An Extended Position Based Dynamics Method for Cloth Simulation

Wei Quan¹, Dexin Teng¹*, Hua Li¹, Cheng Han¹
¹School of Computer Science and Technology, Changchun University of Science and Technology, Jilin, Changchun, 130000, China
*Corresponding author’s e-mail: 229839695@qq.com

Abstract. Position based dynamics is an effective method in real time deformable body simulation, which is widely used. But, as we all know, it is not suitable for greater accuracy applications. Therefore, an extended simulation method based on position dynamics is proposed in this paper. The time step is subdivided effectively, and the number of iterations is controlled when cloth is deforming; We also optimize the Gauss-Seidel iterative solver in parallel using an improved grouping method which ensures the efficiency of the algorithm. This method is simple and easy to implement. Compared with the traditional method, the position error of deformation is significantly reduced while cloth is stretched, and the simulation efficiency is enhanced. Besides, it can be easily integrated into the existing position based dynamic framework.

1. Introduction
In the past decades, cloth simulation has been an active research field in computer graphics, which is closely related to people's lives for it has been highly concerned in the film, game and other industries. Position based Dynamics (PBD)[1] is the most popular method for real-time deformation simulation in games and interactive applications. Its best advantage is fast simulation speed and unconditional stability. But there are also significant limitations, for example, the stiffness of the material depends on the time step and the number of iterations of the simulation. Specifically, as the number of iterations increases or the time step decreases, the cloth becomes more rigid, and the simulation time becomes longer, which not only can't satisfy the high efficiency, but also can't show the lifelike fold.

To solve this problem, Muller et al.[2] released an extended version of PBD called XPBD. It solves the problem that stiffness depends on the number of iterations and it also supports material parameters that are physically meaningful. In general, the best value of the time step is determined by experiments, and the change of iteration times will not affects the simulation result of the cloth, so the simulation speed can be accelerated by reducing the iteration number. However, there are still errors in the simulation results, and the predominant one is the excessive stretching on cloth.

Therefore, a XPBD-based method is proposed in this paper, which simultaneously modifies the time step and the number of iterations. Starting from the time step, the position error is reduced by shortening the time step. Then the number of iterations in each time step is reduced. The formula for calculating the Lagrange increment is improved, and the amount of calculation is reduced. The optimal time step and number of iterations are determined through experiments, which not only solves the stretch problem, but also simplifies the calculation complexity in the solution process. In addition, by optimizing the Gauss-Seidel iterative process for GPU computing architecture, we further accelerate the simulation speed of cloth so as to improve the low efficiency in XPBD.
2. Related work
Most cloths do not stretch under their own weight, but different solving processes can cause different degrees of stretching during the simulation. The most common method for the stretch resistance of cloth is to treat the cloth as an elastic material. In order to reduce visible tension, elastic models usually use large elastic modulus or rigid springs, but this will reduce numerical stability\cite{3}. The implicit integration method proposed by Baffer et al.\cite{4} can reduce the rigid component of the system to a certain extent, and improve the over-stretching problem through the adaptive time step integration, but it requires a lot of calculation, and the simulation speed is slow. Not only that the real-time performance becomes poor. Liu et al.\cite{5} used a local-global approach to solve the deformation of the spring particle model. This method deals with the non-linear term locally, and decompose the tensile transformation by the global method, which can obtain accurate Newton solutions in a short time. It is suitable for real-time applications, but lacking of deformative details when the collision force is large.

Another effective method to reduce the rigid component and maintain the stability of the system is constraint dynamics, which defines the rigid components as constraints and adds them to the dynamic system by means of binding force or Lagrange multiplier. The displacement is iteratively corrected and the stretch deformation of the cloth can be effectively controlled. Harmon et al.\cite{6} defined the stretching interaction between cloth as constraints, and solved the dynamic equation of the stretching constraint by a projection iterative method. Hong et al.\cite{7} defined the implicit constraint dynamics equation, in which the force to be added in current movement is estimated based on the state of motion at the next moment. This method can simulate the animation effect of simple cloth, but the amount of calculation was too large. Muller et al.\cite{8} defined many kinds of constraints based on the position of the particles, and used a non-linear Gauss-Seidel method to solve the constraints one by one to achieve a simple cloth simulation. However, this method cannot effectively control the stretch of the cloth, and it is difficult to reflect the details of deformation during the cloth movement. Bouaziz et al.\cite{9} proposed a new implicit time integration method, and it derived a broad constraint which balanced the accuracy and simplicity of the simulation. Based on the framework of position dynamics, Macklin et al.\cite{10} created a unified solver that can simulate gas, liquid, deformable solid, rigid body and cloth, so that PBD has a more complete system. Macklin et al. proposed the XPBD method, which improves the number of iterations and the time step constraint stiffness problem in PBD, so that it can accurately and effectively simulate any elastic and dissipative potential energy implicitly. It is suitable for a wider range of applications, but there are still some problems, such as low accuracy and insufficient speed.

