u}^*$

Project velocities
$\mathbf{u}^{**} \leftarrow$ project $\mathbf{u}^*$ to $\mathbf{C}^{\mathbf{u}}(\mathbf{u}^{**}) = \mathbf{0}$
$\mathbf{x}^{**} \leftarrow \mathbf{x}^* + \Delta t\ (\mathbf{u}^{**} - \mathbf{u}^*)$

Project positions
$\mathbf{x} \leftarrow$ project $\mathbf{x}^{**}$ to $\mathbf{C}^{\mathbf{x}}(\mathbf{x}) = \mathbf{0}$ and $\tilde{\mathbf{C}}^{\mathbf{u}}(\mathbf{x}) \equiv \mathbf{C}^{\mathbf{u}}\left(\frac{\mathbf{x}-\mathbf{x}^0}{\Delta t}\right) = \mathbf{0}$
$\mathbf{u} \leftarrow \frac{\mathbf{x}-\mathbf{x}^0}{\Delta t}$

4.2 Doubly Constrained PBD

Given the generic dynamic system described above, we propose a constrained dynamics algorithm that handles both position-based constraints $\mathbf{C}^{\mathbf{x}}$ and velocity-based constraints $\mathbf{C}^{\mathbf{u}}$ efficiently. The key difference is to project both positions and velocities to the constraints. A similar idea is applied by the RATTLE algorithm for molecular dynamics [Andersen 1983], which is a constrained-dynamics version of the velocity-Verlet integrator. But, unlike RATTLE, we exploit the robustness of PBD by prioritizing the projection

Fig. 5. We compare liquid rope coiling with different material parameters. In the top row, varying the compliance α, with relaxation time constant $\tau = 0.15$ in all cases. In the bottom row, varying τ, with $\alpha = 0$ in all cases. Please see the elastic effects in the accompanying video. The examples are colored by interpolating the material color palette shown in Section 3.2.

of positions, and computing velocity estimates through finite differences of safe positions.

We start the DC-PBD solver by computing a constraint-free state $(\mathbf{x}^*, \mathbf{u}^*)$ using symplectic Euler integration, same as in regular PBD. Then, we project the velocities to the velocity-based constraints $\mathbf{C^u}$, and we update particle positions with the resulting velocity correction. Altogether, we obtain a velocity-safe state $(\mathbf{x}^{**}, \mathbf{u}^{**})$.

To conclude, we compute the final, position-safe state $(\mathbf{x}, \mathbf{u})$, by projecting the positions to both the position-based and velocity-based constraints. To do this, we need to transform the velocity-based constraints $\mathbf{C^u}$ into position-based constraints, which are added to the regular position-based constraints $\mathbf{C^x}$. We define the velocities through finite differences between the initial and final positions, i.e., $\mathbf{u} = \frac{\mathbf{x}-\mathbf{x}^0}{\Delta t}$, and thus we can turn the velocity-based constraints into position-based constraints of the form $\tilde{\mathbf{C}}^{\mathbf{u}}(\mathbf{x}) \equiv \mathbf{C^u}\left(\frac{\mathbf{x}-\mathbf{x}^0}{\Delta t}\right) = \mathbf{0}$.

The DC-PBD solver is summarized in Algorithm 2. We compute the position projection using Fast Projection, just like in regular PBD. The velocity projection, on the other hand, is a positive semi-definite linear problem, similar in structure to the one tackled by Peer et al. [2015]. Same as they do for generic cases, we solve it using Jacobi iterations. However, unlike the position projection, where Jacobians are recomputed after each Jacobi iteration, in the velocity projection the Jacobians are constant and can be computed just once per time step. We provide full details of the application of the DC-PBD solver to viscoelasticity constraints in Section 5.

To evaluate our DC-PBD solver, we have run a test where we drop on the ground a cube with full viscosity constraints and no compliance (Fig. 4, with $\tau = 0$ and $\alpha = 0$). We have computed the RMS error of particles in the cube w.r.t. their undeformed positions, as a global measure of constraint drift. We have compared constraint drift with our DC-PBD solver vs. regular PBD (applied also to velocity-based constraints, discretized using finite differences). Position projection alone requires a time step smaller than 30 ms to ensure stability. With our DC-PBD solver, on the other hand, the simulation remains stable with time steps twice as large, i.e., 60 ms. We have also found that the number of Jacobi iterations affects the amount of constraint drift (with lower drift in DC-PBD under the same total iteration count, as shown in the plots in Fig. 6), but it has little effect on stability.

Our conclusions about the reasons for the improved stability and robustness of DC-PBD are the following. Projection of particle positions is nonlinear, and nonlinear Jacobi may have trouble converging under large time steps. Projection of velocities, on the other hand, is linear, and linear Jacobi turns out more robust. The position correction in the "Project velocities" step in Algorithm 2 removes much of the position deviation, and further steps of nonlinear position projection are less prone to robustness problems induced by nonlinearity.

