Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2018)
F. Jaillet, G. Zachmann, K. Erleben, and S. Andrews (Editors)

Fast Quadrangular Mass-Spring Systems using Red-Black Ordering

Pontus Pall Oskar Nylén Marco Fratarcangeli

Chalmers University of Technology, Sweden

Figure 1: Left*: a mass-spring network is partitioned into two independent sets of particles colored in red and black (*top*), and rendered as a triangulated mesh (*bottom*). During the animation, each one of these sets is efficiently computed in a single parallel step.* Middle*: a cloth-chair collision (19K springs, 2.5 ms per time step).* Right*: collision between four layers of clothes and a moving sphere (32K springs, 4.1 ms per time step). Timings include both numerical integration and collision handling.*

Abstract

We introduce a practical iterative solver for mass-spring systems which can be trivially mapped to massively parallel architectures, in particular GPUs. We employ our solver for the interactive animation of virtual cloth and show that it is computationally fast, robust and scalable, making it suitable for real-time graphics applications. Under the assumption that the input system is represented by a quadrangular network of masses connected by springs, we first partition the particles into two independent sets. Then, during the animation, the dynamics of all the particles belonging to each set is computed in parallel. This enables a full Gauss-Seidel iteration in just two parallel steps, leading to an approximated solution of large mass-spring systems in a few milliseconds. We use our solver to accelerate the solution of the popular Projective Dynamics framework, and compare it with other common iterative solvers in the current literature.

1. Introduction

Mass-spring systems are widely employed for real-time animation of a broad range of soft bodies such as clothes [Pro95, BFA02], hair [RCT91, LBOK13] and skin in virtual characters [TPBF87, HT04, ARF15]. These systems are composed by a network of masses, represented by particles, connected together by springs which govern their dynamics such that they plausibly replicate the deformations of real soft objects when subjected to external forces (Fig. 1). A mass-spring system is usually described by a large and sparse system of linear equations, which must be solved in a few milliseconds to maintain real-time interactivity. Stationary iterative solvers have become a common choice to address this problem in the domain of interactive physics-based animation [MHHR07, Sta09, MMCK14, FTP16, Wan15] because they are relatively simple to implement and provide an approximate solution of the system very rapidly. In this context, realism can be sacrificed as long as the solution is accurate enough to be believable to the final user.

In particular, the Jacobi and the Gauss-Seidel methods [Saa03] are two of the most widely used solvers. The Jacobi method is trivially parallelizable but requires a high number of iterations to converge to an acceptable solution, hindering its efficiency. The Gauss-Seidel method converges much faster but is inherently serial and as such does not map well on parallel computational ar-

DOI: 10.2312/vriphys.20181066

chitectures. Recent methods based on graph coloring enable the parallelization of the Gauss-Seidel method, removing concurrency issues such as synchronization and intercommunication between threads [FP15, FTP16].

In these approaches, the system is decomposed into independent sets of equations, and all the equations in the same set are computed in parallel leading to a significant performance speed-up. Such decomposition is carried out by using a graph coloring algorithm: the graph of constraints is colored with a *distance-1* algorithm [GR01], such that neighboring constraints have different colors. The constraints belonging to the same color do not depend on each other and can be solved independently in the same parallel step. Thus, it is desirable to use as few independent sets as possible. Finding the minimal number of sets, however, is an NP-complete problem whose complexity may reduce the effectiveness of such approaches.

Contribution. We propose a novel iterative scheme for quadrangular mass-spring systems (Fig. 1) which are employed to animate three-dimensional soft bodies such as cloths and ropes. Without loss of generality, we use the recently introduced Projective Dynamics framework [LBOK13, BML*14] to model the system. We obtain a graph which is bipartite by definition, meaning that only two sets are needed to cover the entire graph using Red-Black ordering. We show that this strategy allows a full Gauss-Seidel iteration in just two parallel steps regardless of the number of constraints in the mass-spring system. We demonstrate the computational speed and robustness of our solver, which maps very well with the hardware architecture of modern GPUs, similarly to Jacobi-based methods, retaining the convergence speed of Gauss-Seidel. This makes our solver *efficient*, *robust* and *scalable*, and thus very suitable for the real-time animation of soft bodies composed of a large number of masses and springs.

2. Related work

2.1. Simulation frameworks

As described in [TPBF87], the shape and motion of soft bodies can be effectively expressed as a set of partial differential equations based on elasticity theory. In order to solve them in the smallest amount of time, such equations are discretized, producing a system of inter-dependent linear equations. One way of discretizing continuous shapes is to sample their geometry into particles with masses, connected by constraints. Such constraints govern the dynamics of the particles during the animation. This model has been used in [Pro95] to simulate convincing non-extensible cloth, and in [Fau99] for interactive solid animation. [BW98] introduced a simulation method to handle large time steps for cloth simulation without sacrificing stability.

