Projective Fluids

Marcel Weiler
Graduate School CE,
TU Darmstadt

Dan Koschier
Graduate School CE,
TU Darmstadt

Jan Bender
Computer Animation Group,
RWTH Aachen University

Figure 1: *Left: Breaking dam scenario with 1.94 million particles, featuring obstacles with complex solid boundaries. Right: Fluid and cloth are simulated together in the Projective Dynamics framework.*

Abstract

We present a new method for particle based fluid simulation, using a combination of Projective Dynamics and Smoothed Particle Hydrodynamics (SPH). The Projective Dynamics framework allows the fast simulation of a wide range of constraints. It offers great stability through its implicit time integration scheme and is parallelizable in large parts, so that it can make use of modern multi core CPUs. Yet existing work only uses Projective Dynamics to simulate various kinds of soft bodies and cloth. We are the first ones to incorporate fluid simulation into the Projective Dynamics framework. Our proposed fluid constraints are derived from SPH and seamlessly integrate into the existing method. Furthermore, we adapt the solver to handle the constantly changing constraints that appear in fluid simulation. We employ a highly parallel matrix-free conjugate gradient solver, and thus do not require expensive matrix factorizations.

Keywords: Projective Dynamics, SPH, fluids, implicit integration

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

1 Introduction

Since Projective Dynamics was proposed by Bouaziz et al. [2014], it has received wide interest from the computer graphics community. Its robustness and efficiency make it a desirable method for the simulation of constrained particle systems.

Our goal is to simulate fluids. In computer graphics, SPH has become predominant for the particle based, Lagrangian simulation of fluids [Ihmsen et al. 2014b]. For our simulations, we propose a new fluid constraint derived from SPH, that adheres to the constraint structure required by Projective Dynamics.

Moreover, we adapt parts of the solver, to improve the performance of Projective Dynamics in scenes where constraints change frequently. In fluid simulation this is the case. The original Projective Dynamics method relies on Cholesky factorization to accelerate the solution of large sparse linear systems.Changes in the constraints are incorporated into the factorization by sparse Cholesky updates, but this procedure is only practical when the number of changes per time step is low. To gain the flexibility required by fluids, we employ a matrix-free conjugate gradient (CG) solver for optimization over the particle positions during the global step.

Meanwhile, our method still adheres to the Projective Dynamics framework and we can handle other constraints in a unified solver as well. We inherit the stability for stiff constraints and large time step sizes, while our solver is able take advantage of multi core systems since the conjugate gradient method is easy to parallelize.

2 Related Work

The physically-based simulation of fluids has been an important and active research topic for several decades. As most fluid simulation techniques are based on the Navier-Stokes equation several methods for a discretization were developed. Especially, Lagrangian discretizations based on SPH have become increasingly popular for interactive free-surface flows. We would like to refer to reader to the work of Bridson [2008] for a general overview of fluid simulation techniques while we recommend the state-of-the-art report of Ihmsen et al. [2014b] to gain insight into recent developments in SPH approaches in the field of computer graphics.

SPH Fluid Simulation Building on the pioneering work on SPH simulation of Monaghan [1992], Müller et al. [2003] introduced the particle-based concept for fluid simulation to the computer graphics community. The method was later extended to a spatially adaptive SPH discretization by Adams et al. [2007]. Due to the fact that a wide range of liquids is incompressible, e.g. water, several approaches to eliminate or reduce compression within the fluid were developed. Becker and Teschner [2007] presented a method that guarantees a maximum compression based on precomputed, scenario-dependent stiffness coefficients. As this method penalizes compression using forces and as it is based on explicit time-integration the maximum stable time step can become arbitrarily small for large stiffness coefficients. The first step towards im-

Motion in Games 2016, October 10-12, 2016, San Francisco, USA

plicit solvers for maintaining incompressibility was taken by Solenthaler and Pajarola [2009] proposing a predictive-corrective scheme that iteratively computes particle pressures. Following the general concept of maintaining incompressibility using implicit solves, methods based on holonomic constraints [Bodin et al. 2012], a discretization of the pressure Poisson equation [He et al. 2012; Ihmsen et al. 2014a], position based dynamics [Macklin and Müller 2013] or power diagrams [de Goes et al. 2015] were presented in the following years. Recently, a method maintaining not only constant density but also a divergence-free velocity field was proposed by Bender and Koschier [2015]. Besides the development of pressure solvers, mathematical models were developed in order to model complex physical phenomena which include the simulation of multiphase flows with high density contrast [Solenthaler and Pajarola 2008], robust solid fluid coupling [Akinci et al. 2012], versatile surface tension [Akinci et al. 2013], robust thin features [He et al. 2014] and highly viscous liquids [Peer et al. 2015; Takahashi et al. 2015; Bender and Koschier 2016].