5 VISCOELASTIC POSITION-BASED FLUIDS

We integrate conformation constraints in a PBF model to simulate viscoelastic incompressible fluids. However, unlike the original PBF model, we employ our DC-PBD solver for improved convergence, and we adopt XPBD to support constraint compliance. We start this section by outlining the complete simulation model, and we then describe in detail how to handle conformation constraints for velocity and position projection respectively.

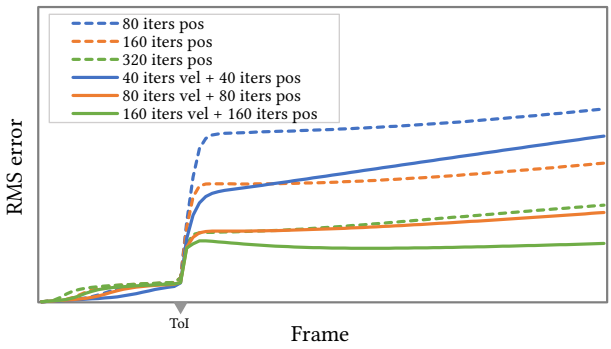


Fig. 6. Comparison of constraint drift with our DC-PBD solver (with velocity and position projection) vs. position projection alone, with a time step of 30 ms and various iteration counts. We drop a fully viscous cube (Fig. 4), which impacts the ground at time ToI, and we measure the RMS error of particle positions w.r.t. an undeformed cube as the simulation evolves. Position projection alone suffers higher error under the same total iteration count, and it requires a smaller time step to be stable (30 ms, vs. 60 ms in the case of DC-PBD).

5.1 PBF Model

PBF [Macklin and Müller 2013] simulates fluids using a SPH discretization [Monaghan 1992], with incompressibility as a density constraint on particle positions. Following the SPH discretization, given a set of particles, each one with mass m_j, attribute value a_j, and position $\mathbf{x}_j$, the value of the attribute a at an arbitrary position $\mathbf{x}_i$ is evaluated as:

$$a(\mathbf{x}_i) = \sum_j \frac{m_j}{\rho_j} a_j W_{ij}, \tag{8}$$

with $W_{ij} = W(\mathbf{x}_{ij}) = W(\mathbf{x}_i - \mathbf{x}_j)$ being the evaluation at $\mathbf{x}_i$ of a smoothing kernel with support radius h and centered at $\mathbf{x}_i$, and ρ_j the density field evaluated at $\mathbf{x}_j$. We employ the SPH-based attribute evaluation for the computation of the fluid velocity in conformation constraints (7).

Combining incompressibility and viscoelasticity, the DC-PBD solver proceeds as follows. For the position projection step of DC-PBD, we enforce both density and conformation constraints on particle positions. In Section 5.3, we describe how we formulate position-based conformation constraints per simulation particle. We have experimented with various strategies to combine incompressibility and viscoelasticity in the position projection, and we have observed best convergence by staggering one Jacobi iteration of incompressibility over all particles with one Jacobi iteration of viscoelasticity over all particles.

For the velocity projection step of DC-PBD, we enforce only conformation constraints on particle velocities, as we describe next in Section 5.2. Once this is done, we update the conformation tensor $\mathbf{Q}$ on each particle by solving the linear system (6).

Both for the position and velocity projection steps, we adopt the XPBD method [Macklin et al. 2016], which modifies the original PBD iterations to support constraint compliance, as also done in other constrained dynamics methods [Tournier et al. 2015]. In our viscoelasticity model, constraint compliance allows us to account for the compliance α of viscoelastic stress defined in Section 3.2.

5.2 Discrete Velocity-Based Constraints

To implement the velocity-based, implicit conformation constraint (7) within the PBF framework, we express the constraint on each simulation particle. To do this, we compute the fluid velocity $\mathbf{u}_i$ at the position of the i^{th} particle using the SPH formulation (8) and, accordingly, we evaluate the SPH velocity gradient:

$$\nabla \mathbf{u}_i = \sum_j \frac{m_j}{\rho_j} \mathbf{u}_{ji} \nabla W_{ij}^T, \tag{9}$$

with $\mathbf{u}_{ji} = \mathbf{u}_j - \mathbf{u}_i$.

Since the conformation tensor $\mathbf{Q}$ is symmetric, we rearrange it as a six-dimensional vector $\mathbf{q}$:

$$\mathbf{q} = (Q_{xx}, Q_{yy}, Q_{zz}, Q_{xy}, Q_{xz}, Q_{yz})^T; \qquad \bar{\mathbf{q}} = (1, 1, 1, 0, 0, 0)^T.$$

Rewriting the conformation constraint (7) using the vector notation, and plugging in the expression of the velocity gradient (9), we obtain the discrete version of the velocity-based constraint:

$$\mathbf{C}_i(\mathbf{u}) = \mathbf{q}_i^0 - \bar{\mathbf{q}} + \Delta t \sum_j \frac{m_j}{\rho_j} \mathbf{A}_{ij} \mathbf{u}_{ji} = \mathbf{0}, \tag{10}$$

$$\text{with } \mathbf{A}_{ij} = \begin{pmatrix} 2\partial_x W_{ij} & 0 & 0 \\ 0 & 2\partial_y W_{ij} & 0 \\ 0 & 0 & 2\partial_z W_{ij} \\ \partial_y W_{ij} & \partial_x W_{ij} & 0 \\ \partial_z W_{ij} & 0 & \partial_x W_{ij} \\ 0 & \partial_z W_{ij} & \partial_y W_{ij} \end{pmatrix}.$$

