XPBD

Paper Review: Summary & Implementations

SNU ECE 23
Sanghwa Lee

Macklin et al. (2016).

Paper review

XPBD: Position-Based Simulation of Compliant Constrained Dynamics

Miles Macklin Matthias Müller Nuttapong Chentanez

NVIDIA

Figure 1: *In this example, we see the effect of changing the relative stiffness of volume conservation and stretch and shear constraints on a deformable body. Unlike traditional PBD, our method allows users to control the stiffness of deformable bodies in a time step and iteration count independent manner, greatly simplifying asset creation.*

Abstract

We address the long-standing problem of iteration count and time step dependent constraint stiffness in position-based dynamics (PBD). We introduce a simple extension to PBD that allows it to accurately and efficiently simulate arbitrary elastic and dissipative energy potentials in an implicit manner. In addition, our method provides constraint force estimates, making it applicable to a wider range of applications, such as those requiring haptic user-feedback. We compare our algorithm to more expensive non-linear solvers and find it produces visually similar results while maintaining the simplicity and robustness of the PBD method.

Keywords: physics simulation, constrained dynamics, position based dynamics

Concepts: •**Computing methodologies** → **Real-time simulation;** *Interactive simulation;*

1 Introduction

Position-Based Dynamics [Müller et al. 2007] is a popular method for the real-time simulation of deformable bodies in games and interactive applications. The method is particularly attractive for its simplicity and robustness, and has recently found popularity outside of games, in film and medical simulation applications.

As its popularity has increased, the limitations of PBD have become more problematic. One well known limitation is that PBD's behavior is dependent on the time step and iteration count of the simulation [Bender et al. 2014b]. Specifically, constraints become arbitrarily stiff as the iteration count increases, or as the time step decreases. This coupling of parameters is particularly problematic when creating scenes with a variety of material types, e.g.: soft bodies interacting with nearly rigid bodies. In this scenario, raising iteration counts to obtain stiffness on one object may inadvertently change the behavior of all other objects in the simulation. This often requires stiffness coefficients to be re-tuned globally, making the creation of reusable simulation assets extremely difficult. Iteration count dependence is also a problem even in the case of a single asset, for example, setting the relative stiffness of stretch and bending constraints in a cloth model. To make matters worse, the effects of iteration count are non-linear, making it difficult to intuitively adjust parameters, or to simply rescale stiffness values as a simple function of iteration count.

The recent resurgence in virtual-reality has given rise to the need for higher fidelity and more physically representative real-time simulations. At the same time, the wide-spread use of haptic feedback devices require methods than can provide accurate force estimates. PBD does not have a well defined concept of constraint force, and as such it has mostly been limited to applications where accuracy is less important than speed, and where simulations are secondary effects.

In this paper we present our extended position-based dynamics (XPBD) algorithm. Our method addresses the problems of iteration and time step dependent stiffness by introducing a new constraint formulation that corresponds to a well-defined concept of elastic potential energy. We derive our method from an implicit time discretization that introduces the concept of a total Lagrange multiplier to PBD. This provides constraint force estimates that can be used to drive force dependent effects and devices.

To summarize, our main contributions are:

- Extending PBD constraints to have a direct correspondence to well-defined elastic and dissipation energy potentials.
- Introducing the concept of a total Lagrange multiplier to PBD allowing us to solve constraints in a time step and iteration count independent manner.
- Validation of our algorithm against a reference implicit time stepping scheme based on a non-linear Newton solver.

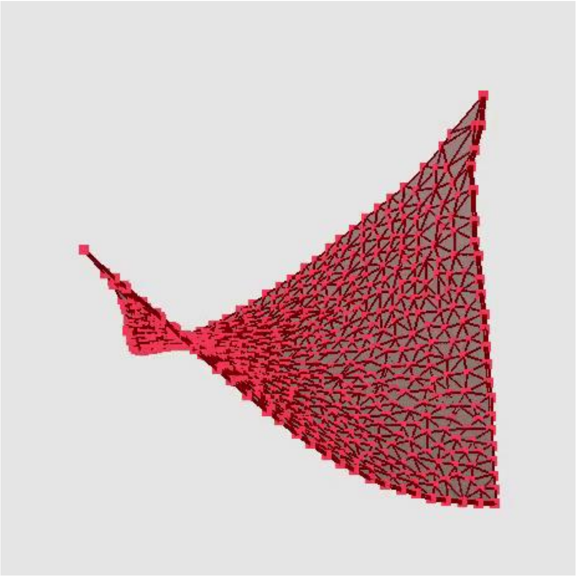
MiG '16,, October 10 - 12, 2016, Burlingame, CA, USA
ISBN: 978-1-4503-4592-7/16/10
DOI: http://dx.doi.org/10.1145/2994258.2994272

Macklin, Miles, et al. "XPBD." Proceedings of the 9th International Conference on Motion in Games, 10 Oct. 2016, https://doi.org/10.1145/2994258.2994272.

