Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2015)
F. Jaillet, G. Zachmann, and F. Zara (Editors)

# Interleaved Cloth Simulation

Dongsoo Han

Advanced Micro Devices

**Abstract**

*Implicit integration is a standard for stiff spring-based cloth simulation because of its stability. However constraints are useful to simulate various physical behaviors such as contact collisions or interaction with rigid bodies. Modified Conjugate Gradient (MCG) could support constraints as a part of implicit integration but constraints could not be added or removed during integration and they were limited to vertex nodes. Normally, a contact constraint has one or two frictional constraints and act inside of triangle or edge rather than vertex node. Also its inequality property makes it harder to be included in MCG. For this reason, constraints are typically applied after implicit integration as a separate step or replaced with springs. In this paper, we propose a novel method to interleave various constraints with stiff springs so that we can take advantages from both sides. Also our Jacobian-free and matrix-free implicit integration allows us to use various nonlinear forces such as pressure or none vertex-centered forces. Interleaving collision constraints into integration step can eliminate unpleasant local deformation.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

## 1. Introduction

Fabric materials have a wide range of properties such as stretch, bending and shear resistance. Some materials show great inextensibility but other materials are stretchable or elastic. Simulating these various materials is a tough task. Especially if we rely on just one simulation method such as spring or constraint, the range of materials would be limited.

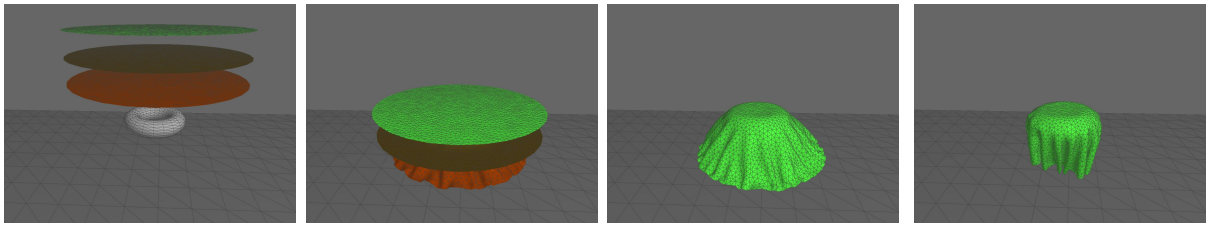
Since implicit backward integration for cloth simulation had been introduced by Baraff and Witkin [BW98], it became a standard method to simulate clothes with stiff springs. Compared to other cloth simulation approaches such as Position Based Dynamics [MHHR07] or constraint-based method [GHF*07] [HDB96], springs can represent wider physical properties especially for bending and shear forces. However springs may have an over-stretching problem, so they may not be suitable to simulate inextensible materials such as jeans. Strain-limiting as in [BMF03] [Pro96] can remedy this problem but it may destroy bending or shear effects since it usually runs after integration and does not consider other forces.

Implicit integration is a key ingredient of simulating stiff springs. It solves a linear equation which has a sparse but large matrix. This matrix should be constructed at each time and it requires to use the Jacobian matrix. Except a simple linear spring, many forces have non-linear behavior, so it may be hard to have the Jacobian. This non-linear forces can be easily observed especially during interaction with other simulation domains such as rigid body, fluid or even cloth-cloth simulation. If there is a pressure acting on the triangle of the cloth from surrounding fluid, the pressure force would be very non-linear since it is a function of triangle area as well as normal vector. If there is a spring connecting two points inside of two separate triangles, it may be hard to formulate the Jacobian of the spring since there are six vertices contributing it.

In this paper, we propose a novel approach to interleave two physics simulation components (stiff force and constraints) using Jacobian-free and matrix-free implicit integration. Our method does not require to linearize the forces or calculate the Jacobian. Also it does not construct or store the matrix. To improve the stability of the solver and allow a larger timestep, we developed the second order implicit integration and special force computation method.



DOI: 10.2312/vriphys.20151330



(a) (b) (c) (d)

Figure 1: Contact constraints with friction are interleaved with implicit integration

## 2. Related Work

Baraff and Witkin [BW98] developed implicit integration method for stiff springs and could achieve large timesteps with high stiffness. By using springs for stretch, shear and bending forces, various textile materials could be simulated but it was difficult to handle inextensible materials. To overcome this problem, Bridson et al. [BMF03] used strain-limiting as a post-processing process after the main simulation solver. Their method is similar to Position-Based Dynamics (PBD) [MHHR07] [Mül08] as solving the stretched edges one by one iteratively. Regarding stability of implicit integration, Choi et al. introduced a second-order backward difference formula (BDF) and post-bucking method.