We apply this constraint to each simulation particle in the velocity-projection step of our DC-PBD solver (see Section 4.2). Each Jacobi iteration with XPBD yields the following update of Lagrange multipliers and particle velocities, respectively:

$$\Delta\lambda_i = \left(\text{diag}\left(\mathbf{J}_i \mathbf{M}^{-1} \mathbf{J}_i^T\right) + \frac{\alpha}{\Delta t} \mathbf{I}\right)^{-1} \left(-\mathbf{C}_i(\mathbf{u}) - \frac{\alpha}{\Delta t} \lambda_i\right), \tag{11}$$

$$\Delta \mathbf{u}_i = \frac{\beta}{n_i} \sum_j \mathbf{J}_{ji}^T \Delta\lambda_j. \tag{12}$$

We relax the Jacobi update with a factor $\frac{\beta}{n_i}$ to ensure convergence, where n_i is the size of the particle neighborhood and β is a scaling coefficient to avoid excessive relaxation ($\beta = 5$ in our examples).

In our implementation, we approximate the constraint Jacobians as:

$$\mathbf{J}_{ik} = \frac{\partial \mathbf{C}_i(\mathbf{u})}{\partial \mathbf{u}_k} = \begin{cases} \Delta t \frac{m_k}{\rho_k} \mathbf{A}_{ik} & \text{if } i \neq k \\ -\Delta t \sum_j \frac{m_j}{\rho_j} \mathbf{A}_{ij} & \text{if } i = k. \end{cases} \tag{13}$$

Note that these Jacobians remain constant during the whole velocity projection.

The constraint response update (11) includes the compliance α defined in Section 3.2. In the original XPBD formulation [Macklin et al. 2016], the compliance is applied to constraints formulated on positions, and hence it is scaled by a factor $\frac{1}{\Delta t^2}$. In our setting, with constraints formulated on velocities, it is scaled by $\frac{1}{\Delta t}$ instead. Fig. 5 and Fig. 9 show the effect of varying the compliance on two different examples.

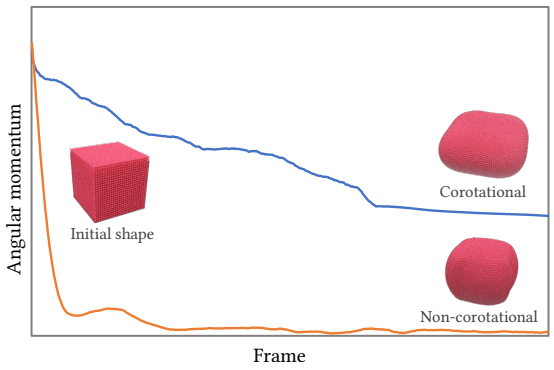


Fig. 7. Plot of angular momentum of a rotating block, with and without our corotational formulation. With a non-corotational velocity gradient, rigid body motion is soon damped. With our approach, residual damping is due only to approximation errors.

5.3 Discrete Position-Based Constraints

For the position-projection step of our DC-PBD solver, we wish to rewrite the discrete velocity-based conformation constraint (10) as a function of particle positions, i.e., in the form $\tilde{\mathbf{C}}_i(\mathbf{x}) = 0$. In principle, we could do this by approximating particle velocities through finite differences of particle positions, $\mathbf{u}_j = \frac{\mathbf{x}_j - \mathbf{x}_j^0}{\Delta t}$. However, this approach would fail to preserve angular momentum and would damp rotational motion of the fluid. At the core of the problem lies the inability of SPH to correctly reconstruct linear fields (e.g., uniform angular velocity). A similar observation was made by Becker et al. [2009] for the computation of deformation gradients and, similar to their corotational deformation gradient, we derive a corotational formulation of the velocity gradient (9) from particle positions. Fig. 7 shows an example simulation of a rotating block with and without our corotational discretization. The difference in rotational damping is evident.

Following Becker et al., we estimate for each particle the best-fit rotation $\mathbf{R}_i$ to the deformations of its neighbor particles:

$$\mathbf{R}_i = \arg\min \sum_j \left| \mathbf{R}_i \left(\mathbf{x}_j^0 - \mathbf{c}_i^0\right) - \left(\mathbf{x}_j - \mathbf{c}_i\right) \right|^2, \tag{14}$$

where $\mathbf{c}_i$ is the center of mass of the neighbor particles.