The Position Based Dynamics framework [MHHR07, MMCK14], presents a model to animate any type of soft body by using different types of constraints. For example, one can use bending constraints to simulate cloths with different folding behaviors, volume preservation constraints to keep the volume of tetrahedral meshes constant [BMM15], and density constraints for animating fluids [MM13]. Position Based Dynamics is fast, easy to implement and controllable. A similar approach is used in Autodesk's Nucleus framework authored by [Sta09].

The Projective Dynamics framework [LBOK13, BML*14] provides a trade-off between the simplicity and efficiency of Position Based Dynamics, with the accuracy of physically-correct methods such as the finite element method (FEM). In Projective Dynamics, constraints are defined with energy potentials derived from physical laws. In this approach, an alternating minimization technique is used to solve the constraints [BPC*11]. Recently, Projective Dynamics has been reformulated as a quasi-Newton method, which enables simulation of a large group of hyperelastic materials [LBK17].

2.2. Parallel solving

In the original formulation of Position Based Dynamics [MHHR07], the constraints are solved in an iterative, sequential Gauss-Seidel fashion and thus does not exploit the parallel capabilities of modern multi-core processors. To address this problem, Macklin and Müller used a parallel Jacobi solver to animate rigid bodies, soft bodies and fluids including collision handling [MM13, MMCK14]. In general, however, Jacobi solvers have a slow rate of convergence, and thus require a high number of iterations to provide an acceptable solution, in particular when the soft objects are composed by many constraints. Under certain circumstances, the convergence rate of Jacobi-based solvers can be significantly increased by using the Chebyshev polynomials [Wan15, WY16].

Recent works have parallelized Gauss-Seidel without having to use concurrency control techniques which slow down the execution. The concurrency problem occurs when the Gauss-Seidel solver accesses the same particle position from multiple threads. This problem can be avoided by using atomic operations, but this impacts the overall performance [BMM15]. A more efficient approach employs graph coloring as proposed by [FP15, FTP16]: the Gauss-Seidel solver is parallelized by using an approximated graph coloring technique. A graph is built, where each node represents a constraint, and two nodes are connected if they share at least one particle. The whole graph is colored in parallel using a randomized technique; each color in the graph corresponds to an independent set of constraints whose dynamics is solved in a single parallel step. The rate of convergence of Gauss-Seidel is preserved, and the computation time spent for each iteration depends on the number of colors. Therefore it is desirable to have as few colors as possible. A similar approach was used for collision handling by [TBV12] to mitigate jittering between animated rigid bodies.

Our approach always uses just two colors, generating large sets of independent equations which are solved in parallel increasing the efficiency of the whole computation.

3. Background

In this section, we describe the theoretical basis of our approach. First, we provide a brief introduction to stationary iterative solvers, namely Jacobi and Gauss-Seidel, that are used to solve large and

sparse systems of linear equations [Saa03] (Sec. 3.1). Then, we describe how the proposed schema can be applied to solve Projective Dynamics in real-time (Sec. 3.2).

3.1. Linear iterative solvers

A set of n linear equations is expressed in matrix form as:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

where $\mathbf{A}$ is a $m \times n$ matrix of scalar coefficients, $\mathbf{x}$ is a vector of n unknowns, and $\mathbf{b}$ is a vector of m entries.

The Jacobi and Gauss-Seidel methods are linear solvers employed to find $\mathbf{x}$ [BBC*94]. Starting from an initial guess of the solution $\mathbf{x}^0$, these solvers calculate an increasingly accurate approximation $\mathbf{x}^{k+1} = f(\mathbf{x}^k)$ of the correct solution by iterating over it. These iterative solvers produce a sufficiently accurate approximation very rapidly, and are therefore commonly used in real-time physics simulation.

Jacobi method. By assuming that $\mathbf{A}$ is non-singular and that all diagonal values a_{ii} are non-zero scalar coefficients, the Jacobi iterative method can be written as:

$$\mathbf{x}_i^{k+1} = (\mathbf{b}_i - \sum_{j \neq i} \mathbf{a}_{ij}\mathbf{x}_j^k)/\mathbf{a}_{ii} \tag{2}$$

All the unknowns $\mathbf{x}_i^{k+1}$ depend on the known terms $\mathbf{x}_i^k$, thus all the equations can be computed in parallel. This approach, however, exhibits a low rate of convergence per iteration, potentially hindering interactivity if the system is composed of many equations.

