GAMES103: Intro to Physics-Based Animation

Constraint Approaches

Huamin Wang

Nov 2021

Topics for the Day

- Strain Limiting and Position Based Dynamics
- Projective Dynamics
- Constrained Dynamics

Strain Limiting
and
Position Based Dynamics

The Stiffness Issue

- Real-world fabrics resist strongly to stretching, once they stretch beyond certain limits.

- But, increasing the stiffness can cause problems.
  - Explicit integrators will be *unstable*
    - Solution: smaller time steps and more computational time.
  - The linear systems involved in Implicit integrators will be *ill-conditioned*.
    - Solution: more iterations and computational time.

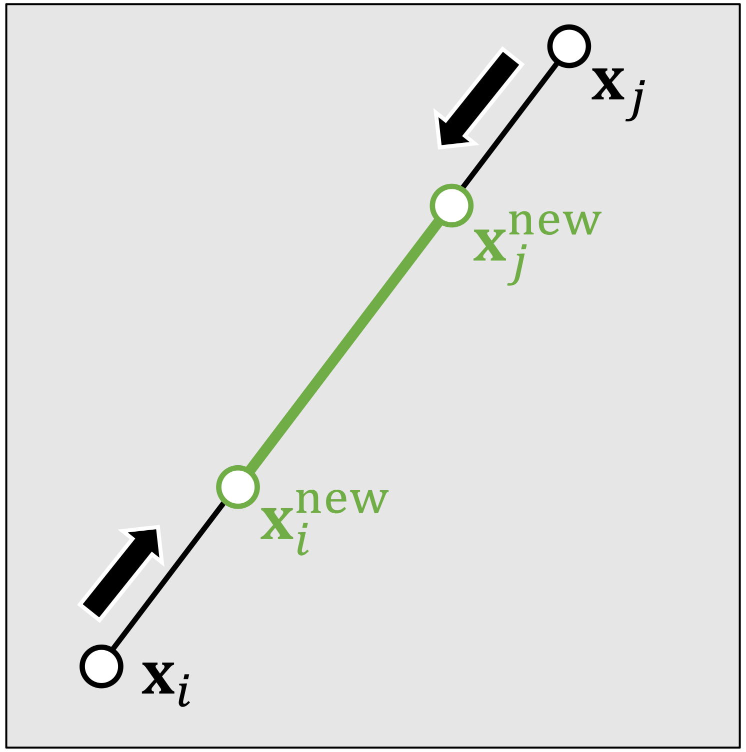
- Can we achieve high stiffness, with a low computational cost?

A Single Spring

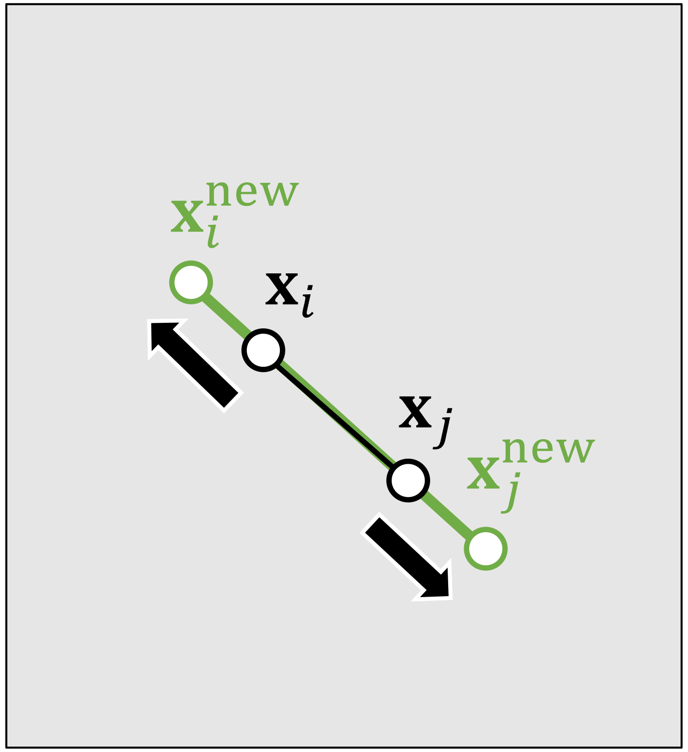
If a spring is infinitely stiff, we can treat the length as a constraint and define a projection function.