Position Based and Projective Dynamics The concept of Position Based Dynamics (PBD) was first introduced by Müller et al. [2007] for the simulation of cloth and cloth balloons. Over the years, the constraint based concept of PBD evolved from a simple and robust simulation method for cloth to a fully fledged framework supporting a variety of constraints developed in the following years. Position-based methods range from approaches for hair, fur and rods [Bender et al. 2015] over cloth and solids [Bender et al. 2014a; Deul et al. 2014] and even to fluids [Macklin and Müller 2013], just to mention a few. Moreover, two-way coupling between any of the approaches can be easily realized [Bender et al. 2015]. For a complete survey on position-based approaches we would like to refer the reader to the state-of-the-art report of Bender et al. [2014b].

Besides all the advantages, PBD cannot accurately handle soft constraints as they are usually enforced through early termination of the iterative constraint solver prior to convergence. While the resulting dynamic behavior is still visually appealing, material properties, e.g. stiffness, are hard to control as they solely rely on the time step size and the solver's residual. In order to solve that problem, Bouaziz et al. [2014] presented the Projective Dynamics framework for general constraints building upon the work for optimization integrator based simulation of mass-spring systems of Liu et al. [2013]. Through reformulation of the implicit Euler scheme as optimization problem and simplification of elastic energies to quadratic distance functions, they simulate elastic objects using a local/global alternating minimization technique. While the local step is intended to satisfy all underlying constraints independently by position alteration following the PBD concept, the global step can be interpreted as "smart" averaging of the computed goal positions which also accounts for inertia terms. This results in a stable and robust simulation of soft constraints for elastic objects. Recently, the convergence of both PBD and Projective Dynamics was improved by Wang et al [2015] using a Chebyshev semi-iterative approach while Narain et al. [2016] generalized Projective Dynamics using an alternating direction method of multipliers for optimization in order to simulate general nonlinear constitutive models.

In contrast to previous work, we present a novel, fully implicit approach based on Projective Dynamics for the simulation of fluids. We formulate an SPH based constraint that seamlessly integrates into the framework and efficiently solve the linear equation system in the global step using a matrix-free conjugate gradient solver. We demonstrate the robustness, stability and efficiency of our approach on complex scenarios with thousands of particles.

3 Projective Fluids Solver

Our fluid simulation method is based on Projective Dynamics and SPH. In this section we describe how we extend Projective Dynamics to handle our fluid constraints.

3.1 Projective Dynamics

Implicit Euler Time Integration Projective Dynamics uses implicit time integration to evolve a system of particles over time, subject to constraints $C(\mathbf{x}) = 0$. Given a vector $\mathbf{x}^{(n)} \in \mathbb{R}^{3m}$ of m stacked particle positions at time step n and their velocities as $\mathbf{v}^{(n)} \in \mathbb{R}^{3m}$, the state of the system at the next time step is calculated as

$$\begin{aligned} \mathbf{x}^{(n+1)} &= \mathbf{x}^{(n)} + \Delta t \mathbf{v}^{(n+1)} \\ \mathbf{v}^{(n+1)} &= \mathbf{v}^{(n)} + \Delta t \mathbf{M}^{-1} \left(\mathbf{f}_{\text{ext}} + \mathbf{f}_{\text{int}} \left(\mathbf{x}^{(n+1)} \right) \right), \end{aligned} \tag{1}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{f}_{\text{ext}}$ is the sum of external forces like gravity, $\mathbf{f}_{\text{int}}(\mathbf{x})$ is the sum of internal (constraint) forces and Δt denotes the time step size. The internal forces try to minimize the constraint potentials $Z_i(\mathbf{x})$, which are introduced in the following section, such that $\mathbf{f}_{\text{int}}(\mathbf{x}) = -\sum_i \nabla Z_i(\mathbf{x})$. Martin et al. [2011] show that Equation (1) can be transformed into an optimization problem:

$$\mathbf{x}^{(n+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2\Delta t^2} \left\| \mathbf{M}^{\frac{1}{2}} \left(\mathbf{x} - \mathbf{s}^{(n)} \right) \right\|_F^2 + \sum_i \nabla Z_i(\mathbf{x}), \tag{2}$$

where $\|.\|_F$ denotes the Frobenius norm and $\mathbf{s}^{(n)} = \mathbf{x}^{(n)} + \Delta t \mathbf{v}^{(n)} + \Delta t^2 \mathbf{M}^{\frac{1}{2}} \mathbf{f}_{\text{ext}}$ is a term describing the particle movement without internal forces. The values of $\mathbf{x}$ that minimize Equation (2) are the particle positions for the next time step.

