ISSN 1213-6972 Journal of WSCG http://www.wscg.eu Vol.28, No.1-2, 2020

Chebyshev's Method on Projective Fluids

Alexander Sommer[1] alexander.sommer@hs-rm.de
Ulrich Schwanecke[1] ulrich.schwanecke@hs-rm.de
Elmar Schoemer[2] schoemer@uni-mainz.de

[1] Computer Vision and Mixed Reality Group, RheinMain University of Applied Sciences Wiesbaden Rüsselsheim, Germany
[2]Institute of Computer Science, Johannes Gutenberg University Mainz, Germany

ABSTRACT

We demonstrate the acceleration potential of the Chebyshev semi-iterative approach for fluid simulations in Projective Dynamics. The Chebyshev approach has been successfully tested for deformable bodies, where the dynamical system behaves relatively linearly, even though Projective Dynamics, in general, is fundamentally nonlinear. The results for more complex constraints, like fluids, with a particular nonlinear dynamical system, remained unknown so far. We follow a method describing particle-based fluids in Projective Dynamics while replacing the Conjugate Gradient solver with Chebyshev's method. Our results show that Chebyshev's method can be successfully applied to fluids and potentially other complex constraints to accelerate simulations.

Keywords: fluid simulation, constraint-based simulation, projective dynamics, nonlinear optimization, animation

1 INTRODUCTION

The physically plausible simulation of the behavior of various objects has been an important research topic in computer graphics for the past decades. In the early days, methods were adapted from simulations in computational physics, like the Finite-Element-Methods (FEM) [14]. These methods are derived from continuum mechanical principles and therefore offer high accuracy. This high accuracy is needed because the goal of simulations in computational physics or chemistry is to replace real-world experiments. In computer graphics, however, physical simulations are mostly used for special effects in film or for interactive applications. High numerical accuracy is therefore often not needed as long as the objects behave plausibly. This results in new methods that are specially tailored to the needs of computer graphics simulation.

A common simulation method in computer graphics is Position-Based-Dynamics (PBD) [13]. Classical approaches use forces to apply a change of momentum in a physical system, which leads to a position change through numerical integration of velocities and accelerations. In PBD there are specially designed constraints, that enforce a change of position in a quasi-static fashion. Constraints in PBD can be used for simulating a broad variety of objects such as rigid-bodies, soft-bodies, cloth, hair, muscles and even fluids. Unfortunately, this technique has limited accuracy as the constraints are not directly derived from continuum mechanical principles. In addition, a large number of iterations are often required to solve the entire system, as the constraints are projected onto each other.

Projective Dynamics (PD) [3] is a relatively new technique that also uses constraints to enforce position changes. But in contrast to PBD, these constraints are highly nonlinear energy potentials directly derived from deformation energies in continuum mechanics. PD bridges the gap between the physically correct FEM and the fast, reliable and robust PBD. Since the emergence of PD, research has continued to expand the number of objects that can be simulated, improve convergence rate and increase speed. The Chebyshev Semi-Iterative approach [17] is one method of accelerating PD by using concepts from linear algebra to efficiently solve the fundamentally nonlinear system in PD. This approach has been successfully applied to deformable bodies, which have a relatively linear behavior.

In this work, we demonstrate the acceleration potential of Chebyshev's method on fluids, which in contrast to rigid or even deformable bodies are particularly complex and result in far more nonlinear dynamic systems. To the best of our knowledge, we are the first to successfully accelerate fluid simulation with Chebyshev's method. We use the approach proposed in [18] for simulating fluids in PD, borrowing Smoothed Particle Hydrodynamics (SPH) methods. Our results also show the potential of Chebyshev's method for other complex constraints in PD (e.g. hair or cloth).

2 RELATED WORK

In the following, we briefly introduce the concept of Projective Dynamics and its extension to fluids. In particular, we also present the notation used, which is required in the section discussing the Chebyshev method.