$$\boxed{\phi(\mathbf{x}) = \|\mathbf{x}_i - \mathbf{x}_j\| - L = 0}$$

Constraint



Stretched case



Compressed Case

A Single Spring

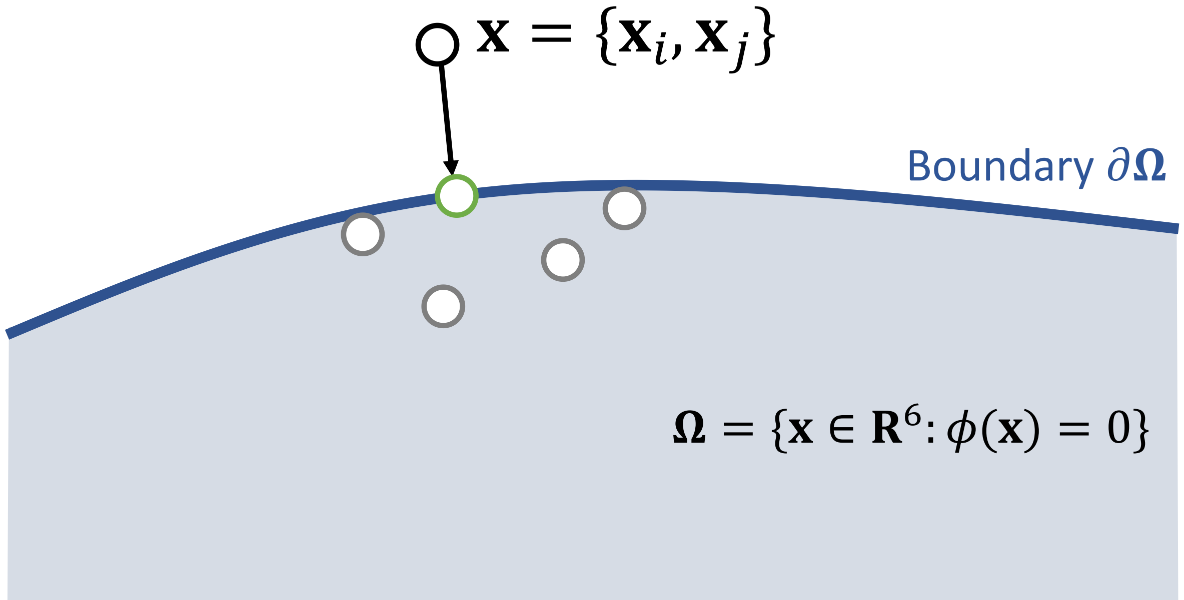
If a spring is infinitely stiff, we can treat the length as a constraint and define a projection function.

$$\{\mathbf{x}_i^{\text{new}}, \mathbf{x}_j^{\text{new}}\} = \text{argmin} \frac{1}{2} \left\{ m_i \|\mathbf{x}_i^{\text{new}} - \mathbf{x}_i\|^2 + m_j \left\|\mathbf{x}_j^{\text{new}} - \mathbf{x}_j\right\|^2 \right\}$$

such that $\phi(\mathbf{x}) = 0$

$$\boxed{\phi(\mathbf{x}) = \left\|\mathbf{x}_i - \mathbf{x}_j\right\| - L = 0}$$

Constraint

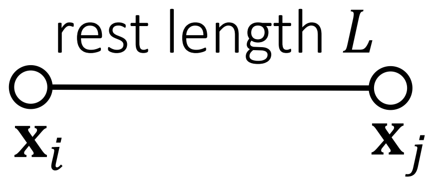


A Single Spring

If a spring is infinitely stiff, we can treat the length as a constraint and define a projection function.