Quadratic Constraint Potentials In general, the constraint potentials $Z_i(\mathbf{x})$ are defined as a material model $\Psi_i(\cdot)$, applied to a strain $E_i(\mathbf{x})$, such that $Z_i(\mathbf{x}) = \Psi_i(E_i(\mathbf{x}))$. $\Psi_i(\cdot)$ can be highly nonlinear, which makes Newton's method the first choice to find the minimum of Equation (2). Unfortunately, this involves finding the Hessian of a system of nonlinear equations in every iteration, which is very expensive. Bouaziz et al. [2014] therefore replace the highly nonlinear potentials with specially designed quadratic potentials. They notice that the set of rest states of a constraint, the so-called constraint manifold, is independent of its potential. For each constraint they introduce an auxiliary variable $\mathbf{p}$. $\mathbf{p}$ contains the positions of the particles in the constraint, which are projected onto the constraint manifold. The resulting constraint potential is a quadratic distance measure between the current and the projected positions:

$$Z(\mathbf{x}, \mathbf{p}) = \frac{w}{2} \|\mathbf{D}\mathbf{x} - \mathbf{P}\mathbf{p}\|_F^2 + \delta_C(\mathbf{p}). \tag{3}$$

$\mathbf{D}$ and $\mathbf{P}$ are constant matrices and $w \geq 0$ is a weight for the constraint. $\delta_C(\mathbf{p})$ is an indicator function that is zero if $\mathbf{p}$ lies on the constraint manifold and ∞ otherwise. The indicator is used to formalize the requirement that $\mathbf{p}$ should lie on the constraint manifold.

Alternating iterative solver With the new constraint potentials from Equation (3), the potential that has to be minimized becomes

$$\begin{aligned} &\frac{1}{2\Delta t^2} \left\| \mathbf{M}^{\frac{1}{2}} \left(\mathbf{x} - \mathbf{s}^{(n)} \right) \right\|_F^2 \\ &\qquad + \sum_i \frac{w_i}{2} \|\mathbf{D}_i \mathbf{S}_i \mathbf{x} - \mathbf{P}_i \mathbf{p}_i\|_F^2 + \delta_{C_i}(\mathbf{p}_i) \end{aligned} \tag{4}$$

with $\mathbf{S}_i$ being a selector matrix that selects the particles belonging to constraint i. The optimization now has to be performed over the positions $\mathbf{x}$ and the auxiliary variables $\mathbf{p}$. By minimizing Equation (4) for $\mathbf{x}$ and $\mathbf{p}$ independently in turn, the global minimum is found. This approach is a form of block coordinate descent and will converge for any pseudo-convex potential [Tseng 2001].

In the *local step*, the particle positions $\mathbf{x}$ are kept fixed, while the minimization for $\mathbf{p}_i$ is performed by projecting the auxiliary variables onto the respective constraint manifold. Since each constraint is treated independently, this step can be easily parallelized.

During the *global step*, the $\mathbf{p}_i$ are kept fixed. Equation (4) then becomes a simple quadratic potential and can be solved by finding the point where its derivative becomes zero. This leads to the system of linear equations

$$\left(\frac{\mathbf{M}}{\Delta t} + \sum_i w_i \mathbf{S}_i^T \mathbf{D}_i^T \mathbf{D}_i \mathbf{S}_i\right) \mathbf{x} = \frac{\mathbf{M}}{\Delta t}\mathbf{s}^{(n)} + \sum_i w_i \mathbf{S}_i^T \mathbf{D}_i^T \mathbf{P}_i \mathbf{p}_i. \tag{5}$$

The matrix on the left hand side of the equation is constant as long as the constraints do not change. In this case, performing Cholesky factorization in a preprocessing step allows for an efficient solve during the simulation.

3.2 The Fluid Constraints

In typical SPH fluid simulations, a particle i is evolved in time, subject to the constraints that the internal pressure at its location should be zero. The relation between pressure $p_i(\mathbf{x})$ and fluid density $\rho_i(\mathbf{x})$ at the particle is modeled by an equation of state (EOS). There are different EOS in use, we chose

$$p_i(\mathbf{x}) = \frac{\rho_i(\mathbf{x})}{\rho_0} - 1,$$

where ρ_0 is the fluid's rest density. Requiring $p_i(\mathbf{x}) = 0$ formalizes the constraint that the fluid density should remain constant. Since it only depends on the particle positions, this constraint has exactly the form that is required by Projective Dynamics. We, therefore, define our fluid constraints as

$$C_i(\mathbf{x}) = \frac{\rho_i(\mathbf{x})}{\rho_0} - 1. \tag{6}$$

To estimate ρ_i at each particle, we use SPH. At point $\mathbf{x}_i$ the density ρ_i can be interpolated from the n neighboring particles using

$$\rho_i = \sum_{j=1}^{n} m_j W_{ij}, \tag{7}$$

where m_j is the mass of particle j and $W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j, h)$ is a kernel function with h being the smoothing length of the kernel. In practice, kernels are chosen to be spherical and have compact support. This way, only the n neighboring particles inside the support radius have to be considered in the interpolation. Furthermore, the kernel needs to be normalized and its shape should be close to a Gaussian [Monaghan 1992] to offer best interpolation results.