3. Background

3.1. Cloth modeling
In the position-based dynamics simulation, all the objects to be simulated are represented by a set of particles, and each particle is a sphere with a given radius. The cloth model in PBD should be a mesh model in any triangulation form, with the only requirement that each edge of the triangle must be shared by at least two other triangles. Then the triangle mesh model of the cloth is converted into a particle model. Each mesh vertex corresponds to a cloth particle, which is used for the next calculation.

3.2. Adding constraints
The mainly bottom constraint of cloth simulation is distance constraint, including stretching constraint and bending constraint. Because the constraint equation of the bending constraint has nothing to do with the side length of the triangle, the two kinds of constraints are independent of each other. So they can be adjusted separately to simulate the cloth of different materials.

3.2.1. Stretching constraint
For the stretching of the cloth, construct stretching constraints on each side of the mesh model. When the distance between the particles of the cloth model is not equal to their original distance, we need to adjust the distance between the particles to meet the original distance constraint (Figure 1).
For two particles connected by any edge, the stretching constraint equation between them is defined as:

\[ C_{\text{stretch}}(P_1, P_2) = |P_1 - P_2| - l_0 \]  

(1)

Where \( l_0 \) is the original length of the edge. According to the constrained projection method, the gradient of the constraint function is calculated at points \( p_1 \) and \( p_2 \), respectively, to obtain the corrected displacement:

\[
\Delta P_1 = -\frac{w_1}{w_1 + w_2} (|P_1 - P_2| - d) \frac{P_1 - P_2}{|P_1 - P_2|} \\
\Delta P_2 = +\frac{w_2}{w_1 + w_2} (|P_1 - P_2| - d) \frac{P_1 - P_2}{|P_1 - P_2|} 
\]

(2)  

(3)

3.2.2. Bending constraint

In the triangular mesh model, the essence of the bending of adjacent triangles is the angle formed by the two normals of the two adjacent triangles, which is the dihedral angle, as shown in Figure 2(a). Therefore, the constraint of the value of the dihedral angle can be defined as the bending constraint.

The bending constraint equation of a pair of adjacent triangles \( T_1(P_1, P_2, P_3) \) and \( T_2(P_1, P_2, P_4) \) is defined as:

\[
C_{\text{bend}}(P_1, P_2, P_3, P_4) = \cos\left(\frac{(p_2 - p_1) \times (p_3 - p_1) \cdot (p_2 - p_1) \times (p_4 - p_1)}{|(p_2 - p_1) \times (p_3 - p_1)| \cdot |(p_2 - p_1) \times (p_4 - p_1)|}\right) - \varphi_0 
\]

(4)

Where \( \varphi_0 \) is the initial dihedral angle between the two triangles, \( n_1 \) and \( n_2 \) are the normal vectors of adjacent triangles. Project \( n_1 \), \( n_2 \), and \( P_3 \), \( P_4 \) onto a two-dimensional plane perpendicular to the
coincidence edge, as shown in the Figure 2(b). By solving the constraints, the angle $\phi$ between $n_1$ and $n_2$ is approximated to $\phi_0$, and the final correction satisfying the constraints is obtained:

$$
\Delta p_i = -\frac{4w_i}{\Sigma_j w_j} \sqrt{1-d^2} (\arccos(d) - \phi_0) q_i
$$

(5)

Where $w_i = \frac{1}{m_i}$ and $q$ is obtained by taking the gradient of the constraint function at each point and normalizing it.

### 3.3. The solver

There are two commonly used solvers in constraint dynamics methods: Jacobi solver and Gauss-Seidel solver. Jacobi solver is a parallel solver. It can use the GPU parallel framework to obtain good acceleration results, but cannot guarantee convergence. The Gauss-Seidel serial solver handles constraints one by one and has good convergence, but the convergence speed is far less than that of the parallel solver. So many people started to study the parallelization of Gauss-Seidel solver. In the process of Gauss-Seidel iterative solution, a single particle may be limited by multiple constraints, and there are dependencies between constraints. Besides, interdependent constraints cannot be calculated in parallel. The starting position of the next constraint is based on the calculation results of the previous constraint. As a result, the next constraint can only be executed after finishing the previous constraint calculation and updating the particle position. It causes the convergence speed to be severely reduced and is difficult to be parallelized.