$$\{\mathbf{x}_i^{\text{new}}, \mathbf{x}_j^{\text{new}}\} = \text{argmin} \frac{1}{2}\left\{m_i \|\mathbf{x}_i^{\text{new}} - \mathbf{x}_i\|^2 + m_j \left\|\mathbf{x}_j^{\text{new}} - \mathbf{x}_j\right\|^2\right\}$$

such that $\phi(\mathbf{x}) = 0$



$$\boxed{\phi(\mathbf{x}) = \left\|\mathbf{x}_i - \mathbf{x}_j\right\| - L = 0}$$

Constraint

$$\mathbf{x}^{\text{new}} \longleftarrow \text{Projection}(\mathbf{x})$$

$$\mathbf{x}_i^{\text{new}} \longleftarrow \mathbf{x}_i - \frac{m_j}{m_i + m_j}\left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\| - L\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|}$$

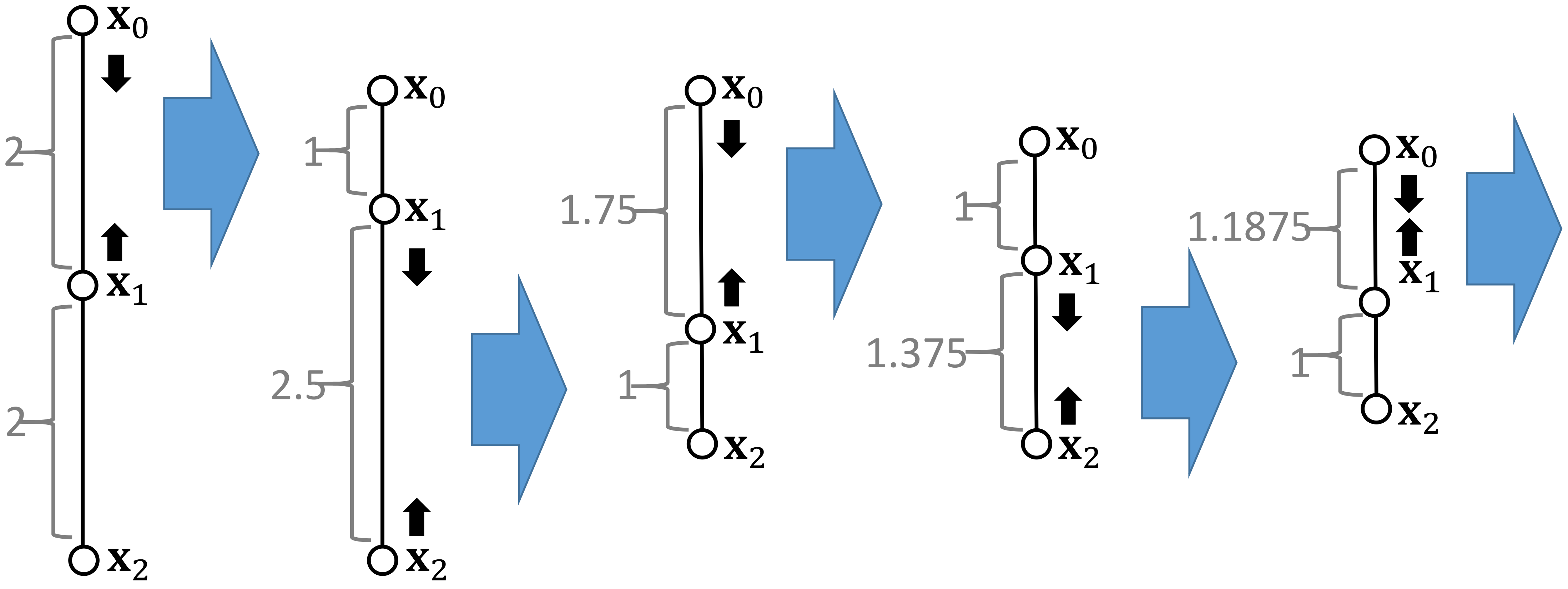
$$\mathbf{x}_j^{\text{new}} \longleftarrow \mathbf{x}_j + \frac{m_i}{m_i + m_j}\left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\| - L\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|}$$

$$\phi(\mathbf{x}^{\text{new}}) = \left\|\mathbf{x}_i^{\text{new}} - \mathbf{x}_j^{\text{new}}\right\| - L = \left\|\mathbf{x}_i - \mathbf{x}_j - \mathbf{x}_i + \mathbf{x}_j + L\right\| - L = 0$$