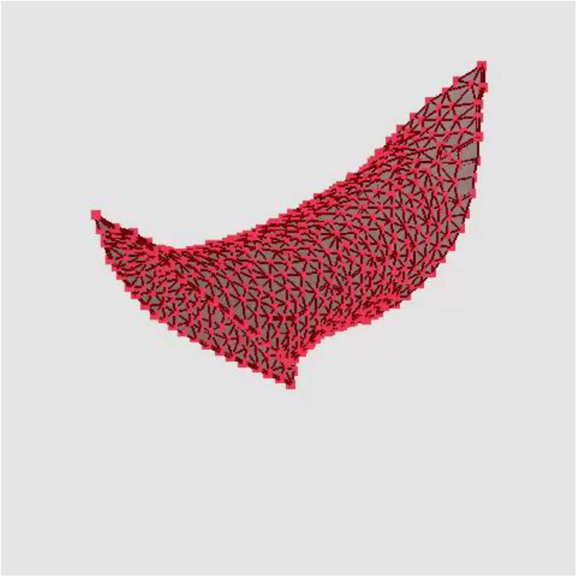
Macklin et al. (2016).

Issues of PBD

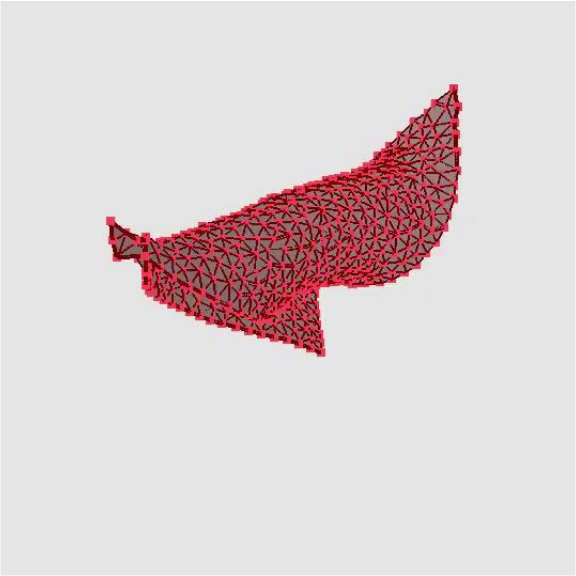
More stiff *Less stiff*



$n_s = 20$



$n_s = 10$

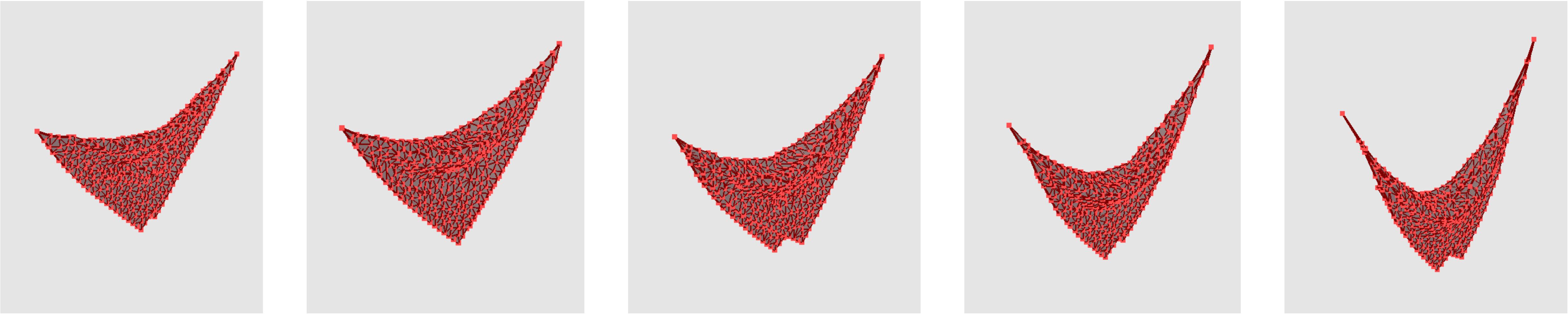


$n_s = 5$

Macklin et al. (2016).