Constraint-based approach can simulate inextensibility effectively. House et al. [HDB96] used constraint dynamics and introduced hierarchical approach for fast convergence. Goldenthal et al. [GHF*07] introduced a fast projection method which used a direct solver to apply global enforcement on inextensible materials. One of the weaknesses of constraint-based approach is that it can not control stiffness easily. In PBD [MHHR07], constraints can be softened by using stiffness value but the soft constraints can turn to be hard if the number of iterations increases. Separating bending spring from hard edge constraints was used in [GHF*07] but it could produce none smoothness artifacts due to its staggered steps.

Beside integration method, collision detection and resolution are crucial part of cloth simulation. Baraff and Witkin [BW98] used vertex constraints for cloth-vs-object collision and stiff springs for cloth-vs-cloth collision. Since those collisions are treated separately and they should be detected before the integration, the usage is limited. Also the collision type is limited to vertex and frictional constraints can not be added. In general case, contact collision is an inequality constraint. So representing contacts as springs can cause gluing effect.

In terms of interleaved method, Baraff and Witkin [BW97], described the problem of one-sided interactions between difference domains and proposed a solution which can combines various simulation domains such as rigid bodies, particles and clothes. Our method is focused to solve the same problem. However our approach is to interleave various constraints into the implicit solver. Bouaziz et al. [BML*14] developed Projective Dynamics which combines projective constraints with implicit euler solver. Even though their approach could mix them, the level of interleaving is not as deep as ours, so it requires multiple iterations where each iteration is actually not interleaved.

In cloth simulation, separating collision resolution from integration is common. Bridson el al. [BFA02] used sequential steps for integration and collision resolution. With this approach, various physics simulation components can be combined as velocity filters. However, it requires substantially small timestep to hide visual artifacts caused from staggered steps.

To our knowledge, Jacobian-free method has not been much discussed in compute graphics community even though it has been active in scientific community. Knoll et al. did a survey research on Jacobian-free method in [KK04]. In our interleaved approach, Jacobian-free method is a key feature since the interleaved constraints can be added or removed inside Conjugate Gradient (CG) iterations while we support nonlinear forces such as pressure force exerted on triangle surface. Since pressure force is a function of triangle area and its normal vector, computing the Jacobian by linearization would be non-trivial. In addition, matrix-free method can be beneficial to the simulation running on GPU because managing a large sparse matrix could be cumbersome on GPU.

## 3. Implicit Integration Review

In spring based cloth simulation, cloth is represented as vertices and springs to exert stretch, bending and shear forces to connected vertices. Each vertex has mass, position and velocity properties. Unlike explicit integration, implicit method uses a state vector at time $t + \Delta t$ as in Equation 1. Basically forces are extrapolated from $t$ to $t + \Delta t$ using their Jacobian.

In the following equation, $X$, $V$ and $F$ are vertex positions, velocities and forces respectively. $M$ is vertex masses. Subscript zero means the current time frame. $h$ is a timestep.

By substituting $\Delta X$ in the second equation in Equation 1, we get the Equation 2 which has unknown $\Delta V$. By rearranging, we get a linear equation $A\Delta\mathbf{V} = b$ as in Equation 3 where $A$ is a matrix and $b$ is a column vector.

$$\begin{pmatrix}\Delta X\\ \Delta V\end{pmatrix} = h\begin{pmatrix}V_0+\Delta V\\ M^{-1}\left[f_0+\frac{\partial f}{\partial X}\Delta X+\frac{\partial f}{\partial V}\Delta V\right]\end{pmatrix} \tag{1}$$

$$\Delta V = hM^{-1}\left[f_0+\frac{\partial f}{\partial X}h(V_0+\Delta V)+\frac{\partial f}{\partial V}\Delta V\right] \tag{2}$$

$$\left(I-hM^{-1}\frac{\partial f}{\partial V}-h^2M^{-1}\frac{\partial f}{\partial X}\right)\Delta V = hM^{-1}\left(f_0+h\frac{\partial f}{\partial X}V_0\right) \tag{3}$$

To compute the matrix $A$, we need to know the Jacobian $\frac{\partial f}{\partial X}$ and $\frac{\partial f}{\partial V}$, which often require to simplify or linearlize the forces.