By default, $m_i = m_j$, but we can also set $m_i = \infty$ for stationary nodes.

Multiple Springs – A Gauss-Seidel Approach

What about multiple springs? The Gauss-Seidel approach projects each spring sequentially in a certain order. Imagine two springs with unit rest lengths…



Multiple Springs – A Gauss-Seidel Approach

Projection (by Gauss-Seidel)

For $k = 0 \ldots K$

For every edge $e = \{i, j\}$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i - \tfrac{1}{2}\left(\|\mathbf{x}_i - \mathbf{x}_j\| - L_e\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}$$

$$\mathbf{x}_j \leftarrow \mathbf{x}_j + \tfrac{1}{2}\left(\|\mathbf{x}_i - \mathbf{x}_j\| - L_e\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}$$

- We cannot ensure the satisfaction of every constraint. But the more iterations we use, the better those constraints are satisfied.
- Although the name is related to Gauss-Seidel, it differs from Gauss-Seidel. It is more relevant to stochastic gradient descent (in machine learning).
- The order matters. The order can cause bias and affect convergence behavior.

Multiple Springs – A Jacobi Approach

- To avoid bias, the Jacobi approach projects all of the edges simultaneously and then linearly blend the results.
- The problem is an even lower convergence rate.
- Again, the more iterations it uses, the better the constraints are enforced.

Projection (by Jacobi)

For $k = 0 \ldots K$

    For every vertex i

$$\mathbf{x}_i^{\text{new}} \leftarrow \mathbf{0}$$

$$n_i \leftarrow 0$$

    For every edge $e = \{i, j\}$

$$\mathbf{x}_i^{\text{new}} \leftarrow \mathbf{x}_i^{\text{new}} + \mathbf{x}_i - \frac{1}{2}\left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\| - L_e\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|}$$

$$\mathbf{x}_j^{\text{new}} \leftarrow \mathbf{x}_j^{\text{new}} + \mathbf{x}_j + \frac{1}{2}\left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\| - L_e\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|}$$

$$n_i \leftarrow n_i + 1$$

$$n_j \leftarrow n_j + 1$$

    For every vertex i

$$\mathbf{x}_i \leftarrow (\mathbf{x}_i^{\text{new}} + \alpha \mathbf{x}_i)/(n_i + \alpha)$$

Position Based Dynamics (PBD)

Position based dynamics (PBD) is based on the projection function.

- The stiffness behavior, i.e., how tightly constraints are enforced, is subject to non-physical factors.
  - The number of iterations
  - The mesh resolution
- The velocity update following projection is important to dynamic effects.
- This method is applicable to other constraints as well, including triangle constraints, volume constraints, and collision constraints.
  - To implement these constraints, simply define their projection functions.

A PBD Simulator

//Do Simulation, update $\mathbf{x}$ and $\mathbf{v}$

$\mathbf{v} \leftarrow \ldots$

$\mathbf{x} \leftarrow \ldots$

//Now PBD starts.