Issues of PBD

(Only stretching considered)



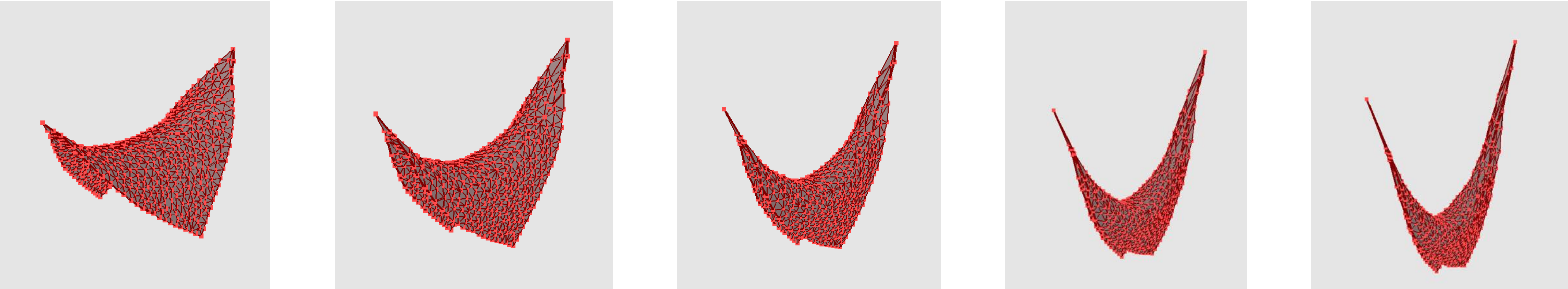
$n_s = 40$ $n_s = 30$ $n_s = 20$ $n_s = 10$ $n_s = 5$

The number of iteration n_s

$(\Delta t = 0.01)$

Macklin et al. (2016).

Issues of PBD



$\Delta t = 0.01$ | $\Delta t = 0.015$ | $\Delta t = 0.02$ | $\Delta t = 0.025$ | $\Delta t = 0.03$

Time step Δt

$(n_s = 20)$

Macklin et al. (2016).

XPBD

eXtended ***P**osition* ***B**ased* ***D**ynamics*

Issues of PBD

Stiffness depends on
- *The number of iteration* n_s
- *Time step* Δt

Macklin et al. (2016).

Theory

Macklin et al. (2016).

Algorithm

Original PBD

predict positions

$\tilde{x} \leftarrow x + \Delta t v + \Delta t^2 w f_{ext}(x)$

for all

project constraints C_i

end for

update positions

$x \leftarrow \tilde{x}$

→

XPBD

predict positions

$\tilde{x} \leftarrow x + \Delta t v + \Delta t^2 w f_{ext}(x)$

for all

generate constraints C_i

compute $\Delta x_i, \Delta \lambda_i$

update x_i, λ_i

end for

update positions

$x \leftarrow \tilde{x}$

Macklin et al. (2016).

Theory

$$\boldsymbol{M}\ddot{\boldsymbol{x}} = -\nabla U^T(\boldsymbol{x})$$

Iteration step n

Constraint function $\boldsymbol{C}$

Inverse stiffness $\boldsymbol{\alpha}$

$$\boldsymbol{M}\left(\frac{\boldsymbol{x}^{n+1} - 2\boldsymbol{x}^{n} + \boldsymbol{x}^{n-1}}{\Delta t^2}\right)$$

$$U(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{C}(\boldsymbol{x})^T\boldsymbol{\alpha}^{-1}\boldsymbol{C}(\boldsymbol{x})$$

$+$ *Lagrange multiplier*

$$\boldsymbol{\lambda} = -\tilde{\boldsymbol{\alpha}}^{-1}\boldsymbol{C}(\boldsymbol{x})$$

Macklin et al. (2016).