### 4. The Extended Position Based Dynamics Simulation Method for Cloth

Aiming at solving the problem of cloth over-stretching, this paper proposed an extended XPBD method, which shortens the time step during the prediction stage to reduce the position error; sets the number of iterations at each time step to 1 during the iterative solution stage; simplifies the formula for solving the Lagrange multiplier; and optimizes the parallel grouping strategy of the Gauss-Seidel iterative solver. It effectively solves the problem of over-stretching of the cloth, and improves efficiency and simulation authenticity.

#### 4.1. Solving process summarizing

Assuming that the cloth distribution model is represented by $N$ vertices and $M$ constraints, the current particle position is $x_i$, and the predicted particle position at the next time is $p_i$. For a given time step $t$, initialize the positions of all particles $x_i$, velocity $v_i$, mass $m_i$. Use the Euler integral method to predict the position and velocity of all particles at the next moment, then use a parallel Gauss-Seidel iterative solver to iteratively solve the constraint to obtain the corrected particle position, and update the particle position and velocity information.

Algorithm 1: Our extended XPBD algorithm

- **Input initial state** $(x^0, v^0)$
- **Shorten the time step** $\Delta t_s = \frac{\Delta t_f}{n_{\text{steps}}}$
- **predict position** $\bar{x} = x^n + \Delta t_s v^n + \Delta t_s M^{-1} f_{\text{ext}}(x^n)$
- **compute $\Delta \lambda$ using Eq(14)**
- **compute $\Delta x$ using Eq(12)**
- **pdate** $\lambda^{n+1} = \Delta \lambda$
- **update** $x^{n+1} = \Delta x + \bar{x}$
- **if** $n < n_{\text{steps}}$, **n = n + 1**, **turn to (3)**
- **update velocities** $v^{n+1} = \frac{1}{\Delta t_s} (x^{n+1} - x^n)$
- **output** $(x^{n+1}, v^{n+1})$
4.2. Prediction stage

Firstly, all forces (such as gravity) that can not be converted into position constraints are recorded as the external force $f_{\text{ext}}(x_i)$, and then the Symplectic Eular integral is performed to predict the velocity and position based on the values of $f_{\text{ext}}(x_i)$. This predicted position refers to the ideal position that the particle should move to, which is calculated according to Newton's law of motion. Because of the existence of various damping, this predicted value can not be used as a real simulation result, and it should be modified.

The predicted position obtained by explicit integration of the external force is:

$$\ddot{x} = x^p + \Delta v^p + \Delta^2 M^{-1} f_{\text{ext}}(x^p)$$

(6)

The reduction of time step is an effective method to reduce the position error in dynamics. We divide the time step of the whole frame into $n$ sub steps:

$$\Delta_t = \frac{\Delta t}{n_{\text{steps}}}$$

(7)

We found that the effect of external force on the position is proportional to the $\Delta t^2$ by equation (6). This is due to the discretization of the second-order differential equation. And it has a large effect on the errors in each single step. For example, halving the time step would result in the position error being reduced to a quarter of the original value. It inspired us to use a smaller time step to effectively reduce the position error.

4.3. Iterative solution

After predicting the position, we used the Gauss-Seidel solver to iteratively correct the predicted position.

We derive equations of motion by starting with Newton's second law:

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

(8)

Here $x = [x_1, x_2, \ldots, x_n]^T$ is the system state (Information such as the position and direction of each particle). The gradient operator $\nabla$ is a row vector of partial derivatives, $U(x)$ is the potential energy of the system. Perform an implicit position-level time discretization of equations of motion (8), where the superscript $n$ indicates the time step index:

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

(9)

Use the constraint equation $C = [C_1(x), C_2(x), \ldots, C_m(x)]^T$ to construct the potential energy $U(x)$:

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

(10)

where $\alpha$ is a block diagonal compliance matrix corresponding to inverse stiffness.

Then the force from the elastic potential energy is given by the negative gradient of $U$ with respect to $x$:

$$f_{\text{elastic}} = -\nabla U(x) = -\nabla C^T \alpha^{-1} C$$

(11)

Substitute formula (11) into equation (9), a Lagrange multiplier is introduced to decompose the force into its direction and scalar components:

$$M\left(\frac{x^{n+1} - 2x^n + x^{n-1}}{\Delta t^2}\right) = \Delta t^2 f_{\text{elastic}} = -\nabla C^T \left(\frac{\alpha}{\Delta t^2}\right)^{-1} C = -\nabla C^T \alpha^{-1} C = \nabla C^T \lambda_{\text{elastic}}$$

(12)
Where \( \lambda_{\text{elastic}} = -\tilde{\alpha}^{-1} C(x) \) is the Lagrange multiplier. This step is the key to the algorithm. The calculation of the Lagrange multiplier takes into account the effect of time, so the stiffness of the material in the final simulation is time independent.