To solve the linear equation, Conjugate Gradient (CG) has been a popular choice. Algorithm 1 shows Modified Conjugate Gradient (MCG) method introduced by Baraff and Witkin [BW98]. Using a filter $S$, it is possible to constrain vertices by removing their DOFs. Preconditioning is often being applied for faster convergence. We omitted the preconditioner in the Algorithm 1 to highlight the filter and compare it to the matrix-free version in the following section.

**Algorithm 1:** Modified Conjugate Gradient (MCG), S is a filter

**Input**: $\mathbf{A}$
**Input**: $\mathbf{b}$
$\mathbf{r} = S(\mathbf{b})$ ;
$\mathbf{p} = S(\mathbf{r})$ ;
$\delta_{new} = \mathbf{r}^T\mathbf{p}$;
$\delta_0 = S(\mathbf{b})^T S(\mathbf{b})$;
**while** $\delta_{new} > tolerance^2\delta_0$ **do**
  $\mathbf{s} = S(\mathbf{Ap})$;
  $\alpha = \delta_{new}/(\mathbf{p}^T\mathbf{s})$;
  $\Delta\mathbf{V} = \Delta\mathbf{V} + \alpha\mathbf{p}$;
  $\mathbf{r} = \mathbf{r} - \alpha\mathbf{s}$;
  $\delta_{old} = \delta_{new}$;
  $\delta_{new} = \mathbf{r}^T\mathbf{r}$;
  $\mathbf{p} = S(\mathbf{r} + \mathbf{p}(\delta_{new}/\delta_{old}))$;
**Output**: $\Delta\mathbf{V}$

## 4. Matrix-free Formulation

To get rid of explicit matrix representation, we need to compute the Jacobian on the fly. By using a first order finite difference, we can get the following equation.

$$\frac{\partial f_i}{\partial x_j}\delta x_j = \frac{f_i(x_j+\varepsilon\delta x_j)-f_i(x_j)}{\varepsilon} \tag{4}$$

$\varepsilon$ is a small perturbation parameter, $0 < \varepsilon \ll 1$, and subscript $i$ and $j$ are vertex indices. Since what we need to have is an extrapolated force rather than the Jacobian itself, the Equation 4 gives enough information. Here, all we need to compute is forces with the current position $x_j$ and forward position $x_j + \varepsilon\delta x_j$. Algorithm 2 explains how to compute forces. With this function, we can compute the right hand side ($b$) of Equation 3 as in Algorithm 3. $\mathbf{f_{ext}}$ is an external force such as gravity.

**Algorithm 2:** ComputeForce($\mathbf{v}$)

**Input**: $\mathbf{v}$
compute forward vertex positions $\mathbf{X}_{forward} = \mathbf{X}_0 + h\varepsilon\mathbf{v}$;
Compute forward forces $\mathbf{f}_{forward}$ exerted on vertices using the forward positions $\mathbf{X}_{forward}$;
**Output**: $\mathbf{f}_{forward}$

**Algorithm 3:** Compute $\mathbf{b}$

$\mathbf{f}_{initial}$ = ComputeForce($\mathbf{0}$);
$\mathbf{f}_{forward}$ = ComputeForce($\mathbf{V}_0$);
$\mathbf{b} = h(\mathbf{f_{ext}} + M^{-1}(\mathbf{f}_{initial} + \frac{\mathbf{f}_{forward}-\mathbf{f}_{initial}}{\varepsilon}))$
**Output**: $\mathbf{b}$

Computing the multiplication between the matrix $\mathbf{A}$ and the vector $\mathbf{p}$ is similar to how to compute $b$. It reuses $\mathbf{f}_{initial}$ which was already computed before. It uses $\mathbf{p}$ as velocity. We do not use the filtering to constrain vertices here. Instead, we developed more flexible and versatile interleaved constraint system in the next section.

**Algorithm 4:** Compute $\mathbf{s} = \mathbf{Ap}$

**Input**: $\mathbf{p}$
**Input**: $\mathbf{f}_{initial}$
$\mathbf{f}_{forward}$ = ComputeForce($\mathbf{p}$);
$\mathbf{s} = \mathbf{p} - hM^{-1}(\frac{\mathbf{f}_{forward}-\mathbf{f}_{initial}}{\varepsilon})$
**Output**: $\mathbf{s}$

With this Jacobian-free and matrix-free formulation, we can conveniently generate $b$ and replace the matrix and vector multiplication with simple force calculation. Therefore we do not need to linearize forces any more and can support various non-linear forces.