Since only the n particles inside the support radius contribute to the density, only these particles are part of a fluid constraint in our solver. Thus, from equations (6) and (7) we get the fluid constraint

$$C_i(\mathbf{x}) = \frac{1}{\rho_0}\left(\sum_{j=1}^{n} m_j W_{ij}\right) - 1. \tag{8}$$

To avoid the problem of particle deficiencies at frees surfaces, we restrict the constraint to positive values. This approach is equivalent to clamping negative pressures to zero, a common solution in SPH free surface flow, see e.g. [Ihmsen et al. 2014a], [Bender and Koschier 2016].

Note that since particle neighborhoods change, our constraints may contain different particles in each time step.

3.3 Constraint Projection

Now that we have established our fluid constraint, we need to find a way to project it onto its constraint manifold. This means we want to find auxiliary variables $\mathbf{p}_i = \mathbf{S}_i\mathbf{x} + \Delta\mathbf{p}_i$, for each constraint i, such that $C_i(\mathbf{p}_i) = 0$. In the following paragraphs we drop the subscript for better readability.

We solve this problem by linearizing the constraint and iteratively approaching its rest state in a Newton fashion, an idea also known from Position Based Dynamics [Müller et al. 2007] and discussed in depth in [Bender et al. 2014b]. As a first guess, we set $\mathbf{p}^{(0)} = \mathbf{S}_i\mathbf{x}$. In the n-th iteration, the linearization yields

$$C(\mathbf{p}) \approx C(\mathbf{p}^{(n)}) + \nabla_{\mathbf{p}} C(\mathbf{p}^{(n)})^T \Delta\mathbf{p}^{(n)}. \tag{9}$$

D'Alembert's principle restricts $\Delta\mathbf{p}^{(n)}$ to the constraint gradient:

$$\Delta\mathbf{p}^{(n)} = \lambda \nabla_{\mathbf{p}} C(\mathbf{p}^{(n)}), \tag{10}$$

where the scalar λ is a Lagrange multiplier. By inserting Equation (10) into Equation (9) and solving for λ, we get

$$\lambda = \frac{C(\mathbf{p}^{(n)})}{\nabla_{\mathbf{p}} C(\mathbf{p}^{(n)})^T \nabla_{\mathbf{p}} C(\mathbf{p}^{(n)})}. \tag{11}$$

To calculate the approximate position correction we need the constraint gradient. Taking the derivative of Equation (8) with respect to $\mathbf{p}$ reveals

$$\nabla_{\mathbf{p}} C(\mathbf{p}^{(n)}) = \frac{1}{\rho_0}\sum_{j=1}^{n} m_j \nabla_{\mathbf{p}} W_{ij},$$

with which the position correction becomes

$$\Delta\mathbf{p}^{(n)} = \frac{1}{\rho_0}\sum_{j=1}^{n} m_j \lambda \nabla_{\mathbf{p}} W_{ij}.$$

Because of the linearization performed in Equation (9), $C(\mathbf{p}^{(n)} + \Delta\mathbf{p}^{(n)})$ will in general not be zero. Therefore, the correct projected position is found by iterative updates $\mathbf{p}^{(n+1)} = \mathbf{p}^{(n)} + \Delta\mathbf{p}^{(n)}$. The iteration continues until $C(\mathbf{p}^{(n)} + \Delta\mathbf{p}^{(n)}) < \epsilon$, with ϵ being a small constant. In our experiments, we found that on average three to four iterations were sufficient to reach $\epsilon = 10^{-14}$. Only in rare cases more than ten iterations were required. Algorithm 1 illustrates the iterative projection procedure.

Note that in Macklin et al. [2013] take a similar approach, but in their work the constraints are not independently projected onto their rest states. Instead, after each solver iteration $\mathbf{x}$ is updated so that intermediate particle positions are known to all constraints.