2.1 Projective Dynamics

PD is a constraint-based simulation method using an implicit integration scheme while projecting the constraints in an alternating Gauss-Seidel fashion onto a common solution space. An object can be described as a mesh consisting of m vertices. Each vertex $(\mathbf{x}_i, \mathbf{v}_i), i \in \{1 \dots m\}$ is characterized by a position $\mathbf{x}_i$ and a velocity $\mathbf{v}_i$. All positions and velocities of the system can be stored inside two matrices $\mathbf{X} \in \mathbb{R}^{m\times 3}$ and $\mathbf{V} \in \mathbb{R}^{m\times 3}$. The implicit time integration from time t_n to t_{n+1} of these characteristics leads to:

$$\begin{aligned} \mathbf{X}_{n+1} &= \mathbf{X}_n + h \cdot \mathbf{V}_{n+1} \\ \mathbf{V}_{n+1} &= \mathbf{V}_n + h \cdot \mathbf{M}^{-1}\mathbf{F}(\mathbf{X}_{n+1}) \end{aligned} \tag{1}$$

with h being the simulation time-step and $\mathbf{M} \in \mathbb{R}^{m\times m}$ a constant mass matrix. Forces $\mathbf{F}$ acting on the system can be characterized as internal or external. External forces $\mathbf{f}_{ext}$, like gravity, are held constant during the simulation. Internal forces can be understood as material properties and are expressed by strictly position-depended scalar energy potentials $\mathbf{f}_{int}(\mathbf{X}) = -\sum_i \nabla W_i(\mathbf{X})$. As shown in [11], for finding the unknown $\mathbf{X}_{n+1}$, the coupled system (1) can be converted into the optimization problem

$$\min_{\mathbf{X}_{n+1}} \left(\frac{1}{2h^2} \left\| \mathbf{M}^{\frac{1}{2}} (\mathbf{X}_{n+1} - (\mathbf{X}_n + h\mathbf{V}_n + h^2\mathbf{M}^{-1}\mathbf{f}_{ext})) \right\|_F^2 + \sum_i W_i(\mathbf{X}_{n+1}) \right). \tag{2}$$

One of the key points in PD is the replacement of the generally highly nonlinear energy function $W_i(\mathbf{X})$ by specially designed constraints. A fulfilled constraint describes the rest state or undeformed configuration. [3] showed that each constraint can be solved independently with an auxiliary position variable $\mathbf{P}$ by projection to a common constraint manifold minimizing the distance between the current and the projected positions, i.e.

$$W(\mathbf{X}_{n+1}) = \min_{\mathbf{P}} \frac{w}{2} \|\mathbf{C}\mathbf{X}_{n+1} - \mathbf{D}\mathbf{P}\|_F^2 + \delta_C(\mathbf{P}),$$

with constraint depended constant matrices $\mathbf{C}$ and $\mathbf{D}$, an indicator function δ_C that evaluates if the constraint is fulfilled and a weighting factor w.

For solving the whole problem, two minimization steps are necessary: a local and a global solve. During the local solve the minimization is executed over the auxiliary variables $\mathbf{P}_i$, while the positions $\mathbf{X}_{n+1}$ are kept fixed. This means that vertex positions are projected to their closest positions on the constraint manifold. Since the auxiliary variables $\mathbf{P}_i$ are independent for each constraint this can be done in parallel. In the second minimization step the already projected auxiliary variables $\mathbf{P}_i$ are fixed and (2) is minimized over the vertex positions $\mathbf{X}_{n+1}$. This finds the configuration with the best compromise between all constraints. Since the unknown $\mathbf{X}_{n+1}$ is quadratic in all terms of (2), the minimization is equivalent to solving the linear system

$$\left(\frac{\mathbf{M}}{h^2} + \sum_i w_i \mathbf{C}_i^T \mathbf{C}_i \right) \mathbf{X}_{n+1} = \frac{\mathbf{M}}{h^2} (\mathbf{X}_n + h\mathbf{V}_n) + \sum_i w_i \mathbf{C}_i^T \mathbf{D}_i \mathbf{P}_i + \mathbf{f}_{ext}. \tag{3}$$

Since the objective function is bounded for an arbitrary choice of constraints, each iteration is guaranteed to weakly decrease energies. Therefore the vertex positions $\mathbf{X}_{n+1}$ converge iteratively towards the state of minimal energy, without the need for any safeguards.

2.2 Projective Fluids

For the simulation of fluids in a constraint-based framework, a fluid density constraint can be used, which was first proposed for position based simulations in [9]. Later, [18] translated the methodology for the use in PD. They defined the (incompressible) fluid constraint as the state in which internal pressures force the continuum into a resting state of constant density ρ_0, i.e.

$$W_i(\mathbf{X}) = \frac{1}{\rho_0} \left(\sum_k m_k K(\mathbf{x}_i - \mathbf{x}_k, l) \right) - 1, \tag{4}$$

where local densities are calculated by the standard SPH density estimator as an interpolation of neighboring particles by an SPH Kernel function K with a smoothing kernel length l.