Choosing the perturbation parameter ε is delicate. If it is too large, the derivative will be poorly approximated and if it is too small, the computation will suffer from floating-point roundoff error. Knoll et al. [KK04] suggests ways to choose ε. For us, we simply use square root of machine epsilon.

## 5. Interleave Various Constraints

MCG constrains vertices by removing their DOFs using filtering. Even though it can effectively apply constraints, there are shortcomings. The constraint cannot be introduced or removed during MCG iterations. So all constraints should be known before starting MCG. Also inequality or limits cannot be applied to the constraints. It controls only positional DOFs so it is hard to constrain velocities. With these reasons, general contact constraints from cloth-vs-cloth and cloth-vs-object cannot be naturally supported by MCG.

One of the biggest benefits from our Jacobian-free and matrix-free method is that any change on vertex velocity will be immediately applied to CG solver during its iteration process. We do not need to reconstruct the linear equation and restart the CG.

Algorithm 5 shows our Jacobian and Matrix-free Conjugate Gradient (JMCG) solver with interleaved constraints. When we compute forces, we apply the filter S which is the same one in MCG so that we can support DOF based constraints. But the interleaved constrain filtering is located inside CG iteration. At each iteration, we compute forward vertex position $\mathbf{X}_{forward}$ using a full timestep and filtered updated velocity $\mathbf{V}_0 + \Delta\mathbf{V}$. Unlike MCG, our filter S can be introduced or removed during iteration and conditional according to positions or velocities so that it can support inequality or limits of constraints.

With these new positions and velocities, we can apply various constraints and velocity filtering such as contact constraints, strain-limiting or shape preserving constraints. We can also apply constraint limits here. Once we apply the constraints, we update $\mathbf{V_0}$ instead of $\Delta\mathbf{V}$. Updating $\Delta\mathbf{V}$ can cause CG to diverge. Thanks to our Jacobian-free method, updating $\mathbf{V_0}$ will be reflected to the linear system at the next iteration without affecting convergence.

Since we perform handling contact constraint within CG iteration, checking collision may become a huge bottleneck. Especially checking continuous collision detection per CG iteration would be too much expensive.

To avoid frequent bounding volume hierarchy (BVH) update and running broad phase collision detection, we define the maximum vertex displacement length similar to CFD condition in fluid simulation. When we update bounding volumes, we inflate them with this maximum length and find potential colliding pairs. After having them, we run implicit integration. Within it, when we run constrain filtering, we check the proximity-based narrow phase collisions from the pairs. To avoid having too many continuous collisions, we try to find and resolve proximity contact collisions as much as possible. At each iteration, we check each vertex's displacement and if it is larger than the maximum length condition, we update its bounding volume and its parents' bounding volumes and find a new potential colliding pairs. With this approach, we can minimize the need to update BVH and check broad phase collisions. Also we can reduce the cost of checking continuous collisions by doing more proximity checking because proximity collision is much cheaper than continuous collision in general.

**Algorithm 5:** Jacobian and Matrix-free Conjugate Gradient (JMCG), S is a filter

compute $\mathbf{b}$
  $\mathbf{f}_{initial}$ = ComputeForce($\mathbf{0}$);
  $\mathbf{f}_{forward}$ = ComputeForce(S($\mathbf{V}_0$));
  $\mathbf{b} = h(\mathbf{f_{ext}} + M^{-1}(\mathbf{f}_{initial} + \frac{\mathbf{f}_{forward} - \mathbf{f}_{initial}}{\varepsilon}))$;
$\mathbf{r} = \mathbf{b}$ ;
$\mathbf{p} = \mathbf{r}$ ;
$\delta_{new} = \mathbf{r}^T\mathbf{p}$;
$\delta_0 = \mathbf{b}^T\mathbf{b}$;
**while** $\delta_{new} > tolerance^2\delta_0$ **do**
  compute $\mathbf{Ap}$
  $\mathbf{f}_{forward}$ = ComputeForce(S($\mathbf{p}$));
  $\mathbf{s} = \mathbf{p} - hM^{-1}(\frac{\mathbf{f}_{forward} - \mathbf{f}_{initial}}{\varepsilon})$;
  $\alpha = \delta_{new}/(\mathbf{p}^T\mathbf{s})$;
  $\Delta\mathbf{V} = \Delta\mathbf{V} + \alpha\mathbf{p}$;
  $\mathbf{r} = \mathbf{r} - \alpha\mathbf{s}$;
  $\delta_{old} = \delta_{new}$;
  $\delta_{new} = \mathbf{r}^T\mathbf{r}$;
  constraint filtering
  $\mathbf{X}_{forward} = \mathbf{X}_0 + hS(\mathbf{V}_0 + \Delta\mathbf{V})$;
  apply constraints and update $\mathbf{V_0}$ ;
  $p = r + p(\delta_{new}/\delta_{old})$;
**Output**: $\Delta V$