3.4 Linear System Solve

After all auxiliary variables have been projected onto their corresponding constraint manifold, we have to solve Equation (5) for $\mathbf{x}$ in the *global step*. We chose the matrices $\mathbf{D}$ and $\mathbf{P}$ to

Algorithm 1 The constraint projection algorithm.

function PROJECTCONSTRAINT($\mathbf{p}$) ▷ call with $\mathbf{p} = \mathbf{S}_i\mathbf{x}$
  $C \Leftarrow$ calcConstraintPotential $(\mathbf{p})$
  while $C > \epsilon$ **do**
    $\nabla C \Leftarrow$ calcConstraintGradient $(\mathbf{p})$
    if $||\nabla C||$ **is** 0 **then**
      break ▷ already fond a minimum
    $\mathbf{p} \Leftarrow \mathbf{p} - \frac{C}{\|\nabla C\|^2}\nabla C$ ▷ apply position correction
    $C \Leftarrow$ calcConstraintPotential $(\mathbf{p})$
  return $\mathbf{p}$

be the identity matrices. Experiments with differential coordinate matrices, as proposed by Bouaziz et al. [2014], have not shown a significant performance increase. The system matrix thus becomes $\mathbf{A} = \frac{\mathbf{M}}{\Delta t^2} + \sum_i w_i \mathbf{S}_i^T \mathbf{S}_i$, and the right hand side becomes $\mathbf{b} = \frac{\mathbf{M}}{\Delta t^2}\mathbf{s}^{(n)} \sum_i w_i \mathbf{S}_i^T \mathbf{p}_i$.

Projective Dynamics owes much of its performance to the fact that the system matrix is constant and can be factorized in advance. This allows the global step to be solved very efficiently. In our fluid simulation on the other hand, the constraints change in every time step as particles flow past each other and their neighborhoods change. The system matrix has to be updated in each step, and so does the Cholesky factorization. A step that has previously been considered preprocessing has now become performance-critical. When only a small number of constraints change, sparse Cholesky updates can be used to avoid a complete refactorization, but in our fluid simulation almost all constraints change each time step. In practice this is unfeasible, and so we decided to use a matrix-free conjugate gradient (CG) solver instead.

CG is a popular iterative algorithm for solving large, sparse systems of linear equations. For an excellent derivation of the algorithm we refer the reader to [Shewchuk 1994]. CG is used to solve sparse, linear systems of the form $\mathbf{Ax} = \mathbf{b}$ for a square, symmetric, positive definite matrix $\mathbf{A}$. The main advantage of CG for our simulation is, that it only references $\mathbf{A}$ through its multiplication with a vector.We make use of this fact by providing only the result of the matrix-vector multiplication and not the matrix $\mathbf{A}$ itself. Similarly, we can calculate the right hand side vector without building any matrix for $\mathbf{b}$. Our matrix-free algorithm is illustrated in Algorithm 2. Note that we parallelize over the constraints and, therefore, need to apply updates to the result vectors atomically. Since the number of concurrent threads on a modern CPU is significantly lower than the number of constraints, threads only rarely block each other.

3.5 Inherited Traits

Our solver inherits several important traits from Projective Dynamics. The first notable property is that there are no hard constraints in Projective Dynamics. This leaves our fluid slightly compressible. Since we use implicit time integration we can combat this problem with high stiffness parameters w_i. Even with large time step sizes, we do not have to worry about stability. We adapt the time step size during our simulation according to the CFL condition.

4 Implementation Details

We implemented our simulation framework in C++. We employ the Eigen 3.2 library for basic linear algebra computations and use Intel Threading Building Blocks for parallelization. By nature, matrix-free CG consists only of vector operations which can be trivially parallelized. Parallelization of the local step is straightforward as well, since the constraints in Projective Dynamics are viewed as

Algorithm 2 The functions to calculate the product $\mathbf{r} = \mathbf{\Sigma x}$ and the right hand side vector $\mathbf{b}$ in a matrix-free fashion.

function MATRIXFREEATIMES($\mathbf{x}$)
  $\mathbf{r} \Leftarrow \mathbf{0}$ ▷ to accumulate system matrix times vector
  for each C_i **do in parallel**
    for $j \Leftarrow 1$ **to** number of particles in C_i **do**
      $l \Leftarrow$ global index of particle j
      atomic $\mathbf{r}_l \Leftarrow \mathbf{r}_l + w_i\mathbf{x}_l$
  for $i \Leftarrow 1$ **to** number of particles **do in parallel**
    $\mathbf{r}_i \Leftarrow \mathbf{r}_i + \frac{\mathbf{M}_{ii}}{\Delta t^2}\mathbf{x}_i$
  return $\mathbf{r}$
function MATRIXFREEB($\mathbf{x}$)
  $\mathbf{b} \Leftarrow \mathbf{0}$ ▷ to accumulate right hand side vector
  for each C_i **do in parallel**
    $k \Leftarrow$ number of particles in C_i
    $\mathbf{p} \Leftarrow$ empty vector of size $3k$
    for $j \Leftarrow 1$ **to** k **do**
      $l \Leftarrow$ global index of particle j
      $\mathbf{p}_j \Leftarrow \mathbf{x}_l$
    $\mathbf{p} \Leftarrow$ PROJECTCONSTRAINT$(\mathbf{p})$
    for $j \Leftarrow 1$ **to** k **do**
      $l \Leftarrow$ global index of particle j
      atomic $\mathbf{b}_l \Leftarrow \mathbf{b}_l + w_i\mathbf{p}_j$
  for $i \Leftarrow 1$ **to** number of particles **do in parallel**
    $\mathbf{b}_i \Leftarrow \mathbf{b}_i + \frac{\mathbf{M}_{ii}}{\Delta t^2}\mathbf{s}_i^{(n)}$ ▷ inertia term
  return $\mathbf{b}$