Then, for the estimation of the velocity gradient of the i^{th} particle, we define corotational finite-difference velocities for all particles in its neighborhood, by compensating for the rotation $\mathbf{R}_i$. Specifically:

$$\mathbf{u}_j = \frac{\mathbf{x}_j - \mathbf{x}_j^r}{\Delta t}, \qquad \text{with } \mathbf{x}_j^r = \mathbf{c}_i + \mathbf{R}_i \left(\mathbf{x}_j^0 - \mathbf{c}_i^0\right). \tag{15}$$

By substituting this velocity computation in the velocity gradient (9), we rewrite (10) to obtain the position-based expression of the

Table 1. Parameter values and performance statistics for all our benchmarks (all rendered at 30 fps). The table lists: the total number of particles, the time step Δt, the amount of steps per frame, the number of iterations of the viscoelasticity constraint projection per step, the total time per frame (in seconds), and τ and α values. Some materials: † thick cream; †† runny cream; ‡ strawberry syrup.

Scene & Fig	Particles	Time step	Steps/frame	Iters	Time/frame	τ	α
Blocks (Fig. 4)	10k	1/240	8	20	0.16s	See Fig.	
Coiling (Fig. 5)	89k	1/240	8	40	1.08s	See Fig.	
Interactive (Figs. 8, 10)	15k	1/200	3	8	0.03s	0.1	$0 \leq \alpha \leq 1$
Waffle (Fig. 9)	80k	1/300	10	40	1.42s	0.1	$0^{\dagger}$, $0.01^{\dagger\dagger}$
Honey (Fig. 12)	105k	1/300	10	10	0.15s	0	0.005
Cake (Fig. 1)	150k	1/750	25	15	1.13s	$0.1^{\dagger}$, $0.5^{\ddagger}$	$0^{\dagger}$, $0.015^{\ddagger}$
Armadillos (Fig. 11)	12M	1/150	5	15	19s	0.5	0.01

discrete conformation constraint:

$$\tilde{\mathbf{C}}_i(\mathbf{x}) = \mathbf{q}_i^0 - \bar{\mathbf{q}} + \sum_j \frac{m_j}{\rho_j} \mathbf{A}_{ij} \left(\mathbf{x}_{ji} - \mathbf{x}_{ji}^r \right) = \mathbf{0}, \tag{16}$$

We apply this constraint to each simulation particle in the position-projection step of our DC-PBD solver (see Section 4.2). Each Jacobi iteration with XPBD yields the following update of Lagrange multipliers and particle positions, respectively:

$$\Delta\lambda_i = \left(\text{diag}\left(\tilde{\mathbf{J}}_i \, \mathbf{M}^{-1} \, \tilde{\mathbf{J}}_i^T \right) + \frac{\alpha}{\Delta t^2} \mathbf{I} \right)^{-1} \left(-\tilde{\mathbf{C}}_i(\mathbf{x}) - \frac{\alpha}{\Delta t^2} \lambda_i \right), \tag{17}$$

$$\Delta\mathbf{x}_i = \frac{\beta}{n_i} \sum_j \tilde{\mathbf{J}}_{ji}^T \, \Delta\lambda_j. \tag{18}$$

The Jacobians are defined as $\tilde{\mathbf{J}}_i = \frac{1}{\Delta t} \mathbf{J}_i$, and they need to be recomputed after each Jacobi update of particle positions.

6 RESULTS

We have tested our viscoelasticity model on multiple benchmarks. They were all executed on a Hexa-core Intel i7-3930K CPU with 32 GB of RAM, and a NVIDIA GeForce 1070 GTX GPU with 1920 CUDA Cores. The PBF model with the DC-PBD solver is programmed entirely on the GPU. Table 1 shows the major performance statistics and parameter settings for all the benchmarks (all rendered at 30 fps). Next, we discuss the results. Please watch the accompanying video.

Interactive scenes (Figs. 8 and 10). These scenes demonstrate the suitability of the DC-PBD solver in interactive applications where very small time steps cannot be used. The improved convergence and stability enables even interactive performance on moderately complex scenarios. Both the hand-and-bowl scene and the ice cream scene consist of 15k particles each. In these examples, we use a Leap Motion™ device to track hand motions and move a virtual hand or other objects. The tests also show interactive modification of the material parameters, e.g., increasing the compliance α to model ice cream melting.

Large-scale simulations (Figs. 9, 11 and 12). These scenes represent several computationally demanding benchmarks with complex behaviors. The simulation of whipped cream poured onto a waffle (Fig. 9) runs up to 90k particles with a computation time of 1.42 seconds per frame. This simulation requires high particle density to correctly resolve the ridges on the cream's surface. We compare two scenarios, with the same value of relaxation time τ, but with different compliance α. With increased viscoelasticity (i.e., lower α), the cream coils in a regular manner.