Gauss-Seidel method. The Gauss-Seidel method solves each equation sequentially based on the current state, and directly updates the result. The method is defined as follows:

$$\mathbf{x}_i^{k+1} = (\mathbf{b}_i - \sum_{j<i} \mathbf{a}_{ij}\mathbf{x}_j^{k+1} - \sum_{j>i} \mathbf{a}_{ij}\mathbf{x}_j^k)/\mathbf{a}_{ii} \tag{3}$$

In Eq. 3, the leftmost sum in the numerator considers the currently updated values $\mathbf{x}^{k+1}$, while the rightmost sum considers the values obtained at the previous iteration $\mathbf{x}^k$. This is in contrast to the Jacobi method which only computes one sum using the already known values at iteration k. While the Gauss-Seidel method has a faster rate of convergence, $\mathbf{x}_i^{k+1}$ must be computed in a sequential manner and, as such, it is not trivial to implement in a parallel fashion.

3.2. Projective Dynamics

The Projective Dynamics framework bridges the gap between Position Based Dynamics and finite element methods, providing a more accurate model for simulating soft bodies. In Projective Dynamics, a constraint is described with a potential energy. In the case of the spring constraint, Hooke's law is used. For each spring with stiffness s, connecting two particles $\mathbf{p}_i$ and $\mathbf{p}_j$ as endpoints, and rest length L_{ij}, the potential energy is expressed as:

$$E(p_i, p_j) = \frac{1}{2} s(||\mathbf{p}_i - \mathbf{p}_j|| - L_{ij})^2 \tag{4}$$

The constraints (that is, the springs) are considered satisfied when the energy is minimal, i.e. zero. By considering the set $\mathbb{U}$ of all the springs in the system, which may be identified by the pair of indices of their connecting particles i and j, Eq. 4 can be written as a minimization problem [LBOK13]:

$$E(\mathbf{p}) = \min_{d \in U} ||(\mathbf{p}_i - \mathbf{p}_j) - L_{ij}||^2 \tag{5}$$

This is an unconstrained non-linear least squares problem, which is solved by using a variant of the Alternating Direction Method of Multipliers (ADMM), an iterative local/global alternation approach [BPC*11, NOB16]. For each iteration, first each spring is solved individually, regardless the state of the connecting springs (*local* step). Then, these local configurations are used to build a linear system of equations to project the position of the particles as near as possible to the just found local solution (*global* step). By alternating between the local and the global step, the minimization problem is solved in a few iterations. In the case of mass-spring systems, the global linear system is usually rather big (the number of equations is equal to the number of springs), sparse and symmetric positive definite. While the local step can be efficiently computed in parallel, the major computational cost is finding the solution of the global linear system. In Sec. 4, we provide our method to solve such system efficiently, without requiring to explicitly store the matrix (a so-called *matrix free* solver), reducing the cost both in terms of time and memory.

Algorithm 1: Projective Dynamics - parallel Red-Black Gauss-Seidel solver

```
1  Numerical integration (Verlet)
2  loop numNonLinearIterations times
3  |  Local step
4  |  loop numLinearIterations times
5  |  |  for each P ∈ (P_R, P_B) do
6  |  |  |  for each p_i ∈ P do in parallel
7  |  |  |  |  Global step
8  |  |  |  end
9  |  |  end
10 |  end
11 end
```

4. Our Red-Black Method

4.1. Overview

The overall process to solve the minimization problem in Eq. 5 according to the Projective Dynamics method can be summarized as follows. First, two independent set of particles P_R and P_B are built as explained in Sec. 4.2. Then, the Projective Dynamics method is solved for each frame according to Alg. 1.

In step 1, the dynamics are advanced using Verlet integration (Sec. 4.3). Then, the ADMM method is applied in steps 2-7: the local steps are performed (Sec. 4.4), and the results are used to solve the linear system in the global step, where all the equations are solved in two parallel steps (Sec. 4.5). The outer loop is iterated until a feasible solution is found.