The local solve is done by finding a position correction vector $\Delta\mathbf{P}$ in the projected positions $\mathbf{P}$ that satisfies the constraint $W(\mathbf{P} + \Delta\mathbf{P}) = 0$. A first-order approximation of the constraint with a restriction of $\Delta\mathbf{P}$ in the direction of the constraint gradient that conserves the linear and angular momenta according to D'Alembert's principle leads to exactly one scalar Lagrange multiplier λ that solves the system (see [2])

$$W(\mathbf{P} + \Delta\mathbf{P}) \approx W(\mathbf{P}) + \lambda \nabla W^T \nabla W.$$

The gradient of the SPH density estimator is just defined by the gradient of the kernel function. For an arbitrary particle k two cases need to be distinguished, i.e. whether k is a neighboring particle or not and thus one gets

$$\nabla_{\mathbf{p}_k} W_i = \frac{1}{\rho_0} \begin{cases} \sum_j \nabla_{\mathbf{p}_k} K(\mathbf{p}_i - \mathbf{p}_j, l) & \text{if} \quad k = i \\ -\nabla_{\mathbf{p}_k} K(\mathbf{p}_i - \mathbf{p}_j, l) & \text{if} \quad k = j \end{cases}$$

Since the result is only a first order approximation the fluid constraint (4) in general will not vanish. Therefore iterative updates $\mathbf{P}_{\zeta+1} = \mathbf{P}_\zeta + \Delta\mathbf{P}_\zeta$ are used to converge $\mathbf{P}_\zeta$ towards the correct projected positions $\mathbf{P}$, where the (4) vanishes. The iteration can be stopped when the constraint is smaller than a predefined constant, i.e. $W(\mathbf{P}_\zeta + \Delta\mathbf{P}_\zeta) < \varepsilon$. Usually after three to four iterations $\varepsilon < 10^{-14}$ is reached, which is sufficient [18].

3 CHEBYSHEV'S METHOD

The Chebyshev method is a semi-iterative approach to solve a linear system $\mathbf{Ax} = \mathbf{b}$, like the global solve (3) in PD. Contrary to the Conjugate Gradient method (CG) proposed in [18], it doesn't depend on inner products, which offers more potential for parallelization [17], since less safeguards are needed to prevent race conditions. The key idea behind the method is to obtain better results from an iterative solution, by blending in previous results as a convex combination with blending weights $\nu_{i,k} > 0$, $\sum_{i=0}^{k} \nu_{i,k} = 1$. These blending coefficients $\nu_{i,k}$ can be determined by minimizing the error

$$\mathbf{e}_k = \mathbf{x}'_k - \mathbf{x} = \sum_{i=0}^{k} \nu_{i,k}(\mathbf{x}_i - \mathbf{x}) \tag{5}$$

where $\mathbf{x}'_k$ is the total iterative solution after k iterations, $\mathbf{x}$ is the exact solution and

$$\mathbf{x}_i = \mathbf{Q}^{-1}(\mathbf{R}\mathbf{x}_{i-1} + \mathbf{b}) \tag{6}$$

is the result of the i-th iteration, where $\mathbf{A} = \mathbf{Q} - \mathbf{R}$ and $\mathbf{Q}$ being an easily invertible matrix (e.g. a diagonal matrix). Substituting (6) into (5) results in

$$\begin{aligned}\mathbf{e}_k &= \sum_{i=0}^{k} \nu_{i,k}\left(\mathbf{Q}^{-1}(\mathbf{R}\mathbf{x}_{k-1} + \mathbf{b}) - \mathbf{x}\right) = \sum_{i=0}^{k} \nu_{i,k}\mathbf{Q}^{-1}\mathbf{R}\mathbf{e}_{k-1} \\ &= \sum_{i=0}^{k} \nu_{i,k}\left(\mathbf{Q}^{-1}\mathbf{R}\right)^i \mathbf{e}_0 = p_k(\mathbf{Q}^{-1}\mathbf{R})\mathbf{e}_0\end{aligned}$$

with $p_k(\mathbf{Y}) = \sum_i \nu_{i,k}\mathbf{Y}^i$ being a polynomial.