## 6. Improve Stability

Like the original MCG introduced by Baraff and Witkin [BW98], our JMCG is not unconditionally stable, even though it allows much larger timestep compared to explicit integration. To improve the stability and have a much larger timestep, we developed second order implicit integration and special force computation method.

The following equation shows second order finite difference derived from Taylor series expansion. By using forward and backward forces, we can easily achieve second order Ja-

cobian calculation. Algorithm 6 shows second order matrix-free Conjugate Gradient method.

$$\frac{\partial f_i}{\partial x_j}\delta x_j = \frac{f_i(x_j+\varepsilon\delta x_j) - f_i(x_j-\varepsilon\delta x_j)}{\varepsilon} \tag{5}$$

**Algorithm 6:** Second order JMCG, S is a filter

compute $\mathbf{b}$
  $\mathbf{f}_{initial}$ = ComputeForce($\mathbf{0}$);
  $\mathbf{f}_{backward}$ = ComputeForce(S(-$\mathbf{V}_0$));
  $\mathbf{f}_{forward}$ = ComputeForce(S($\mathbf{V}_0$));
  $\mathbf{b} = h(\mathbf{f_{ext}} + M^{-1}(\mathbf{f}_{initial} + \frac{\mathbf{f}_{forward}-\mathbf{f}_{backward}}{\varepsilon}))$;
$\mathbf{r} = \mathbf{b}$ ;
$\mathbf{p} = \mathbf{r}$ ;
$\delta_{new} = \mathbf{r}^T\mathbf{p}$;
$\delta_0 = \mathbf{b}^T\mathbf{b}$;
**while** $\delta_{new} > tolerance^2\delta_0$ **do**
  compute $\mathbf{Ap}$
  $\mathbf{f}_{backward}$ = ComputeForce(S($-\mathbf{p}$));
  $\mathbf{f}_{forward}$ = ComputeForce(S($\mathbf{p}$));
  $\mathbf{s} = \mathbf{p} - hM^{-1}(\frac{\mathbf{f}_{forward}-\mathbf{f}_{backward}}{\varepsilon})$;
  $\alpha = \delta_{new}/(\mathbf{p}^T\mathbf{s})$;
  $\Delta\mathbf{V} = \Delta\mathbf{V} + \alpha\mathbf{p}$;
  $\mathbf{r} = \mathbf{r} - \alpha\mathbf{s}$;
  $\delta_{old} = \delta_{new}$;
  $\delta_{new} = \mathbf{r}^T\mathbf{r}$;
  constraint filtering
  $\mathbf{X}_{forward} = \mathbf{X}_0 + hS(\mathbf{V}_0 + \Delta\mathbf{V})$;
  apply constraints and update $\mathbf{V_0}$ ;
  $p = r + p(\delta_{new}/\delta_{old})$;
**Output**: $\Delta V$

In case of linear spring, we can use slightly different way to compute force when the spring is compressed. Instead using the $\mathbf{X}_{forward}$ to compute force direction vector, we use $\mathbf{X}_0$. Also we project the two forward vertex positions into the new force direction vector and use them to compute the current spring length as in Equation 6. $k_s$ is a spring constant. With this special force computation, we could observe big improvement of stability. This approach is different from post buckling method proposed by Choi et al. [CK02].

$$f_i = \begin{cases} k_s(|\mathbf{X}_{forward,ij}| - L)\frac{\mathbf{X}_{forward,ij}}{|\mathbf{X}_{forward,ij}|} & : |\mathbf{X}_{forward,ij}| \geq L \\ k_s(\mathbf{X}_{forward,ij}\cdot\frac{\mathbf{X}_{0,ij}}{|\mathbf{X}_{0,ij}|} - L)\frac{\mathbf{X}_{0,ij}}{|\mathbf{X}_{0,ij}|} & : |\mathbf{X}_{forward,ij}| < L \end{cases} \tag{6}$$

## 7. Results

Firgure 1, 3 and 4 show cloth simulations with pin or contact constraints interleaved within JMCG. The pinned vertex is constrained on the horizon line. Therefore the vertex can only move following the invisible line.