independent. Each constraint is projected onto its constraint manifold and their influence on the right hand side of the system of linear equations is updated atomically.

To accelerate the search for neighboring particles, we use the parallel spatial hashing algorithm proposed by Ihmsen et al. [2011]. Collisions with static objects were modeled using solid boundary particles. To get a good distribution of particles on an arbitrary object, Poisson-Disk samples are created on the surface of an input mesh. These particles are then used in the density estimation of SPH. We use the corrected density computation proposed by Akinci et al. [2012] to account for the inherently irregular sampling and to not have to sample the object's volume. For the density calculation we use the cubic spline kernel proposed by Monaghan et al. [1992]. It is cheap to compute and works well inside the fluid, but SPH will inherently underestimate the fluid density at free surfaces. This leads to negative pressure values and causes particles near the surface to clump together. To circumvent this problem, we clamp the pressure to non-negative values, an approach that is popular in computer graphics [Ihmsen et al. 2014b]. After each simulation step artificial viscosity is applied. We use the XSPH variant of the viscosity term proposed by Schechter et al. [2012]. This formulation was already successfully used by Bender and Koschier [2016].

5 Results

In this section we present the results of our simulations and discuss their meaning. The scenarios we chose here each concern the reproduction of desired effects, or properties of our solver.

A common test case in fluid simulation for computer graphics is the dam break. Figure 2 shows our version of the scenario. The breaking and overtaking waves that can be observed here are a desirable result and typical for liquids. A closely related scenario is the double breaking dam. In Figure 3 we show the simulation of two fluid blocks that start with a diagonal offset. On collision, the simulated liquid clearly forms the expected splashes and thin sheets.

Figure 2: *A breaking dam scenario. A block of water flows under gravity and shows typical breaking waves.*

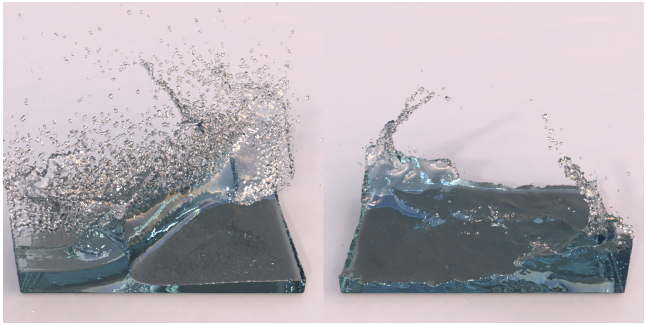

Figure 3: *A diagonal double dam break scenario in a rectangular domain shows typical splashes and thin sheets.*

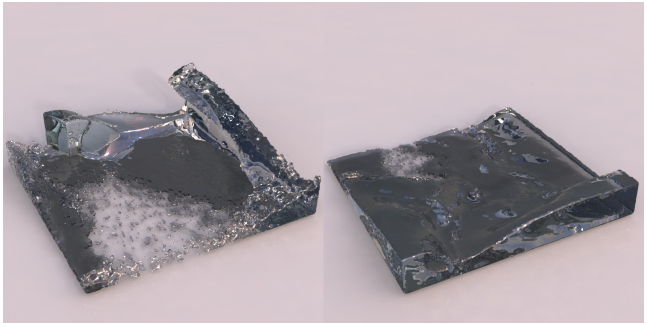

Figure 4: *Water spurts from an inlet into a box. Even at higher velocities and constraint stiffness values our solver remains stable.*

The scenario depicted in Figure 4 is a test for the stability of our method. Since Projective Dynamics only simulates soft constraints, we use high stiffness values in the range of $w_i = 10^6$ to counter compression of the fluid. Nevertheless we are able to stably simulate this scenario with an average time step size of more than 6ms (see Table 1). Still the maximum density never rises more than 0.01% over the rest density during the whole simulation.