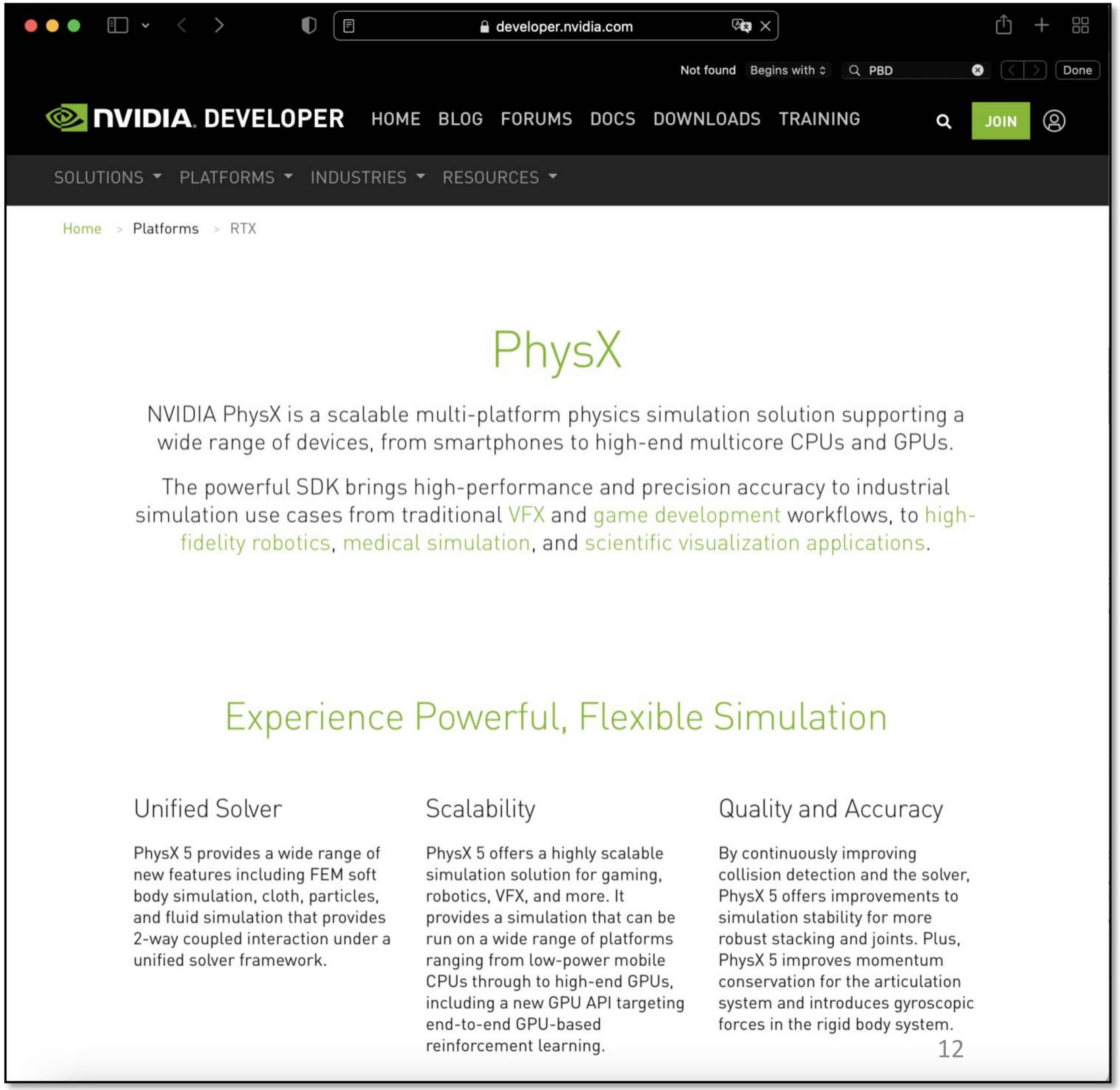
$\mathbf{x}^{\text{new}} \leftarrow \text{Projection}(\mathbf{x})$

$\mathbf{v} \leftarrow \mathbf{v} + (\mathbf{x}^{\text{new}} - \mathbf{x})/\Delta t$