Theory

Simplifying equations,

$$\begin{cases} \boldsymbol{M}(\boldsymbol{x}^{n+1} - \widetilde{\boldsymbol{x}}) - \nabla \boldsymbol{C}(\boldsymbol{x}^{n+1})^T \boldsymbol{\lambda}^{n+1} = \mathbf{0} \\ \boldsymbol{C}(\boldsymbol{x}^{n+1}) + \widetilde{\boldsymbol{\alpha}} \boldsymbol{\lambda}^{n+1} = \mathbf{0} \end{cases}$$

Solve by iterations

$$\begin{bmatrix} \boldsymbol{K} & -\nabla \boldsymbol{C}^T(\boldsymbol{x}_i) \\ \nabla \boldsymbol{C}(\boldsymbol{x}_i) & \widetilde{\boldsymbol{\alpha}} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\lambda}_i) \\ \boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{\lambda}_i) \end{bmatrix} \longrightarrow \begin{aligned} \boldsymbol{\lambda}_{i+1} &= \boldsymbol{\lambda}_i + \Delta \boldsymbol{\lambda} \\ \boldsymbol{x}_{i+1} &= \boldsymbol{x}_i + \Delta \boldsymbol{x} \end{aligned}$$

Macklin et al. (2016).

Approximations

Assumption 1. $\boldsymbol{K} \approx \boldsymbol{M}$

Local error of $O(\Delta t^2)$ - Does not change the solution

Assumption 2. $\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{\lambda}_i) = \mathbf{0}$

True for the first iteration $\boldsymbol{x}_0 = \tilde{\boldsymbol{x}}, \boldsymbol{\lambda}_0 = \mathbf{0}$

$$\downarrow$$

$$\begin{bmatrix} \boldsymbol{M} & -\nabla \boldsymbol{C}^T(\boldsymbol{x}_i) \\ \nabla \boldsymbol{C}(\boldsymbol{x}_i) & \tilde{\boldsymbol{\alpha}} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{\lambda}_i) \end{bmatrix}$$

Macklin et al. (2016).

Final equations

$$\begin{cases} \Delta\boldsymbol{\lambda} = \dfrac{-\boldsymbol{C}(\boldsymbol{x}_i) - \tilde{\boldsymbol{\alpha}}\boldsymbol{\lambda}_i}{\nabla\boldsymbol{C}\boldsymbol{M}^{-1}\nabla\boldsymbol{C}^T + \tilde{\boldsymbol{\alpha}}} \\ \Delta\boldsymbol{x} = \boldsymbol{M}^{-1}\nabla\boldsymbol{C}(\boldsymbol{x}_i)^T\Delta\boldsymbol{\lambda} \end{cases}$$

For all constraints $\boldsymbol{C}$,

1. *Calculate $\boldsymbol{C}$ and $\nabla\boldsymbol{C}$*
2. *Find $\Delta\boldsymbol{\lambda}$*
3. $\boldsymbol{\lambda}_i \mathrel{+}= \Delta\boldsymbol{\lambda}$
4. $\boldsymbol{x}_i \mathrel{+}= \Delta\boldsymbol{x}$

Implement

Macklin et al. (2016).

Implementation

Macklin et al. (2016).

Implementation

```
void ClothSimulator::updatePBD() {
    explicitEuler();

    double itr = Parameter::getInstance().getIteration();
    for (int n = 0; n < itr; n++) {
        calDistanceConstraint();
        //calBendingConstraint();
    }

    integrationForPBD();
}
```

Improved with XPBD

TBA

Macklin et al. (2016).