Since we are using the boundary handling method proposed by Akinci et al. [2012], we are able to handle collisions with arbitrary solid objects. An example of the interaction between fluid and complex boundaries is shown in Figure 1, left. The right side shows an experimental unified simulation of our fluids with other Projective Dynamics constraints. Furthermore, the artificial viscosity term shows that our framework can easily be combined with other SPH methods.

Timings Table 1 illustrates the performance of our solver on the different scenarios described above. All measurements were run on a computer with two Intel Xeon E5-2697 processors with 12 cores each, clocked at 2.7GHz, and 64GB of RAM. Our solver performed a maximum of 30 local/global iterations, while constraint projection and conjugate gradient solver ran until convergence.

Scene	# fluid particles	avg. time per step	avg. time step size
three dragons	1945600	106.29s	0.00210s
double breaking dam	240000	12.73s	0.00174s
breaking dam	200000	11.30s	0.00207s
inlet	90000	4.47s	0.00622s
cloth	64000	2.92s	0.00122s

Table 1: *Time measurements for different simulations.*

Adaptive Δt While the employed implicit time integration is unconditionally stable, the time step size is still limited by the CFL condition. Fluid behavior can only be correctly recreated when particles travel less than their radius in one time step. Otherwise particle collisions may be missed. This means that the time step size has to be lowered when the fluid flows faster. When the fluid is moving slowly on the other hand, we can be generous with the time step size used in our simulation. In all videos we allow a maximum time step size of $\Delta t_{\text{max}} = 0.01s$ and adapt it depending on the maximum particle velocity in the scene. The average time step sizes for our different scenarios are shown in Table 1.

6 Conclusion and Future Work

In this paper we have presented a new method for Lagrangian fluid simulation, that uses Projective Dynamics to resolve our SPH based pressure constraints. In contrast to other implicit SPH solvers we maintain the ability to simulate compressible fluids. Still we can keep the density deviation below 0.01% by using high stiffness values, while our implicit time integration allows us to use large Δt. Since in its core our solver still operates in the Projective Dynamics framework, fluid and other constraints can be solved in a unified manner. We employ a matrix-free CG solver to efficiently handle constraints that change in every time step. Due to the highly parallel nature of CG and Projective Dynamics, our algorithm can make use of modern multi core CPUs. Matrix free CG also shows promising performance on the GPU [Weber et al. 2013]. Consequently, we are considering an implementation on graphics hardware for further speedup.

In future, we are planning to scrutinize the performance of our method in complex scenarios with many different kinds of constraints. First experiments with the simulation of interactions between fluid and soft bodies have shown the potential of this approach. The reproduction of additional fluid properties, like surface tension, is something we want to investigate as well.

Acknowledgments

The work of the authors is supported by the Excellence Initiative of the German Federal and State Governments and the Graduate School of Computational Engineering at TU Darmstadt. The dragon model is courtesy of the Stanford Computer Graphics Lab.

References

ADAMS, B., PAULY, M., KEISER, R., AND GUIBAS, L. J. 2007. Adaptively Sampled Particle Fluids. *ACM Trans. on Graphics 26*, 3, 48.

AKINCI, N., IHMSEN, M., AKINCI, G., SOLENTHALER, B., AND TESCHNER, M. 2012. Versatile Rigid-Fluid Coupling for Incompressible SPH. *ACM Trans. on Graphics 31*, 4, 62:1–62:8.

AKINCI, N., AKINCI, G., AND TESCHNER, M. 2013. Versatile Surface Tension and Adhesion for SPH Fluids. *ACM Trans. on Graphics 32*, 6, 1–8.

BECKER, M., AND TESCHNER, M. 2007. Weakly Compressible SPH for Free Surface Flows. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 1–8.

BENDER, J., AND KOSCHIER, D. 2015. Divergence-Free Smoothed Particle Hydrodynamics. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 1–9.

BENDER, J., AND KOSCHIER, D. 2016. Divergence-Free SPH for Incompressible and Viscous Fluids. *IEEE Trans. on Visualization and Computer Graphics*.

BENDER, J., KOSCHIER, D., CHARRIER, P., AND WEBER, D. 2014. Position-Based Simulation of Continuous Materials. *Computers & Graphics 44*, 1, 1–10.

BENDER, J., MÜLLER, M., AND MACKLIN, M. 2014. A Survey on Position-Based Simulation Methods in Computer Graphics. *Computer Graphics Forum 33*, 6, 228–251.

BENDER, J., MÜLLER, M., AND MACKLIN, M. 2015. Position-based simulation methods in computer graphics. In *Eurographics 2015 Tutorials*, Eurographics Association.