A polynomial of a matrix $\mathbf{Y}$ can be minimized by minimizing $\|p_k(\mathbf{Y})\|_2 = \max_{\lambda_i} |p_k(\lambda_i)|$, the analog polynomial function for the scalar Eigenvalues λ_i of matrix $\mathbf{Y}$. Finding the eigenvalues of a large linear system is not practical, but the spectral radius ρ gives a measure of the range in which the eigenvalues must lie. This means t hat p_k can be minimized over a range $x \in [-\rho, \rho]$, i.e.

$$p_k(x) = \arg\min\left(\max_{-\rho \le x \le \rho} |p_k(x)|\right). \tag{7}$$

[17] showed that $p_k(x) = T_k\left(\frac{x}{\rho}\right) / T_k\left(\frac{1}{\rho}\right)$ with the Chebyshev polynomials of the first kind $T_k(x)$ (for more details on Chebyshev Polynomials see e.g. [7]) is a solution to (7). Furthermore, [17] showed in detail that, based on relations for these polynomials, an iterative solution with a minimized error term only depending on the range limiter ρ is given by

$$\mathbf{x}'_{k+1} = \omega_{k+1}\left(\mathbf{Q}^{-1}(\mathbf{R}\mathbf{x}'_k + \mathbf{b}) - \mathbf{x}'_{k-1}\right) + \mathbf{x}'_{k-1} \tag{8}$$

with a recursive resulting factor $\omega_{k+1} = \frac{4}{4-\rho^2\omega_k}$ $\forall i \ge 1$ and $\omega_1 = 1$.

The idea of using the Chebyshev approach for PD is to replace the global solve with the Chebyshev Formula (8) using the results of the previous iterations for $\mathbf{x}'_k$ and $\mathbf{x}'_{k-1}$. Furthermore, the matrix $\mathbf{Q}$ is chosen as the diagonal matrix of the projective system matrix. For a linear system, ρ is well defined by the spectral radius of $\mathbf{Q}^{-1}\mathbf{R}$. For PD where the combination of local and global solves forms a fundamentally nonlinear system this definition of ρ can't be used any longer and an alternative approach is needed. [17] proposed to use a fixed ρ for the whole simulation. This ρ can be found in two steps before the actual simulation starts. First it is initialized by the error rate in the last iteration K of a test run, i.e. $\rho = \frac{\|\mathbf{e}_K\|_2}{\|\mathbf{e}_{K-1}\|_2}$. After that, ρ is modified in a couple of further test runs to minimize the convergence rate. When ρ is overestimated the results begin to oscillate and when ρ is underestimated the results converge slower.

4 IMPLEMENTATION

Algorithm 1 PD solve from t = $n \rightarrow$ t = $n+1$

1: $\mathbf{X}^{(0)} \leftarrow$ InitialPositionGuess($\mathbf{X}_n$, $\mathbf{V}_n$)
2: **for** iterations $k = 0 \ldots K$-1 **do**
3: **for all** constraints i **do**
4: $\mathbf{P}_i \leftarrow$ ProjectConstraint($\mathbf{C}_i$, $\mathbf{X}^{(k)}$) //(4)
5: **end for**
6: $\tilde{\mathbf{X}}^{(k+1)} \leftarrow$ SolveLinearSystem($\mathbf{X}_n$, $\mathbf{V}_n$, $\mathbf{P}_i$) //(3)
7: $\mathbf{X}^{(k+1)} = \omega_{k+1}\left(\tilde{\mathbf{X}}^{(k+1)} - \mathbf{X}^{(k-1)}\right) + \mathbf{X}^{(k-1)}$ //(8)
8: **end for**
9: $\mathbf{X}_{n+1} = \mathbf{X}^{(K)}$
10: $\mathbf{V}_{n+1} = (\mathbf{X}_{n+1} - \mathbf{X}_n)/h$

We implemented a fluid simulation framework in C++ 11, using OpenMP for parallelization and the Eigen library for mathematical operations. The parallelization is used, among other things, to project fluid constraints in parallel onto their manifolds, while updating the solution vector atomically to prevent race conditions, as described in [18]. We used parallel Spatial Hashing by [8] for finding particle neighbors (see also [15]) and a cubic b-spline kernel [12] as the SPH kernel for interpolating field quantities from neighboring particles. Boundary and object collision handling is done by surface sampled boundary particles as proposed in [1]. The particle sampling on arbitrary surfaces is done with an improved parallelized version