From equations (11) and (12) discrete constrained equations of motion can be derived:

\[
M (x^{n+1} - \tilde{x}) - \nabla C^T (x^{n+1}) \lambda^{n+1} = 0
\]

\[
C(x^{n+1}) + \tilde{\alpha} \lambda^{n+1} = 0
\]

We get the final equation of the system by simplifying:

\[
\begin{bmatrix}
M & -\nabla C^T (x_i) \\
\nabla C(x_i) & \tilde{\alpha}
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta \lambda
\end{bmatrix} =
\begin{bmatrix}
0 \\
h(x_i, \lambda_i)
\end{bmatrix}
\]

(15)

The left matrix of the equation is invariant during the simulation at this time.

Then, it is required that equation (15) can use the Schur complement of the mass matrix \( M \) to obtain a dimensionally reduced system, and calculate the correction \( \Delta \lambda \) of Lagrange multiplier:

\[
\begin{bmatrix}
\nabla C(x_i) M^{-1} \nabla C(x_i)^T + \tilde{\alpha}
\end{bmatrix} \Delta \lambda = -C(x_i) - \tilde{\alpha} \lambda_i
\]

(16)

Finally, the correction quantity \( \Delta x \) of the constraint to the position is obtained:

\[
\Delta x = M^{-1} \nabla C(x_i)^T \Delta \lambda
\]

(17)

Perform position projection for constraint which index is \( i \), \( \Delta x = M^{-1} \nabla C(x_i)^T \Delta \lambda \). The associated Lagrange multiplier increment is:

\[
\Delta \lambda_i = \frac{-C_i(x) - \tilde{\alpha} \lambda_i}{\nabla C_i M^{-1} \nabla C_i^T + \tilde{\alpha} i}
\]

(18)

We perform one iteration for each sub-step, and the initial Lagrange multiplier is always 0, so equation (18) can be simplified as:

\[
\Delta \lambda_i = \frac{-C_i(x)}{\nabla C_i M^{-1} \nabla C_i^T + \tilde{\alpha} i}
\]

(19)

When one iteration is used for each step, there is no need to store the Lagrange multiplier, which effectively simplifies the complexity of the calculation. Compared with the traditional integral solver with a large time step, the small time step significantly reduces the constraint error and damping, and reduces the visible stretch of the cloth.

4.4. Parallelization of the Gauss-Seidel solver

In order to solve the problem of slow convergence, method of grouping constraints before performing parallel calculations can greatly improve the iteration efficiency of the Gauss-Seidel solver. The principle of grouping is that any two constraints of each group will not share the same vertex, so that the constraints in the same group can be solved in parallel. The grouping results of constraints affect the number of calls to the GPU kernel, so the reasonable partitioning strategy can effectively improve the solution speed of the GPU.

The basic PBD constraint grouping algorithm is to loop through all constraints and find constraints without common edges for grouping. The biggest disadvantage is that the result of grouping depends on the traversal order of constraint nodes during the iteration process, and there are large difference between the number of constraints in each group.
In order to optimize the grouping results, we use constraint topology graphs to group constraints. The constrained topology graph is an undirected graph. Any two nodes with common edges in the undirected graph will be painted in different colors. Then the entire constrained topology graph is colored. Any two nodes with the same color must not be connected, so nodes with the same color are considered to be in the same group. And the number of colors is the number of groups. Each group of nodes is then sorted. The degree of the constraint node is defined as the number of its adjacent critical points. The smaller the degree, the fewer its neighboring nodes, and the fewer the dependencies between this constraint and other constraints. So the coloring order of the constraints is placed at a later position, and nodes with more dependencies are preferentially selected for coloring. Through grouping and sorting, the number of nodes in each group can be averaged to the greatest extent, and finally an optimal colored result with the least colors can be obtained.

5. Experimental results and analysis

In order to evaluate the cloth simulation method based on the extended position dynamics proposed in this paper, comparative experiments with different iteration times and time step length were carried out. The experimental environment is Intel Xeon E5-2650 @ 2.00GHz, 32GB RAM, 64-bit Windows 10 operating system. The GPU we used is NVIDIA Quadro 5000, the number of cores is 352, and the memory is 2GB GDDR5. The programming platform is Microsoft Visual Studio 2017 and OpenGL.