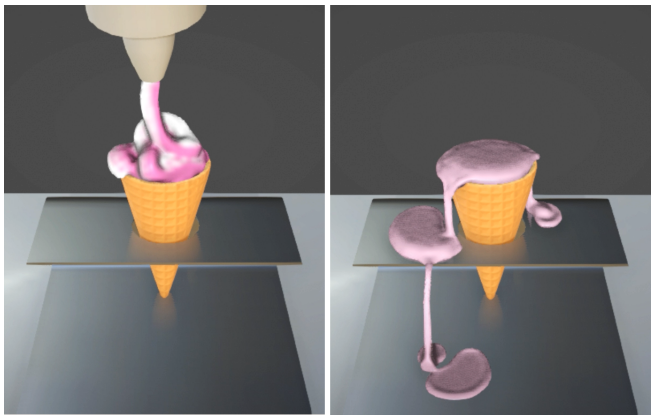


Fig. 8. Screen captures of interactive ice cream simulation. The dispenser is controlled interactively through a Leap Motion™ device, and ice cream is poured into the cone. Increasing the compliance α, the ice cream melts. The scene consists of up to 15k particles, simulated at 30 ms/frame.

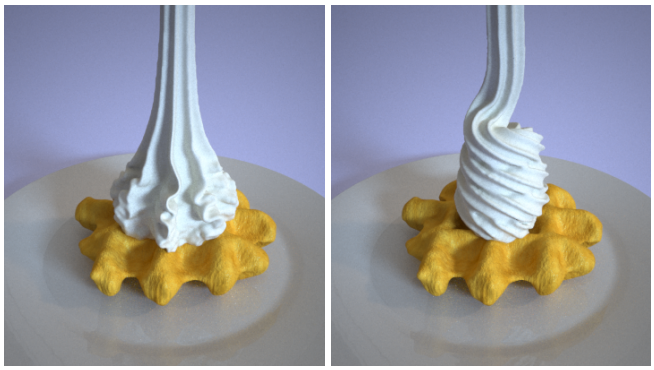


Fig. 9. Two types of whipped cream are poured onto a waffle, with compliance ($\alpha = 0.01$) on the left, and without compliance on the right. High viscosity in the right causes a regular coiling effect.

Fig. 10. Three screen captures of interactive manipulation of a viscoelastic fluid, consisting of up to 15k particles, and simulated at 30 ms/frame. The motion of the hands is tracked using a Leap Motion™ device, and this motion is applied to a virtual hand and a bowl, which interact with the coiling fluid. We also demonstrate interactive changes to material parameters.

The massive test shown in Fig. 11 involves the computation of high viscoelasticity on a large-scale scene, which replicates a benchmark tested by Peer et al. [2015]. They simulated 11 million particles at 144 seconds per frame and 50 fps. We simulate 12 million particles at 19 seconds per frame and 30 fps. Prorating particle count and frame-rate, our method achieves a speed-up of more than 13x, showing that our viscoelasticity model provides superior performance to previous approaches even on large-scale scenes. Note that the iterative Jacobi or Gauss-Seidel solvers of PBD-type methods become particularly slow at such high resolutions, but our method achieves competitive performance. Performance would suffer more, both with our method and with the one by Peer et al., on taller hydrostatic columns.

The pouring honey in Fig. 12 is simulated using 105k particles of viscous fluid, rendered with translucent material. This test runs at 0.14 seconds per frame on average, and exhibits the characteristic buckling of highly viscous materials.

Multiphysics simulation (Fig. 1). One of the features of the PBD-type constrained dynamics solvers is that they can easily accommodate objects and materials with diverse properties. Our model is integrated in the RealFlow commercial software, thereby enabling seamless interaction with other types of PBD-based materials. In this scene, we simulate two types of viscoelastic materials (whipped cream and strawberry syrup) using our conformation constraints, rigid chocolate letters using shape-matching constraints, and soft flowers using distance constraints. The complete scene consists of up to 150k particles and is simulated in 1.13 seconds per frame.

Discussion. In our examples, we have demonstrated that the proposed viscoelasticity model covers efficiently a broad set of simulation scenarios, from interactive scenes to large-scale scenes. To the best of our knowledge, we have shown unprecedented viscous and viscoelastic interactive simulations, with a combination of high viscosity and scene complexity (i.e., particle count) not possible before. Our solution also outperforms previous methods on large-scale scenes, even though the type of constraint solver may not be a priori best suited for such scenes. A key feature for the performance of our solution is the efficiency and robustness of the constraint solver, which allows time steps of moderate size, few iterations per time step, and massive parallelization within each iteration.

The high performance of our solution is a combined result of the constraint-based formulation, the constrained dynamics solver, and the GPU-based parallelization. The impact of our DC-PBD constrained dynamics solver in contrast to regular PBD is discussed in Section 4.2. To evaluate the impact of the constraint-based formulation, we have compared its performance to a stress-based formulation of the polymeric fluid model described in Section 3.1, discretized

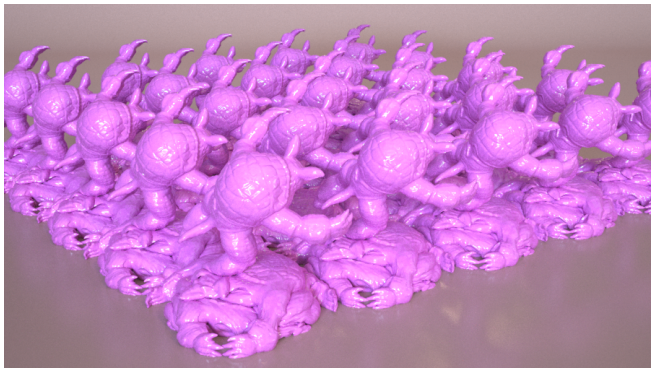


Fig. 11. Massive viscoelastic simulation, with 12 million particles simulated at 19 seconds per frame. This example demonstrates that our viscoelasticity model achieves higher performance than previous methods (an approximate speedup of 13x over [Peer et al. 2015]), even on large-scale scenes.