Implementation

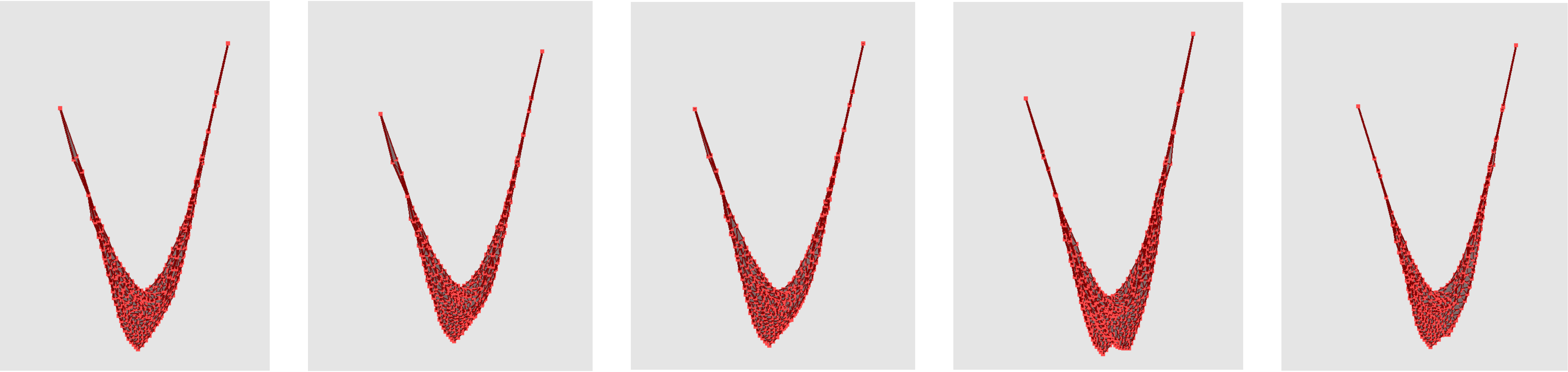
```
void ClothSimulator::calDistanceConstraint() {
double dt = Parameter::getInstance().getTimestep();
double k = Parameter::getInstance().getStretchStiffness();

for (int i = 0; i < nE; i++) {
int idx1 = edges[i][0];
int idx2 = edges[i][1];

Vector3d p1 = pos_itr[idx1];
Vector3d p2 = pos_itr[idx2];
Vector3d n = (p1 - p2) / (p1 - p2).norm();

double C = (p1 - p2).norm() - restLengthsOfBending[i];

Vector3d gradC_p1 = n;
Vector3d gradC_p2 = - n;

double w1 = inv_mass[idx1];
double w2 = inv_mass[idx2];

double alpha = 1 / (k * dt * dt);
double dlambda = (-C - alpha * lambda_itr[i]) / (w1 * gradC_p1.norm() * gradC_p1.norm() + w2 * gradC_p2.norm() * gradC_p2.norm() + alpha);
lambda_itr[i] += dlambda;

Vector3d dp1 = w1 * gradC_p1 * dlambda;
Vector3d dp2 = w2 * gradC_p2 * dlambda;

pos_itr[idx1] += dp1;
pos_itr[idx2] += dp2;
}
}
```

1. *Calculate* $\boldsymbol{C}$ *and* $\nabla\boldsymbol{C}$
2. *Find* $\Delta\boldsymbol{\lambda}$
3. $\boldsymbol{\lambda}_i \mathrel{+}= \Delta\boldsymbol{\lambda}$
4. $\boldsymbol{x}_i \mathrel{+}= \Delta\boldsymbol{x}$

Macklin et al. (2016).

Results – iteration

(Only stretching considered)



$n_s = 40$ $n_s = 30$ $n_s = 20$ $n_s = 10$ $n_s = 5$

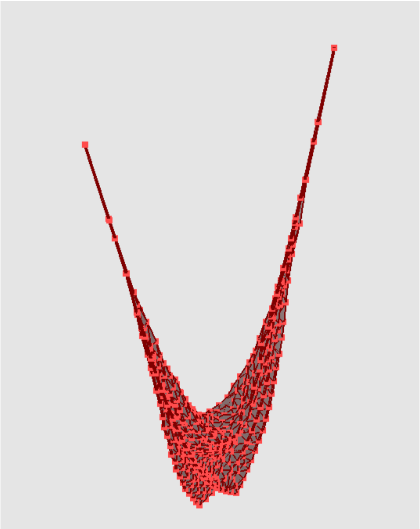
Does not depend on n_s

$(\Delta t = 0.01, k = 0.1)$

Macklin et al. (2016).