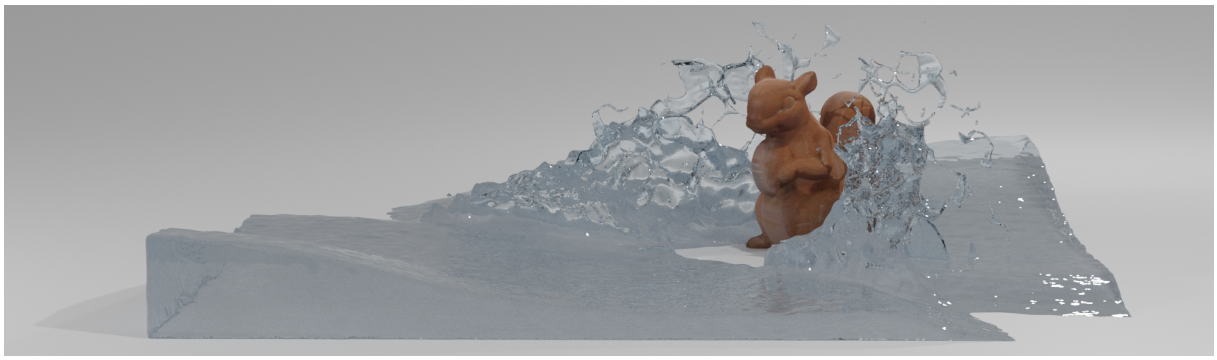

Figure 1: A diagonal double dam break in a squared bounding box with one collision object and 298k fluid particles.

Solver	*Dam Break* 30k	*Dam Break* 88k	*2Dam Break* 128k	*Squirrel* 298k
CG 1 It.	73ms	277ms	484ms	883ms
CG	94ms	616ms	889ms	2558ms
LDLT	**88ms**	306ms	486ms	1057ms
Jacobi 1 It.	89ms	**276ms**	476ms	1163ms
Jacobi	101ms	381ms	555ms	1225ms
Chebyshev	89ms	295ms	**459ms**	**995ms**

Table 1: Timing results for different scenarios. Fastest runtimes without artifacts in bold.

of Poisson disk sampling [4]. The simulation time-step size is restricted by the CFL condition [5]. The pseudo-code for one PD solve, also referred to as one time-step from a time t = n to t = $n+1$ is shown in Algorithm 1. The corresponding formulas from this paper are added in parentheses behind code lines.

Figure 2: A dam break scenario in a rectangular bounding box with 88k particles.

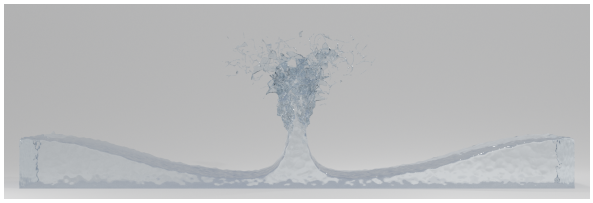

Figure 3: A double dam break scenario in a rectangular bounding box with two bodies of water on opposite sides with a total of 128k particles.

5 RESULTS

In this section, we present the simulation results and discuss the time advantage over previously used solving methods. We use four different simulation scenarios to emphasize our statements. The first one is a dam break, which is a classical test case in fluid simulation. In this setup, a block of water containing 30k particles flows under the influence of gravity in a rectangular bounding box. For the second scenario we used the same setup (see Fig. 2), but this time with 88k particles. The third setup, the double dam break, involves two blocks of water in a rectangular bounding box on opposite sides containing a total of 128k particles (see Fig. 3). In the last setup, two blocks of water with each 149k particles in opposing corners of a squared bounding box are set loose, while a squirrel as a complex collision object stands inside the bounding box (see Fig. 1).

As mentioned in section 3 the first thing that needs to be done is finding an equivalent to the spectral radius ρ that minimizes the convergence rate. Figure 4 exemplary displays the averaged runtime of the first 80 simulation time-steps for different values of ρ, for the second scenario. The minimum is found in $\rho = 0.925$. Furthermore, it shows that for an overestimated ρ the convergence rate slows significantly, probably due to oscillating results, as proposed in section 3.