Fig. 12. Simulation of honey using 105k particles of viscous fluid. This test runs at 0.14 seconds per frame on average, and exhibits the characteristic buckling of highly viscous materials.

using SPH. We encountered impeding robustness problems to implement highly viscous materials (i.e., very low τ) with the stress-based implementation. On a falling cube of $32 \times 32 \times 32$ particles (Fig. 4), the stress-based model is stable for a value of $\tau = 0.015$ and a time step of 0.1 ms. With our constraint-based model, on the other hand, the simulation is stable and of similar accuracy with a time step of 6 ms and an overall speed-up of 7x. To evaluate the impact of the GPU-based parallelization, we have executed the same falling-cube benchmark on an optimized CPU implementation (with Intel TBB-based parallelization). The GPU-based implementation is 12x faster.

Some viscosity models advocate defining viscous stress as a function of shear rate [Peer et al. 2015; Zhu et al. 2015]. In the purely viscous case, our model converges instead to a null-strain-rate constraint, as discussed in Section 3.2. The difference w.r.t. a null-shear-rate constraint produces a hydrostatic stress, which acts against density changes. We have tested using shear rate constraints in our model, and we have validated that the hydrostatic stress acts as a damping term on density, and it helps the convergence of the incompressibility constraint. For compressible liquids, we could perhaps add higher compliance to the hydrostatic part of the conformation constraint, but we have not explored this avenue.

7 LIMITATIONS AND FUTURE WORK

In this paper, we have presented a novel model of viscoelasticity for fluid simulation. Our model formulates viscoelasticity using constraints, and can be solved efficiently within the PBD constrained dynamics framework. We have designed viscoelasticity constraints inspired by a constitutive model of viscoelasticity for polymeric fluids, which employs a conformation tensor with simple treatment of purely viscous vs. elastic effects. To enable efficient and robust simulation, we formulate the constraints implicitly, and we describe their integration in the state-of-the-art XPBD solver, with further improvements.

Our DC-PBD solver might be applicable to other types of constraints, beyond those handled in our work. One such example is friction, which shares a dissipative nature with viscosity, but incorporates constraints on the deviatoric stress. Another example is incompressibility. Similar to the divergence-free SPH method by Bender and Koschier [2017], incompressibility constraints could be applied on both positions and velocities within our DC-PBD solver.

Despite the rich effects achieved with our method and the range of materials that can be simulated, there are still some limitations that suggest interesting future work. Our work inherits some of the generic limitations of the PBD and PBF approach, in particular the convergence limitations of Jacobi or Gauss-Seidel solvers. It would be interesting to take advantage of the connection between PBD and minimization formulations of implicit integration, to explore efficient optimization algorithms, as done by others after the projective dynamics method [Bouaziz et al. 2014]. However, those optimization algorithms cannot be trivially extended to fluids as they make strong connectivity assumptions [Weiler et al. 2016].

Our method cannot handle large elastic deformations accurately, which would require storing some explicit measure of rest state. Additionally, while some of our examples demonstrate the simulation of fine features, the ability to resolve such fine features is eventually limited by the particle resolution of the simulation. Fusing codimensional representations [Zhu et al. 2015] with constraint-based viscoelasticity would enable even richer effects under manageable computational cost.

To conclude, even though our method is parameterized using two physics-based parameters, it is difficult to design them purely from measurable physical quantities in a discretization-independent manner. Our model is derived from a constitutive model for polymeric fluids, and the parameters could be set from geometric and physical quantities only for such fluids. However, we apply the model to other types of viscoelastic fluids too, and in that case the model can be regarded as empirical or phenomenological, and parameters could be estimated from measurements. In our examples, we have opted for an artist-driven parameter design, which nevertheless proves effective thanks to the narrow set of parameters. Another problem of the parameterization is that many details of the constitutive model are reduced to just one parameter, the constraint compliance. This limits the ability to represent non-Newtonian fluids with complex dependence on any of these material parameters, e.g., some pseudoplastic fluids.

ACKNOWLEDGEMENTS

The authors wish to thank Rosa Sánchez for her help with demo production and the video, Daniel Lobo for assistance with the Leap Motion, Alex Ribao for assistance with rendering in RealFlow, and Dr. Juan Carlos Fernández for discussions on the polymeric fluid model. We also thank the anonymous reviewers for their feedback, and other members of the MSLab at URJC and the RealFlow team at Next Limit. This work was supported in part by the Spanish

Ministry of Economy, through grant TIN2015-70799-R and Iván Alduán's Torres Quevedo PTQ-15-07401 contract.