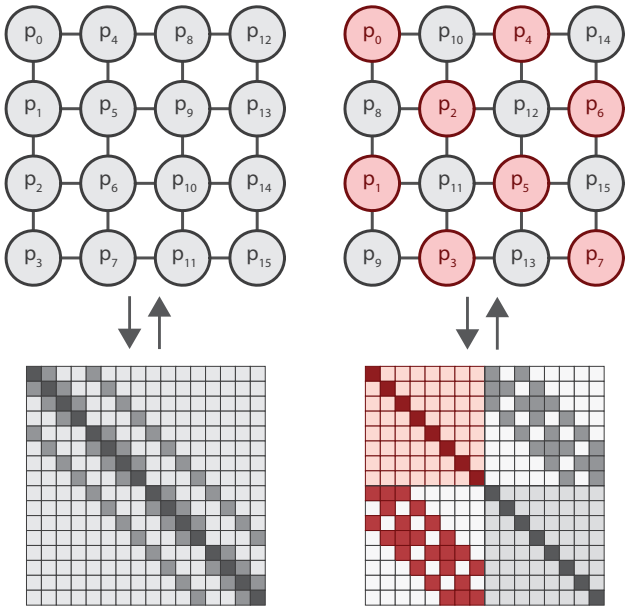

Figure 2: *The mass-spring system (top) with the corresponding coefficient matrices (bottom). Left: The input mass-spring-system. Right: the particles are re-ordered in independent sets which are solved in parallel.*

4.2. Graph coloring

The main idea of the Red-Black Gauss-Seidel solver is to partition the system of equations in two independent sets. Each set can then be solved independently of the other, which allows us to parallelize computation [Saa03].

The mass-spring system can be seen as a graph, where the nodes represent particles and the edges correspond to springs. By coloring this graph such that connected particles do not have the same color, the particles are partitioned into independent sets. Then, the dynamics of all the particles belonging to the same set is solved in parallel. Ideally, it is desirable to have a small number of similarly sized partitions to maximize parallelization and workload balance.

We use a specialized case of this technique, namely *Red-Black ordering*, where only two colors are needed. The equations in the linear system produced by the global step in Projective Dynamics are reordered to produce a result similar to the example depicted in Fig. 2. In the top row, the input mass-spring system (left) and the remapped one (right) are visible. In the bottom row the corresponding coefficient matrices are shown. Each row in the matrix corresponds to a linear equation as in Eq. 6. Starting from the input configuration of the mass-spring system, we remap the indices of the particles in the network, so that all the unknowns on the diagonal are solved in one parallel step.

4.3. Numerical integration

We use the Verlet scheme for time integration [AE15], described in Alg. 2. This method directly manipulates the particle positions, and is reasonably accurate, stable and easy to implement.

Algorithm 2: Verlet integration

1 **for each** *particle* $\mathbf{p}_i$ **do in parallel**
2 $\quad \mathbf{f}_{ext} \Leftarrow \mathbf{f}_g - \mathbf{v}_i \mathbf{f}_{damping}$
3 $\quad \mathbf{p}_i(t) \Leftarrow 2\mathbf{p}_i(t) - \mathbf{p}_i(t-h) + h^2 \mathbf{f}_{ext}$
4 **end**

For each particle $\mathbf{p}_i$, we compute the position at the next time step h. This is done in parallel. In step 2, the external forces are calculated, considering gravity $\mathbf{f}_g$, damping $\mathbf{f}_{damping}$ and the velocity $\mathbf{v}_i$. In step 3, the position is advanced by one time step.

4.4. Local step

For each spring $\mathbf{s}_{ij}$, with rest length L_{ij} and connecting particles $\mathbf{p}_i$ and $\mathbf{p}_j$, we compute $\mathbf{d}_{ij}$, which represents the local solution for $\mathbf{s}_{ij}$ minimizing Eq. 4.

Algorithm 3: Projective Dynamics, local step

1 **for each** *spring* $\mathbf{s}_{ij} \in U$ **do in parallel**
2 $\quad \mathbf{d}_{ij} = L_{ij}(\mathbf{p}_i - \mathbf{p}_j)/\|\mathbf{p}_i - \mathbf{p}_j\|$
3 **end**

4.5. Global Step

The set $\{\mathbf{d}_{ij}\}$ is used to project the position of the particles towards the local solution of the springs. This is done by linearizing Eq. 5 and assembling a linear system of equations as explained in [LBOK13, BML*14]. Since the resulting matrix $\mathbf{A}$ is big and sparse, we solve the system using a matrix-free approach to reach a solution faster and saving memory, similarly to [Wan15, FTP16]. Each equation in the linear system is assembled on the fly according to the following formula:

$$\mathbf{p}_i^{k+1} = \frac{\frac{m_i}{h^2}\mathbf{y}_i + \sum_{j<i} s_{ij}\left(\mathbf{p}_j^{k+1} + \mathbf{d}_{ij}\right) + \sum_{j>i} s_{ij}\left(\mathbf{p}_j^{k} + \mathbf{d}_{ij}\right)}{\frac{m_i}{h^2} + \sum_{j \neq i} s_{ij}} \tag{6}$$

where $\mathbf{y}_i$ is the position of the particle i at the end of the Verlet integration, $\mathbf{p}_i^k$ is the position of the particle at the current linear iteration k, m_i is the mass of the particle, $\mathbf{p}_j^k$ are all the neighboring particles, s_{ij} is the stiffness of the spring, and h is the time step.