$\mathbf{x} \leftarrow \mathbf{x}^{\text{new}}$

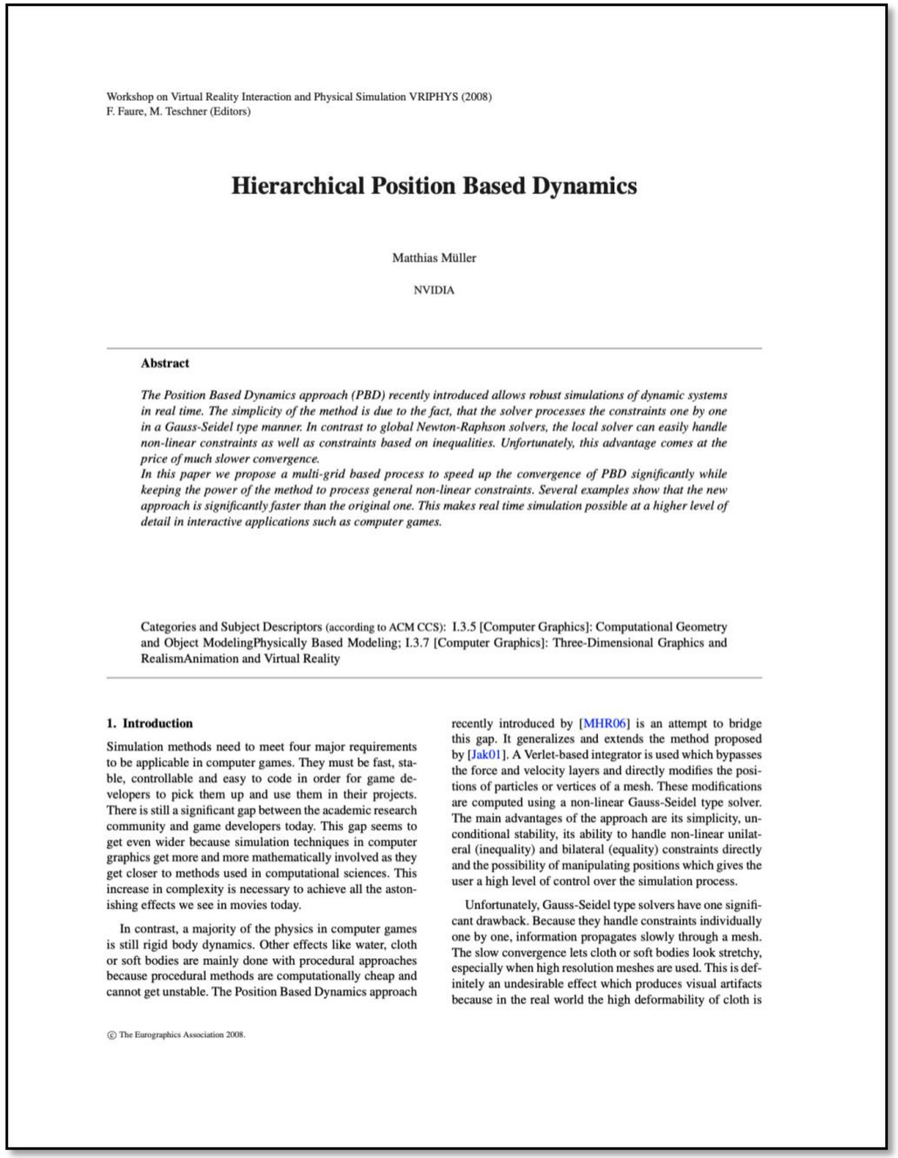
Pros and Cons of PBD

- Pros
  - Parallelable on GPUs (PhysX)
  - Easy to implement
  - Fast in low resolutions < 1000 points
  - Generic, can handle other coupling and constraints, including fluids
- Cons
  - Not physically correct
  - Low performance in high resolutions
    - Hierarchical approaches (can cause oscillation and other issues…)
    - Acceleration approaches, like Chebyshev



After-Class Reading

Muller. 2008. *Hierarchical Position Based Dynamics.* VRIPHYS.


Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2008)
F. Faure, M. Teschner (Editors)

Hierarchical Position Based Dynamics

Matthias Müller

NVIDIA

Abstract

The Position Based Dynamics approach (PBD) recently introduced allows robust simulations of dynamic systems in real time. The simplicity of the method is due to the fact, that the solver processes the constraints one by one in a Gauss-Seidel type manner. In contrast to global Newton-Raphson solvers, the local solver can easily handle non-linear constraints as well as constraints based on inequalities. Unfortunately, this advantage comes at the price of much slower convergence.
In this paper we propose a multi-grid based process to speed up the convergence of PBD significantly while keeping the power of the method to process general non-linear constraints. Several examples show that the new approach is significantly faster than the original one. This makes real time simulation possible at a higher level of detail in interactive applications such as computer games.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object ModelingPhysically Based Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and RealismAnimation and Virtual Reality

1. Introduction

Simulation methods need to meet four major requirements to be applicable in computer games. They must be fast, stable, controllable and easy to code in order for game developers to pick them up and use them in their projects. There is still a significant gap between the academic research community and game developers today. This gap seems to get even wider because simulation techniques in computer graphics get more and more mathematically involved as they get closer to methods used in computational sciences. This increase in complexity is necessary to achieve all the astonishing effects we see in movies today.

In contrast, a majority of the physics in computer games is still rigid body dynamics. Other effects like water, cloth or soft bodies are mainly done with procedural approaches because procedural methods are computationally cheap and cannot get unstable. The Position Based Dynamics approach recently introduced by [MHR06] is an attempt to bridge this gap. It generalizes and extends the method proposed by [Jak01]. A Verlet-based integrator is used which bypasses the force and velocity layers and directly modifies the positions of particles or vertices of a mesh. These modifications are computed using a non-linear Gauss-Seidel type solver. The main advantages of the approach are its simplicity, unconditional stability, its ability to handle non-linear unilateral (inequality) and bilateral (equality) constraints directly and the possibility of manipulating positions which gives the user a high level of control over the simulation process.

Unfortunately, Gauss-Seidel type solvers have one significant drawback. Because they handle constraints individually one by one, information propagates slowly through a mesh. The slow convergence lets cloth or soft bodies look stretchy, especially when high resolution meshes are used. This is definitely an undesirable effect which produces visual artifacts because in the real world the high deformability of cloth is

© The Eurographics Association 2008.

Strain Limiting

Strain limiting aims at using the projection function for correction only.

Still do some dynamics

A Simulator with Strain Limiting

//Do Simulation, update $\mathbf{x}$ and $\mathbf{v}$

$\mathbf{v} \leftarrow \ldots$

$\mathbf{x} \leftarrow \ldots$

//Now strain limiting starts.

$\mathbf{x}^{\text{new}} \leftarrow \text{Projection}(\mathbf{x})$

$\mathbf{v} \leftarrow \mathbf{v} + (\mathbf{x}^{\text{new}} - \mathbf{x})/\Delta t$

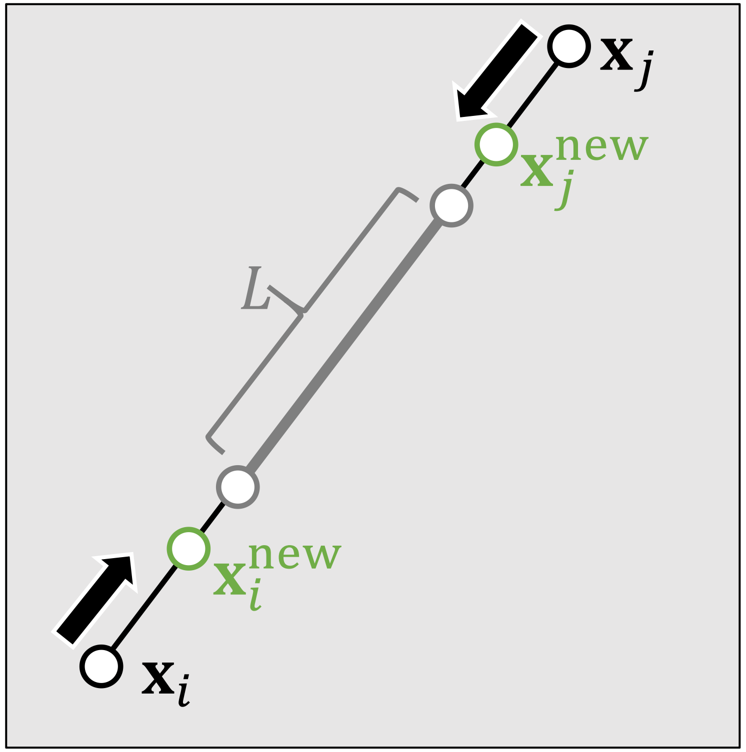
$\mathbf{x} \leftarrow \mathbf{x}^{\text{new}}$