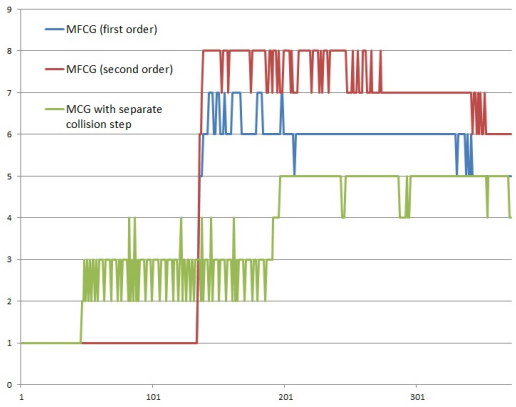



Figure 2: Comparing CG iterations from JMCG (first order), JMCG (second order) and MCG with separate collision step. The horizon axis is simulation frame and the vertical axis is the number of CG iterations. The simulation scene is in Figure 1. In all three cases, stretch spring constant was 100000 and bending spring constant was 10000. Timestep was 240 Hz without any sub-steps.

Figure 2 shows the comparison among first order JMCG, second order JMCG and MCG with separate collision step. The simulation scene is shown in Figure 1. The second order JMCG has slightly higher CG iterations than the first order JMCG. MCG shows low CG iterations but it is probably because the collision is separate from integration process.

We do not compare the simulation time between JMCG and MCG because interleaved contact constraint can contribute a lot on the performance. However, based on our observation, JMCG is comparable to MCG as long as the CG iteration is not too high. It may possible to run constraint filtering in JMCG at every few iterations but we believe it may cause CG to converge slowly.

## 8. Conclusion

Matrix-free approach can be beneficial in terms of memory consumption and sparsity management. GPU would be a big beneficiary. If cloth mesh topology changes frequently such as tearing, our matrix-free method can handle it efficiently since there will be no reconstructing or managing matrix explicitly.

The Jacobian-free method allows us to use non-linear forces without linearizing them. Previously two-way interaction with other physical material such as fluid often used penalty force or impulse in sequential manner. With our ap-

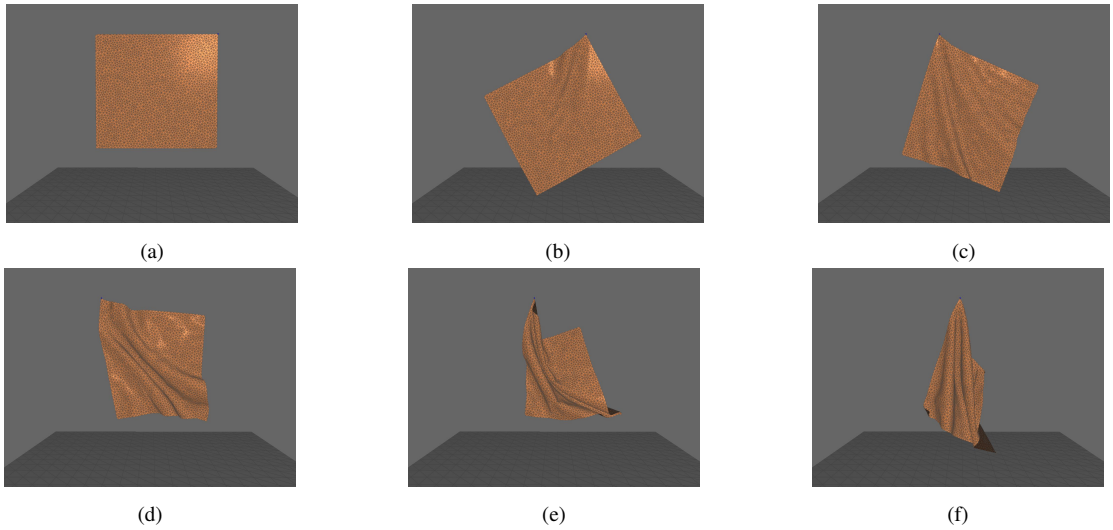


Figure 3: Cloth is constrained at the corner on the horizontal line.

proach, it should be possible to interleave it within integration solver. We plan to do more research on this topic.