The Red-Black ordering ensures that particles belonging to the same color do not share any constraint, thereby allowing the system to be solved in only two parallel steps. Inside each parallel step, the solving is equivalent to the lexicographic Gauss-Seidel method.

5. Experimental Results

For the sake of comparison, we implemented the Red-Black Gauss-Seidel (RBGS) and Jacobi solvers on the GPU using C++/CUDA, while the sequential Gauss-Seidel solver was implemented on the CPU using C++. All the tests were run on an NVIDIA GeForce GTX970 GPU and an Intel Xeon E5-1620 CPU.

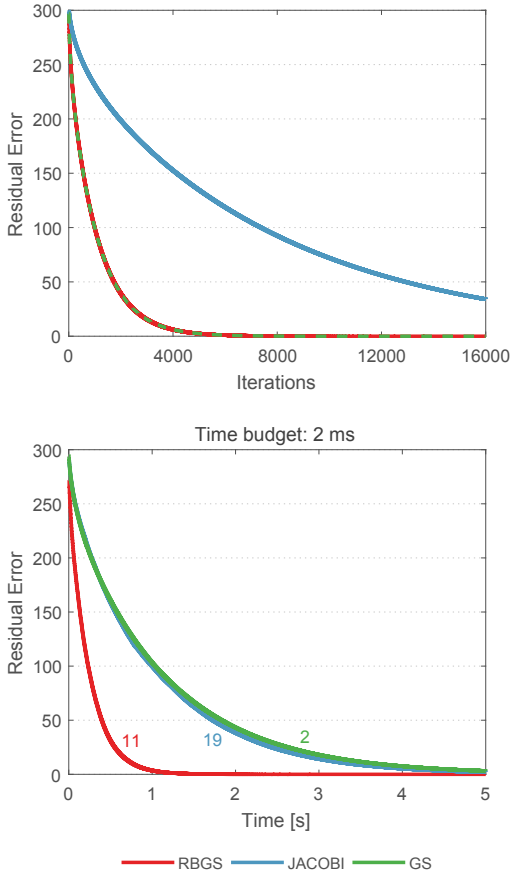

Figure 3: *Performance using a stretched network of springs shrinking to its original size. The particle system consists of 10 000 particles and 19 800 constraints. Top: Error over iterations, using 11 iterations per frame. Bottom: Error over time, using a time budget of 2 ms. The number of iterations used for each solver is reported next to the curves.*

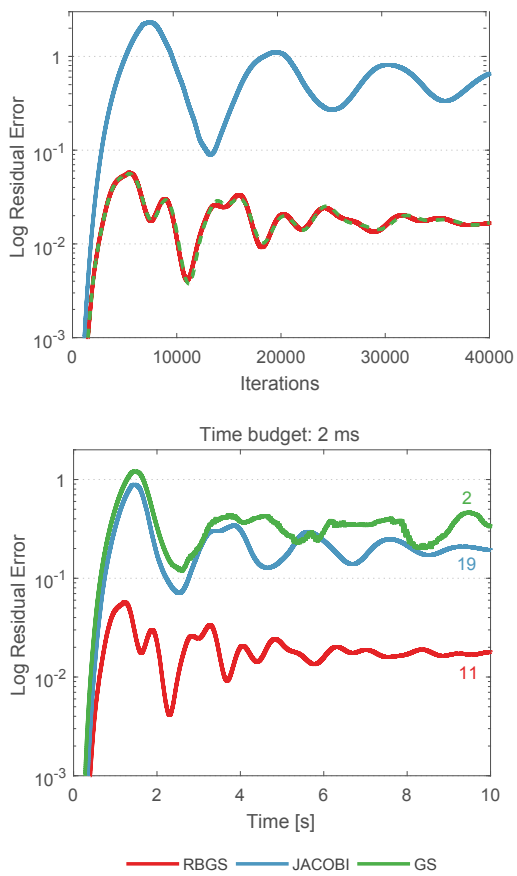

Figure 4: *Performance using a hanging cloth consisting of 10 000 particles and 19 800 constraints (Fig. 5). Top: Error over iterations, using 11 iterations per frame. Bottom: Error over time, using a time budget of 2 ms. The number of iterations used for each solver is reported next to the curves.*

5.1. Performance

We compare the RBGS solver with both parallel Jacobi and sequential Gauss-Seidel solvers using two different scenarios. In the first scenario, a network of 10 000 particles is stretched uniformly along all its dimensions, with the numerical integration disabled. During the animation, we measure how fast the particle system converges to the original size. In the second scenario, a piece of cloth is fixed by two of its corners and left free to oscillate under the gravity force. We compared the numerical results both with respect to the number of iterations per frame, and by fixing the time budget to 2 ms. Results for both scenarios are shown in Fig. 3 and Fig. 4, where the absolute error is measured as follows:

$$\sum_{N}(|\mathbf{p}_i - \mathbf{p}_j| - L_{ij})^2 \tag{7}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are particle positions and L_{ij} is the rest length of the connecting spring.