REFERENCES

Iván Alduán and Miguel A. Otaduy. 2011. SPH Granular Flow with Friction and Cohesion. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*. 25–32.

Iván Alduán, Angel Tena, and Miguel A. Otaduy. 2016. DYVERSO: A Versatile Multi-Phase Position-Based Fluids Solution for VFX. *Comput. Graphics Forum* (2016), in press.

Hans C Andersen. 1983. Rattle: A "velocity" version of the shake algorithm for molecular dynamics calculations. *J. Comput. Phys.* 52, 1 (1983), 24 – 34.

Luiz Fernando de Souza Andrade, Marcos Sandim, Fabiano Petronetto, Paulo Pagliosa, and Afonso Paiva. 2015. Particle-based Fluids for Viscous Jet Buckling. *Comput. Graph.* 52, C (2015), 106–115.

David Baraff. 1989. Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies. In *Annual Conference on Computer Graphics and Interactive Techniques*. 223–232.

David Baraff. 1996. Linear-time Dynamics Using Lagrange Multipliers. In *Annual Conference on Computer Graphics and Interactive Techniques*. 137–146.

Adam W. Bargteil, Chris Wojtan, Jessica K. Hodgins, and Greg Turk. 2007. A Finite Element Method for Animating Large Viscoplastic Flow. *ACM Trans. Graph.* 26, 3 (2007).

Christopher Batty and Robert Bridson. 2008. Accurate Viscous Free Surfaces for Buckling, Coiling, and Rotating Liquids. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*. 219–228.

Christopher Batty, Andres Uribe, Basile Audoly, and Eitan Grinspun. 2012. Discrete Viscous Sheets. *ACM Trans. Graph.* 31, 4 (2012), 113:1–113:7.

Markus Becker, Markus Ihmsen, and Matthias Teschner. 2009. Corotated SPH for Deformable Solids. In *Eurographics Conference on Natural Phenomena*. 27–34.

Jan Bender and Dan Koschier. 2017. Divergence-Free SPH for Incompressible and Viscous Fluids. *IEEE Trans. Visual Comput. Graphics* 23, 3 (2017), 1193–1206.

Jan Bender, Matthias Müller, Miguel A. Otaduy, Matthias Teschner, and Miles Macklin. 2014. A Survey on Position-Based Simulation Methods in Computer Graphics. *Comput. Graphics Forum* 33, 6 (2014), 228–251.

Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2010. Discrete Viscous Threads. *ACM Trans. Graph.* 29, 4 (2010).

Robert Byron Bird, Robert Calvin Armstrong, Ole Hassager, and Charles F Curtiss. 1977. *Dynamics of polymeric liquids*. Vol. 1. Wiley New York.

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 (2014), 154:1–154:11.

Mark Carlson, Peter J. Mucha, R. Brooks Van Horn III, and Greg Turk. 2002. Melting and Flowing. In *SIGGRAPH'02*.

Yuanzhang Chang, Kai Bao, Youquan Liu, Jian Zhu, and Enhua Wu. 2009. A particle-based method for viscoelastic fluids animation. In *ACM sympos. Virtual Reality Software and Technology*.

Simon Clavet, Philippe Beaudoin, and Pierre Poulin. 2005. Particle-based viscoelastic fluid simulation. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*.

Gilles Daviet and Florence Bertails-Descoubes. 2016. A Semi-implicit Material Point Method for the Continuum Simulation of Granular Materials. *ACM Trans. Graph.* 35, 4 (2016), 102:1–102:13.

Abhijit P. Deshpande, J. Murali Krishnan, and P. B. Sunil Kumar (Eds.). 2010. *Rheology of complex fluids*. Springer New York.

Nick Foster and Dimitri Metaxas. 1996. Realistic Animation of Liquids. *Graph. Models Image Process.* 58, 5 (1996), 471–483.

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

Tolga G. Goktekin, Adam W. Bargteil, and James F. O'Brien. 2004. A Method for Animating Viscoelastic Fluids. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 463–468.

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. 2007. Efficient Simulation of Inextensible Cloth. *ACM Trans. Graph.* 26, 3 (2007).

Ernst Hairer, Christian Lubich, and Gerhard Wanner. 2002. *Geometric Numerical Integration*. Vol. 31. Springer-Verlag.

Xiaowei He, Ning Liu, Sheng Li, Hongan Wang, and Guoping Wang. 2012. Local Poisson SPH For Viscous Incompressible Fluids. *Comput. Graphics Forum* 31, 6 (2012), 1948–1958.

Geoffrey Irving, Craig Schroeder, and Ronald Fedkiw. 2007. Volume Conserving Finite Element Simulations of Deformable Models. *ACM Trans. Graph.* 26, 3 (2007).

Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. 2008. Staggered Projections for Frictional Contact in Multibody Systems. *ACM Trans. Graph.* 27, 5 (2008), 164:1–164:11.

Gergely Klár, Theodore Gast, Andre Pradhana, Chuyuan Fu, Craig Schroeder, Chenfanfu Jiang, and Joseph Teran. 2016. Drucker-prager Elastoplasticity for Sand Animation. *ACM Trans. Graph.* 35, 4 (2016), 103:1–103:12.