5.1. Looking for the right time step

In order to find the most suitable time step for cloth simulation, take the cloth that sags naturally under gravity as an example. The cloth model is composed of a 30x30 particle grid, and the particles are connected by 5,310 distance constraints. Set the initial time step to 0.05s, divide the time step into 5, 10, 50, 100 equal parts (Figure 3.b, c, d, e), and perform an iteration for each time step, with the original XPBD (Figure 3.a). The comparison chart is as follows.

Through experimental comparison, we can find that when the time step value increased, as is shown in Figure 4b and c, the cloth is over-stretched. Meanwhile, when the time step is reduced to a certain degree, eg. dividing into 50 or 100 equal parts, the stretch of the cloth tends to be stable, but the simulation time is still increasing. The critical value is 50, which can solve the stretch problem of cloth without wasting too much unnecessary simulation time. Finally, we choose the value of time step 0.05/50s.

5.2. Simulation effect comparison

Figure 4 shows the comparison of the experimental results of free fall of cloth on the ring under the action of gravity. The cloth model consists of a 30x30 particle grid, and the particles are connected by 5,310 distance constraints. Figure (a): 50 iterations of 1 sub-step of PBD, each frame takes 2.21ms; Figure (b): 50 iterations of 1 sub-step of XPBD, each frame takes 1.93ms; Figure (c): 50 sub-steps of improved XPBD, 1 iteration, each frame takes time 1.42ms. Compared to the original XPBD, our method shows less stretching, and a 37% increase in simulation speed.
Through comparison, it is found that with the increase of the number of iterations of the traditional PBD method, the cloth performance is too stiff. Although the XPBD method optimizes this, the problem of over-stretching of the cloth occurs. The improved XPBD method proposed in this paper solves the over-stretching problem of XPBD, while retaining the realistic wrinkles of the cloth simulation. And also we accelerate the simulation speed of the cloth through an improved parallel solver.

6. Conclusion
In order to achieve realistic effect, cloth should be simulated with no over-stretching and rich folds. For the folding problem, we use the XPBD method to avoid the problem of stiffness dependence, and for the problem of excessive stretching, we proposed an extended XPBD method. Based on the iterative solution of XPBD, we reduced the time step and execute a single constraint iteration for each sub step at the same time. It not only avoids the over-stretching of cloth, but also preserves the realistic drape. Finally, we accelerated the GS iterative solution process by optimizing the GPU-based grouping parallel algorithm, and improved the simulation speed of the cloth.

Acknowledgments
This work is supported by two projects of Jilin Province, China, which are major science and technology bidding projects (No. 20170203004GX), natural science foundation (No. 20190201255JC).

References
[1] Müller M, Heidelberger B, Hennix M, et al. (2007) Position based dynamics. Journal of Visual Communication and Image Representation, 18: 109-118.
[2] Macklin M, Muller M, Chentanez N, et al. (2016) XPBD: position-based simulation of compliant constrained dynamics. In: Motion in Games. New York. pp. 49-54.
[3] Hauth M, Etzmuss O, Strasser W, et al. (2003) Analysis of numerical methods for the simulation of deformable models. The Visual Computer, 19: 581-600.
[4] Breen D E, House D H, Wozny M J, et al. (1994) Predicting the drape of woven cloth using interacting particles. In: International Conference on Computer Graphics and Interactive Techniques. New York. pp. 365-372.
[5] Liu T, Bargteil A W, Obrien J F, et al. (2013) Fast simulation of mass-spring systems. International Conference on Computer Graphics and Interactive Techniques, 32:214.
[6] Goldenthal R, Harmon D, Fattal R, et al. (2007) Efficient simulation of inextensible cloth. International Conference on Computer Graphics and Interactive Techniques, 26:49.
[7] Hong M, Choi M, Jung S, et al. (2005) Effective Constrained Dynamic Simulation Using Implicit Constraint Enforcement. In: International Conference on Robotics and Automation, Barcelona. pp. 4520-4525
[8] Bender J, Muller M, Otaduy M A, et al. (2014) A Survey on Position-Based Simulation Methods in Computer Graphics. Computer Graphics Forum, 33: 228-251
[9] Bouaziz S, Martin S, Liu T, et al. (2014) Projective dynamics: fusing constraint projections for fast simulation. International Conference on Computer Graphics and Interactive Techniques, 33:154.

[10] Macklin M, Muller M, Chentanez N, et al. (2014) Unified particle physics for real-time applications. International Conference on Computer Graphics and Interactive Techniques, 33:153.