RBGS uses two parallel steps per iteration. Thus, it performs approximately half the number of iterations compared to the Jacobi solver in the same time budget. Nevertheless, it finds a more accurate solution faster, due to its higher rate of convergence per iteration. This results in a less elastic behavior as depicted in Fig. 5. The serial Gauss-Seidel solver can only achieve 2 iterations using a time budget of 2 ms leading to an unstable behavior, as shown in the accompanying video.

5.2. Test cases

In this section, we present a few challenging test cases demonstrating the flexibility of our approach. All of these test cases are presented in the accompanying video. The sequence in Fig. 6 demonstrates that collisions between multiple cloths are handled without any interpenetration. In this example, a time step of 2 ms and 8 physics updates per frame is used. The cloths consists of 16 000 particles and 32 000 constraints in total. The example runs in 35 fps. The funnel sequence in Fig. 7 shows collisions between several cloths and a funnel. The cloths consist of 30 000 particles and

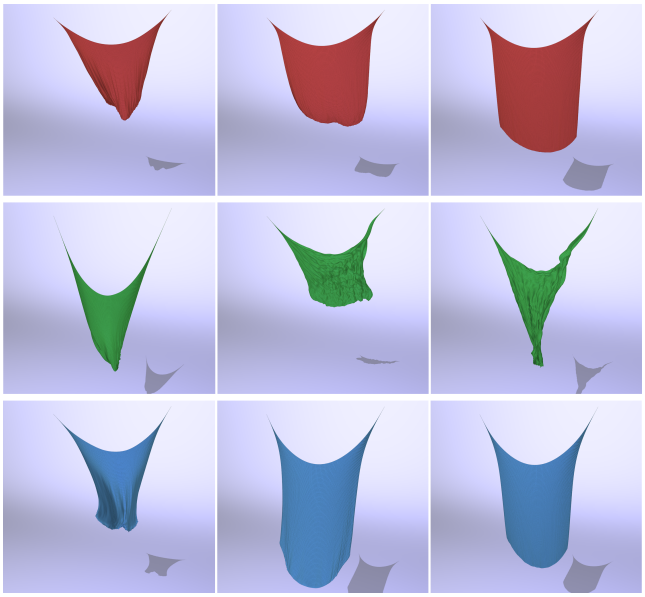

Figure 5: *Still frames from the animation corresponding to the plots in Fig. 4. Top: our solver, middle: serial Gauss-Seidel, bottom: parallel Jacobi.*

59 000 constraints in total. A time step of 2 ms is used, with 8 physics updates per frame. The frame rate for this example was 35 fps. In Fig. 8 we show how RBGS can handle efficiently changes of topology in the mass-spring system. The user first shoots a number of cannonballs at the cloth (4K particles, 8K springs), then tears it by picking the cloth and dragging with the mouse. Changes of the topology do not require that the coloring has to be recomputed. The frame rate never drops below 60 fps.

6. Limitations and future research

We presented a novel Red-Black Gauss-Seidel solver for soft body dynamics. Using a quadrangular structure of particles and constraints, our solver outperformed both standard Jacobi and Gauss-Seidel in terms of computational speed and rate of convergence. It supports topology changes such as tearing and utilizes a matrix-free approach reducing the amount of required memory. By using the approach by [JTPSH15], any triangulated mesh can be transformed into a quadrangular mesh, allowing it to be simulated using our solver.

However, the quadrangular structure is also the main limitation of our approach, since it influences the dynamics of the particle system. In fact, many constraints (such as bending), are difficult to implement. Furthermore, it is not possible to insert structural springs along the diagonals of the quadrangles, without adding more colors.

In fact, we consider this work as a first step towards a more generalized solver which expects a connectivity known *a priori*. Under this assumption, it is possible to develop a number of interesting optimizations. For example, it becomes easy to partition the input particles and constraints into independent sets without using potentially complex coloring strategies, as done in [FTP16, FP15]. Also, knowing the input connectivity, a geometric multigrid approach can be developed in a straightforward way, potentially increasing the overall efficiency of the solver. Extending the solver to support volumetric structures is also possible, and we plan to explore this research direction in the future.