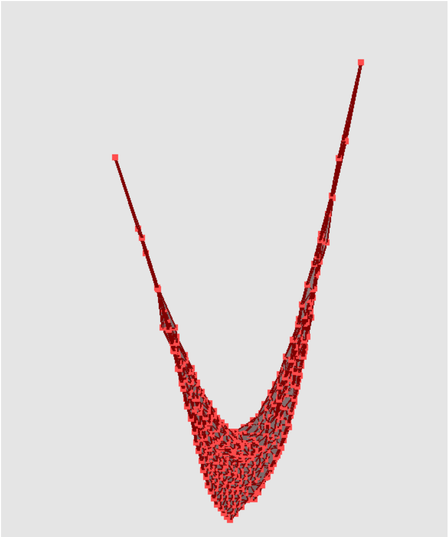
Results – time step

(Only stretching considered)



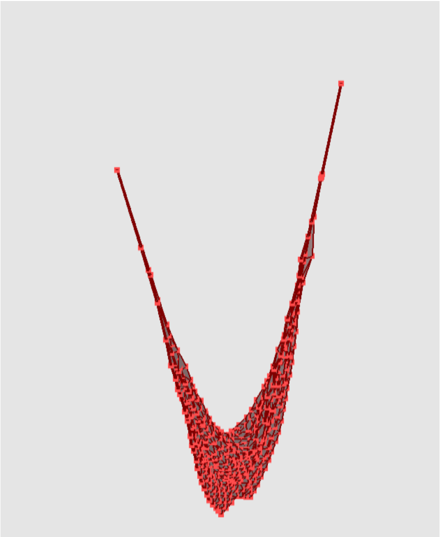
$\Delta t = 0.005$

$\Delta t = 0.01$



$\Delta t = 0.015$

$\Delta t = 0.02$



$\Delta t = 0.025$

Does not depend on Δt

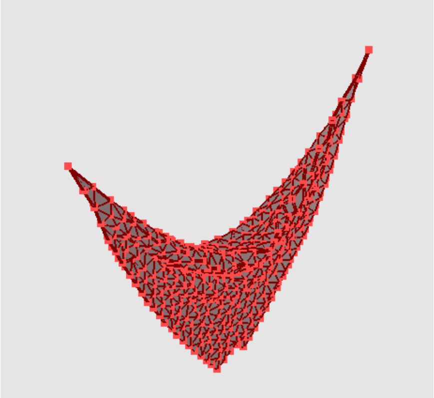
$(n_s = 20, k = 0.1)$

Macklin et al. (2016).

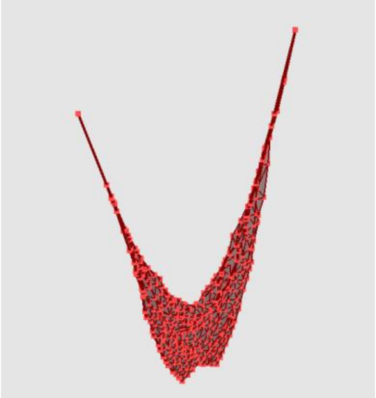
Limitations?

Acted similar to PBD when $k \approx 1$

> To clearly show how the PBD behavior changes with varying iteration counts we have used an artificially low constraint stiffness of $k = 0.01$. We note that XPBD does not converge faster than PBD, simply that behavior remains consistent at different iteration counts. Indeed, in the limit of zero compliance XPBD is equivalent to a constraint stiffness of $k = 1$ in PBD.



$\Delta t = 0.01$



$\Delta t = 0.03$

$(n_s = 20, k = 0.9)$

Macklin et al. (2016).

Conclusion

Lagrange multiplier

Theory

$$\begin{cases} \Delta\boldsymbol{\lambda} = \dfrac{-\boldsymbol{C}(\boldsymbol{x}_i) - \widetilde{\boldsymbol{\alpha}}\boldsymbol{\lambda}_i}{\nabla\boldsymbol{C}\boldsymbol{M}^{-1}\nabla\boldsymbol{C}^T + \widetilde{\boldsymbol{\alpha}}} \\ \Delta\boldsymbol{x} = \boldsymbol{M}^{-1}\nabla\boldsymbol{C}(\boldsymbol{x}_i)^T\Delta\boldsymbol{\lambda} \end{cases}$$

Implementation

1. *Calculate* $\boldsymbol{C}$ *and* $\nabla\boldsymbol{C}$
2. *Find* $\Delta\boldsymbol{\lambda}$
3. $\boldsymbol{\lambda}_i \mathrel{+}= \Delta\boldsymbol{\lambda}$
4. $\boldsymbol{x}_i \mathrel{+}= \Delta\boldsymbol{x}$

$\longrightarrow$

Issues of PBD solved

1. *Does not depend on* n_s
2. *Does not depend on* Δt

Macklin et al. (2016).