Our implicit integration can support various interleaved constraint types such as contact collision, pins or strain-limiting inside its CG iteration. This interleaving can give a lot of flexibility to simulate various physical materials.

For future research, we want to improve the stability and develop a smart way to tune ε in a better way. Also we plan to investigate preconditioner for matrix-free CG.

## References


[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 594–603. URL: http://doi.acm.org/10.1145/566570.566623, doi:10.1145/566570.566623. 2

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 28–36. URL: http://dl.acm.org/citation.cfm?id=846276.846281. 1, 2

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (July 2014), 154:1–154:11. URL: http://doi.acm.org/10.1145/2601097.2601116, doi:10.1145/2601097.2601116. 2

[BW97] BARAFF D., WITKIN A.: Partitioned dynamics, 1997. 2

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 43–54. URL: http://doi.acm.org/10.1145/280814.280821, doi:10.1145/280814.280821. 1, 2, 3, 4

[CK02] CHOI K.-J., KO H.-S.: Stable but responsive cloth. *ACM Trans. Graph. 21*, 3 (July 2002), 604–611. URL: http://doi.acm.org/10.1145/566654.566624, doi:10.1145/566654.566624. 5

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient simulation of inextensible cloth. *ACM Trans. Graph. 26*, 3 (July 2007). URL: http://doi.acm.org/10.1145/1276377.1276438, doi:10.1145/1276377.1276438. 1, 2

[HDB96] HOUSE D., DEVAUL R. W., BREEN D. E.: Towards

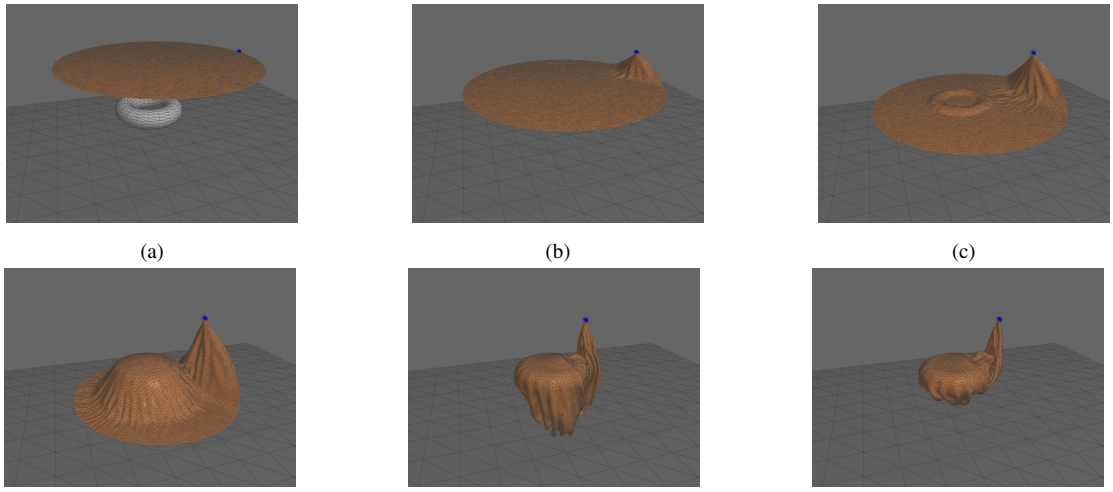


Figure 4: Cloth is pinned on the horizontal line and there is contact collision with torus shaped object. Both pin and contact constraints are interleaved with matrix-free implicit integration.

simulating cloth dynamics using interacting particles. *International Journal of Clothing Science and Technology 8* (1996), 75–94. 1, 2

[KK04] KNOLL D. A., KEYES D. E.: Jacobian-free newton-krylov methods: A survey of approaches and applications. *J. Comput. Phys. 193*, 2 (Jan. 2004), 357–397. URL: http://dx.doi.org/10.1016/j.jcp.2003.08.010, doi:10.1016/j.jcp.2003.08.010. 2, 4

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118. URL: http://dx.doi.org/10.1016/j.jvcir.2007.01.005, doi:10.1016/j.jvcir.2007.01.005. 1, 2

[Mül08] MÜLLER M.: Hierarchical position based dynamics. In *VRIPHYS* (2008), pp. 1–10. 2

[Pro96] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *In Graphics Interface* (1996), pp. 147–154. 1