References

[AE15] ABEL S., ERLEBEN K.: *Numerical methods for linear complementarity problems in physics-based animation: synthesis lectures on computer graphics and animation*. Morgan & Claypool Publishers, United States, 2015. 4

[ARF15] ABU RUMMAN N., FRATARCANGELI M.: Position-based skinning for soft articulated characters. *Comput. Graph. Forum 34*, 6 (Sept. 2015), 240–250. 1

[BBC*94] BARRETT R., BERRY M., CHAN T. F., DEMMEL J., DONATO J., DONGARRA J., EIJKHOUT V., POZO R., ROMINE C., DER VORST H. V.: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, 1994. 3

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. vol. 21, ACM, pp. 594–603. 1

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective Dynamics : Fusing Constraint Projections for Fast Simulation. *ACM Transactions on Graphics 33*, 4 (2014), 154. 2, 4

[BMM15] BENDER J., MÜLLER M., MACKLIN M.: Tutorial: Position-Based Simulation Methods in Computer Graphics. 2

[BPC*11] BOYD S., PARIKH N., CHU E., PELEATO B., ECKSTEIN J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn. 3*, 1 (Jan. 2011), 1–122. 2, 3

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. ACM, pp. 43–54. 2

[Fau99] FAURE F.: *Interactive Solid Animation Using Linearized Displacement Constraints*. Springer Vienna, Vienna, 1999, pp. 61–72. 2

[FP15] FRATARCANGELI M., PELLACINI F.: Scalable partitioning for parallel position based dynamics. *Computer Graphics Forum 34*, 2 (2015), 405–413. 2, 6

[FTP16] FRATARCANGELI M., TIBALDO V., PELLACINI F.: Vivace: A practical gauss-seidel method for stable soft body dynamics. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 214:1–214:9. 1, 2, 4, 6

[GR01] GODSIL C. D., ROYLE G.: *Algebraic graph theory*, vol. 207. Springer, New York, 2001. 2

[HT04] HABER J., TERZOPOULOS D. (Eds.): *Facial Modeling and Animation*, vol. 5 of *ACM SIGGRAPH 2004 Course Notes*. ACM SIGGRAPH, 2004. 1

[JTPSH15] JAKOB W., TARINI M., PANOZZO D., SORKINE-HORNUNG O.: Instant field-aligned meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 34*, 6 (Nov. 2015), 189:1–189:15. 6

[LBK17] LIU T., BOUAZIZ S., KAVAN L.: Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Trans. Graph. 36*, 3 (May 2017), 23:1–23:16. 2

[LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Transactions on Graphics 32*, 6 (2013). 1, 2, 3, 4

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118. 1, 2

[MM13] MACKLIN M., MÜLLER M.: Position based fluids. *ACM Trans. Graph. 32*, 4 (July 2013), 104:1–104:12. 2

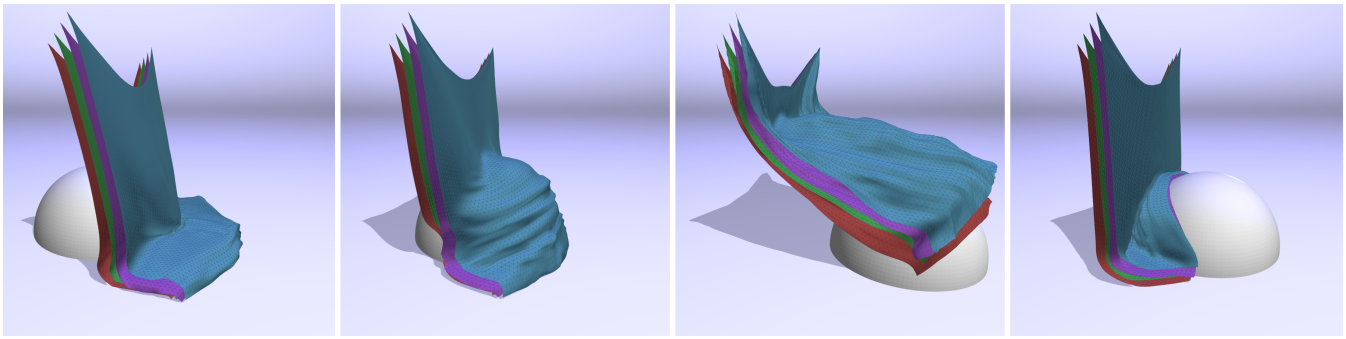

Figure 6: *Three cloths hanging from two points each, colliding with a sphere moving at high velocity.*