Egor Larionov, Christopher Batty, and Robert Bridson. 2017. Variational Stokes: A Unified Pressure-Viscosity Solver for Accurate Viscous Liquids. *ACM Trans. Graph.* 36, 4 (2017), 101:1–101:11.

Toon Lenaerts and Philip Dutré. 2009. Mixing fluids and granular materials. In *Eurographics*, Vol. 28. 213–218.

Miles Macklin and Matthias Müller. 2013. Position Based Fluids. *ACM Trans. Graph.* 32, 4 (2013), 104:1–104:12.

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: Position-based Simulation of Compliant Constrained Dynamics. In *Int. Conf. on Motion in Games*. 49–54.

Joseph J. Monaghan. 1992. Smoothed particle hydrodynamics. *Annu. Rev. Astronomy and Astrophysics* 30 (1992), 543–574.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2006. Position based dynamics. In *Workshop on Virtual Reality Interaction and Physical Simulation*.

Matthias Müller, Richard Keiser, Andrew Nealen, Mark Pauly, Markus Gross, and Marc Alexa. 2004. Point Based Animation of Elastic, Plastic and Melting Objects. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*. 141–151.

James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. 2002. Graphical Modeling and Animation of Ductile Fracture. *ACM Trans. Graph.* 21, 3 (2002), 291–294.

Afonso Paiva, Fabiano Petronetto, Thomas Lewiner, and Geovan Tavares. 2006. Particle-based non-Newtonian fluid animation for melting objects. In *Brazilian Symp. on Computer Graphics and Image Processing*. 78–85.

Afonso Paiva, Fabiano Petronetto, Thomas Lewiner, and Geovan Tavares. 2009. Particle-based viscoplastic fluid/solid simulation. *Computer Aided Design* 41, 4 (2009), 306–314.

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

Andreas Peer and Matthias Teschner. 2017. Prescribed Velocity Gradients for Highly Viscous SPH Fluids with Vorticity Diffusion. *IEEE Trans. Visual Comput. Graphics* to appear (2017).

Daniel Ram, Theodore Gast, Chenfanfu Jiang, Craig Schroeder, Alexey Stomakhin, Joseph Teran, and Pirouz Kavehpour. 2015. A Material Point Method for Viscoelastic Fluids, Foams and Sponges. In *ACM SIGGRAPH/Eurographics Symp. on Computer Animation*. 157–163.

Barbara Solenthaler and Renato Pajarola. 2009. Predictive-corrective incompressible SPH. In *ACM Trans. Graph.*, Vol. 28. ACM, 40.

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

Jos Stam. 2009. Nucleus: Towards a unified dynamics solver for computer graphics. In *Int. Conf. on Computer-Aided Design and Computer Graphics*. 1–11.

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A material point method for snow simulation. *ACM Trans. Graph.* 32, 4 (2013), 102.

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.

Tetsuya Takahashi, Yoshinori Dobashi, Issei Fujishiro, and Tomoyuki Nishita. 2016. Volume preserving viscoelastic fluids with large deformations using position-based velocity corrections. *The Visual Computer* 32, 1 (2016), 57–66.

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

Tetsuya Takahashi, Tomoyuki Nishita, and Issei Fujishiro. 2014. Fast simulation of viscous fluids with elasticity and thermal conductivity using position-based dynamics. *Computers & Graphics* 43 (2014), 21–30.

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 (2017), 105:1–105:11.

Demetri Terzopoulos and Kurt Fleischer. 1988. Modeling Inelastic Deformation: Viscolelasticity, Plasticity, Fracture. *SIGGRAPH Comput. Graph.* 22, 4 (1988), 269–278.

Maxime Tournier, Matthieu Nesme, Benjamin Gilles, and François Faure. 2015. Stable Constrained Dynamics. *ACM Trans. Graph.* 34, 4 (2015), 132:1–132:10.

Huamin Wang, James O'Brien, and Ravi Ramamoorthi. 2010. Multi-resolution Isotropic Strain Limiting. *ACM Trans. Graph.* 29, 6 (2010), 156:1–156:10.

Marcel Weiler, Dan Koschier, and Jan Bender. 2016. Projective Fluids. In *Int. Conf. on Motion in Games*. 79–84.

Chris Wojtan and Greg Turk. 2008. Fast Viscoelastic Behavior with Thin Features. *ACM Trans. Graph.* 27, 3, Article 47 (Aug. 2008), 8 pages.

Yonghao Yue, Brennan Smith, Christopher Batty, Changxi Zheng, and Eitan Grinspun. 2015. Continuum foam: A material point method for shear-dependent flows. *ACM Trans. Graph.* 34, 5 (2015), 160:1–160:20.

Bo Zhu, Minjae Lee, Ed Quigley, and Ronald Fedkiw. 2015. Codimensional non-Newtonian Fluids. *ACM Trans. Graph.* 34, 4, Article 115 (July 2015), 9 pages.

Received June 2017; revised August 2017; final version September 2017; accepted September 2017