BODIN, K., LACOURSIÈRE, C., AND SERVIN, M. 2012. Constraint fluids. *IEEE Trans. on Visualization and Computer Graphics 18*, 516–526.

BOUAZIZ, S., MARTIN, S., LIU, T., KAVAN, L., AND PAULY, M. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. on Graphics 33*, 4, 1–11.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. A K Peters / CRC Press.

DE GOES, F., WALLEZ, C., HUANG, J., PAVLOV, D., AND DESBRUN, M. 2015. Power Particles: An incompressible fluid solver based on power diagrams. *ACM Trans. on Graphics 34*, 4, 50:1–50:11.

DEUL, C., CHARRIER, P., AND BENDER, J. 2014. Position-based rigid body dynamics. *Computer Animation and Virtual Worlds 27*, 2, 103–112.

HE, X., LIU, N., LI, S., WANG, H., AND WANG, G. 2012. Local Poisson SPH for Viscous Incompressible Fluids. *Computer Graphics Forum 31*, 1948–1958.

HE, X., WANG, H., ZHANG, F., WANG, H., WANG, G., AND ZHOU, K. 2014. Robust Simulation of Sparsely Sampled Thin Features in SPH-Based Free Surface Flows. *ACM Trans. on Graphics 34*, 1, 7:1–7:9.

IHMSEN, M., AKINCI, N., BECKER, M., AND TESCHNER, M. 2011. A Parallel SPH Implementation on Multi-Core CPUs. *Computer Graphics Forum 30*, 1, 99–112.

IHMSEN, M., CORNELIS, J., SOLENTHALER, B., HORVATH, C., AND TESCHNER, M. 2014. Implicit Incompressible SPH. *IEEE Trans. on Visualization and Computer Graphics 20*, 3, 426–435.

IHMSEN, M., ORTHMANN, J., SOLENTHALER, B., KOLB, A., AND TESCHNER, M. 2014. SPH Fluids in Computer Graphics. *Eurographics (State of the Art Reports)*, 21–42.

LIU, T., BARGTEIL, A. W., BRIEN, J. F. O., AND KAVAN, L. 2013. Fast Simulation of Mass-Spring Systems. *ACM Trans. on Graphics 32*, 6, 214:1–214:7.

MACKLIN, M., AND MÜLLER, M. 2013. Position Based Fluids. *ACM Trans. on Graphics 32*, 4, 1–5.

MARTIN, S., THOMASZEWSKI, B., GRINSPUN, E., AND GROSS, M. 2011. Example-based Elastic Materials. *ACM Transactions on Graphics 30*, 4, 72:1–72:8.

MONAGHAN, J. 1992. Smoothed Particle Hydrodynamics. *Annual review of astronomy and astrophysics 30*, 1, 543–574.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-Based Fluid Simulation for Interactive Applications. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 154–159.

MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. 2007. Position Based Dynamics. *Visual Communication and Image Representation 18*, 2, 109–118.

NARAIN, R., OVERBY, M., AND BROWN, G. E. 2016. ADMM $\supseteq$ Projective Dynamics: Fast Simulation of General Constitutive Models. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 1–8.

PEER, A., IHMSEN, M., CORNELIS, J., AND TESCHNER, M. 2015. An Implicit Viscosity Formulation for SPH Fluids. *ACM Trans. on Graphics 34*, 4, 1–10.

SCHECHTER, H., AND BRIDSON, R. 2012. Ghost SPH for Animating Water. *ACM Trans. on Graphics 31*, 4, 61:1–61:8.

SHEWCHUK, J. 1994. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Tech. rep.

SOLENTHALER, B., AND PAJAROLA, R. 2008. Density Contrast SPH Interfaces. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 211–218.

SOLENTHALER, B., AND PAJAROLA, R. 2009. Predictive-corrective Incompressible SPH. *ACM Trans. on Graphics 28*, 3, 40:1–40:6.

TAKAHASHI, T., DOBASHI, Y., FUJISHIRO, I., NISHITA, T., AND LIN, M. 2015. Implicit Formulation for SPH-based Viscous Fluids. *Computer Graphics Forum 34*, 2, 493–502.

TSENG, P. 2001. Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization. *Journal of Optimization Theory and Applications 109*, 3, 475–494.

WANG, H. 2015. A Chebyshev Semi-iterative Approach for Accelerating Projective and Position-based Dynamics. *ACM Trans. on Graphics 34*, 6, 246:1–246:9.

WEBER, D., BENDER, J., SCHNOES, M., STORK, A., AND FELLNER, D. 2013. Efficient GPU Data Structures and methods to Solve Sparse Linear Systems in Dynamics Applications. *Computer Graphics Forum 32*, 1, 16–26.