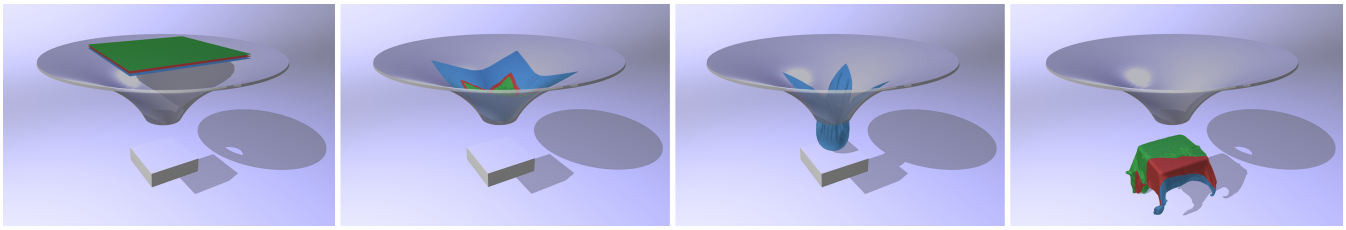

Figure 7: *Three cloths colliding with a funnel. After exiting the end of the funnel, they collide with a box.*

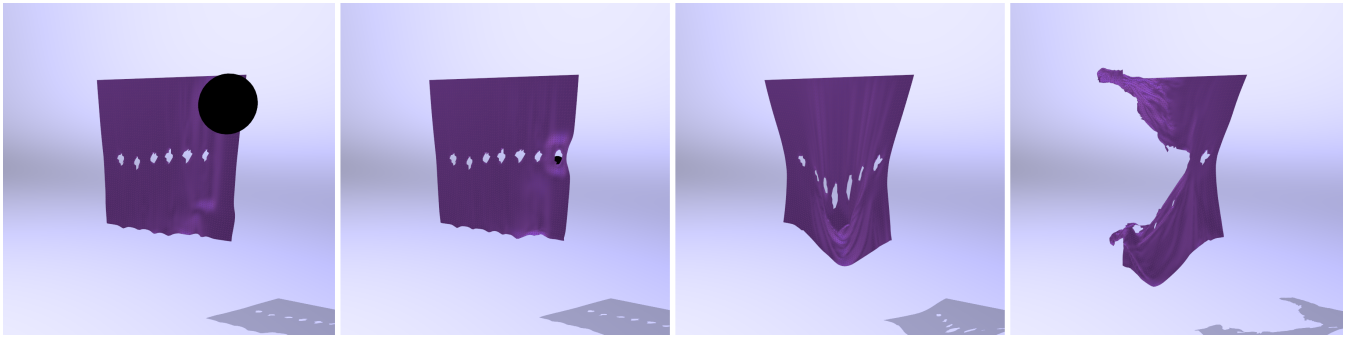

Figure 8: *Interactive change of topology. Cannonballs shot at the cloth by the user, which results in tearing holes. Then, the user picks a point on the cloth and drags it until the cloth breaks.*

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (July 2014), 153:1–153:12. 1, 2

[NOB16] NARAIN R., OVERBY M., BROWN G. E.: ADMM ⊇ projective dynamics: Fast simulation of general constitutive models. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2016), SCA '16, Eurographics Association, pp. 21–28. 3

[Pro95] PROVOT X.: Deformation Constraints in a Mass Spring Model to Describe Rigid Cloth Behavior. *Integr. Vlsi J.* (1995), 147–154. 1, 2

[RCT91] ROSENBLUM R. E., CARLSON W. E., TRIPP E.: Simulating the structure and dynamics of human hair: Modelling, rendering and animation. *The Journal of Visualization and Computer Animation 2*, 4 (1991), 141–148. 1

[Saa03] SAAD Y.: *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003. 1, 3, 4

[Sta09] STAM J.: Nucleus: Towards a unified dynamics solver for computer graphics. In *2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics* (Aug 2009), pp. 1–11. 1, 2

[TBV12] TONGE R., BENEVOLENSKI F., VOROSHILOV A.: Mass splitting for jitter-free parallel rigid body simulation. *ACM Trans. Graph. 31*, 4 (July 2012), 105:1–105:8. 2

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. *ACM SIGGRAPH Comput. Graph. 21*, 4 (1987), 205–214. 1, 2

[Wan15] WANG H.: A Chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 246:1–246:9. 1, 2, 4

[WY16] WANG H., YANG Y.: Descent methods for elastic body simulation on the gpu. *ACM Trans. Graph. 35*, 6 (Nov. 2016), 212:1–212:10. 2