For evaluating the quality of the solver we took the average runtime during one time-step. Since all compared solvers have different convergence behavior, we used a common criteria for stopping the local/global iteration. It is stopped when the maximum Euclidean distance between two corresponding particle locations in the current and the previous iteration is bellow a fixed threshold. We compared our solver to other commonly used solvers (see Table 1). For the Conjugate Gradients (CG) solver, we used the matrix-free implementation proposed in [18]. As an iterative method, we implemented a standard Jacobi method [16]. We also compared our results to a direct solving method the LDLT Cholesky decomposition [6]. For this method, we used Eigen's `SimplicialLDLT` with sparse matrices. We tested the CG and the Jacobi method until convergence for the global solve (as stated in [18]) and for a single global solve iteration only. While the CG method with only one iteration was often the fastest in our tests,

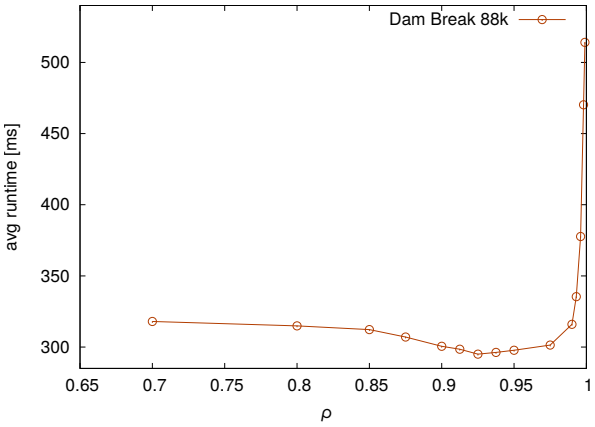

Figure 4: Average runtime for the dam break scenario with 88k particles for different values for the equivalent to the spectral radius ρ.

it produced artifacts and therefore isn't feasible. The Chebyshev method doesn't generate any artifacts and is the fastest method of all tested fully working methods for higher particle counts. This shows, that even for complex and highly nonlinear constraints, like the fluid constraint, the Chebyshev method accelerates the simulation. Compared to [18]'s CG solver (until convergence), we were able to reduce solving times between 41% and 52% without creating undesired visual side effects. All results have been produced on a MacBookPro with an Intel Core i7-4980HQ processor with 4 cores (8 threads), clocked at 2.80GHz and 16GB of RAM.

6 FUTURE WORK

In the future, we want to further investigate the acceleration potential of our solving approach on massively parallel hardware like the GPU. Since the approach does not depend on inner products we believe that there is a strong potential of increasing the performance even more. Furthermore, we plan on testing the solver in more complex simulation setups with different non-fluid constraints. Additionally, it would be interesting to see if the small substep method from [10] for extended position-based dynamics (XPBD), could also further improve Chebyshev's method in PD.

REFERENCES

[1] Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.*, 31(4):62:1–62:8, July 2012.

[2] Jan Bender, Matthias Müller, Miguel A. Otaduy, Matthias Teschner, and Miles Macklin. A survey on position-based simulation methods in computer graphics. *Comput. Graph. Forum*, 33(6):228–251, September 2014.

[3] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.*, 33(4):154:1–154:11, July 2014.

[4] John Bowers, Rui Wang, Li-Yi Wei, and David Maletz. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph.*, 29(6):166:1–166:10, December 2010.

[5] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100:32–74, 1928.

[6] Dariusz Dereniowski and Marek Kubale. Cholesky factorization of matrices in parallel and ranking of graphs. pages 985–992, 01 2003.

[7] Gene H. Golub and Richard S. Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order richardson iterative methods. *Numerische Mathematik*, 3(1):147–156, December 1961.

[8] Markus Ihmsen, Nadir Akinci, Markus Becker, and Matthias Teschner. A parallel sph implementation on multi-core cpus. *Comput. Graph. Forum*, 30:99–112, 03 2011.

[9] Miles Macklin and Matthias Müller. Position based fluids. *ACM Trans. Graph.*, 32(4):104:1–104:12, July 2013.

[10] Miles Macklin, Kier Storey, Michelle Lu, Pierre Terdiman, Nuttapong Chentanez, Stefan Jeschke, and Matthias Müller. Small steps in physics simulation. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '19, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. Example-based elastic materials. *ACM Trans. Graph.*, 30(4):72:1–72:8, July 2011.

[12] J. Monaghan and C. Lattanzio. A refined method for astrophysical problems. *Astronomy and Astrophysics*, 149:135–143, 07 1985.

[13] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109 – 118, 2007.

[14] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 137–146, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[15] Juraj Onderik. Efficient neighbor search for particle-based fluids. *Journal of the Applied Mathematics Statistics and Informatics (JAMSI)*, 2, 01 2007.

[16] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, USA, 2nd edition, 2003.

[17] Huamin Wang. A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph.*, 34(6):246:1–246:9, October 2015.

[18] Marcel Weiler, Dan Koschier, and Jan Bender. Projective fluids. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, pages 79–84, New York, NY, USA, 2016. ACM.