Spring Strain Limit

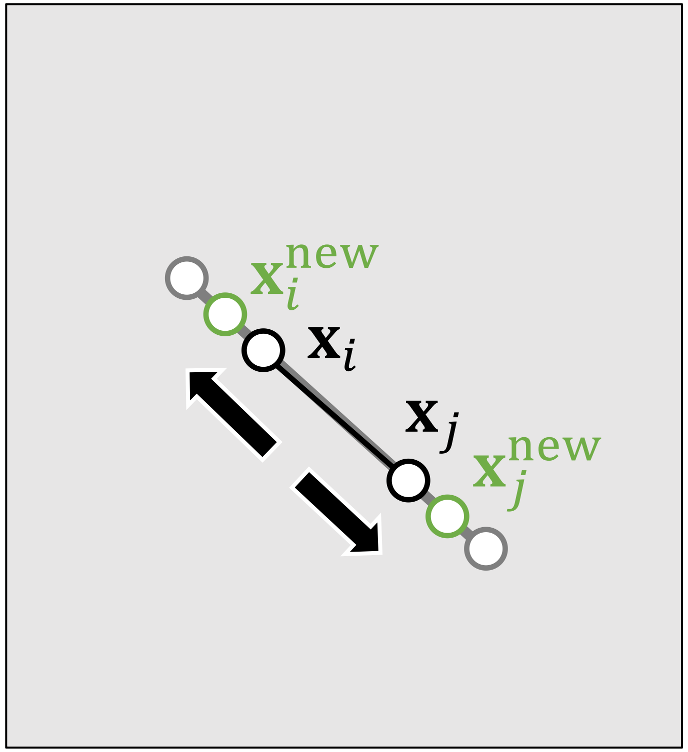
We can set the spring strain, i.e., the stretching ratio σ, to be within a limit.

$$\boxed{\sigma^{\min} \leq \frac{1}{L}\left\|\mathbf{x}_i - \mathbf{x}_j\right\| \leq \sigma^{\max}}$$

Constraint



Stretched case



Compressed Case

Spring Strain Limit

We can set the spring strain, i.e., the stretching ratio σ, to be within a limit.

$$\sigma^{\min} \le \frac{1}{L}\|\mathbf{x}_i - \mathbf{x}_j\| \le \sigma^{\max}$$

Constraint

$$\mathbf{x}^{\text{new}} \leftarrow \text{Projection}(\mathbf{x})$$

$$\sigma \leftarrow \frac{1}{L}\|\mathbf{x}_i - \mathbf{x}_j\|$$

$$\sigma_0 \leftarrow \min\left(\max\left(\sigma, \sigma^{\min}\right), \sigma^{\max}\right)$$

$$\mathbf{x}_i^{\text{new}} \leftarrow \mathbf{x}_i - \frac{m_j}{m_i + m_j}\left(\|\mathbf{x}_i - \mathbf{x}_j\| - \sigma_0 L\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}$$

$$\mathbf{x}_j^{\text{new}} \leftarrow \mathbf{x}_j + \frac{m_j}{m_i + m_j}\left(\|\mathbf{x}_i - \mathbf{x}_j\| - \sigma_0 L\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}$$

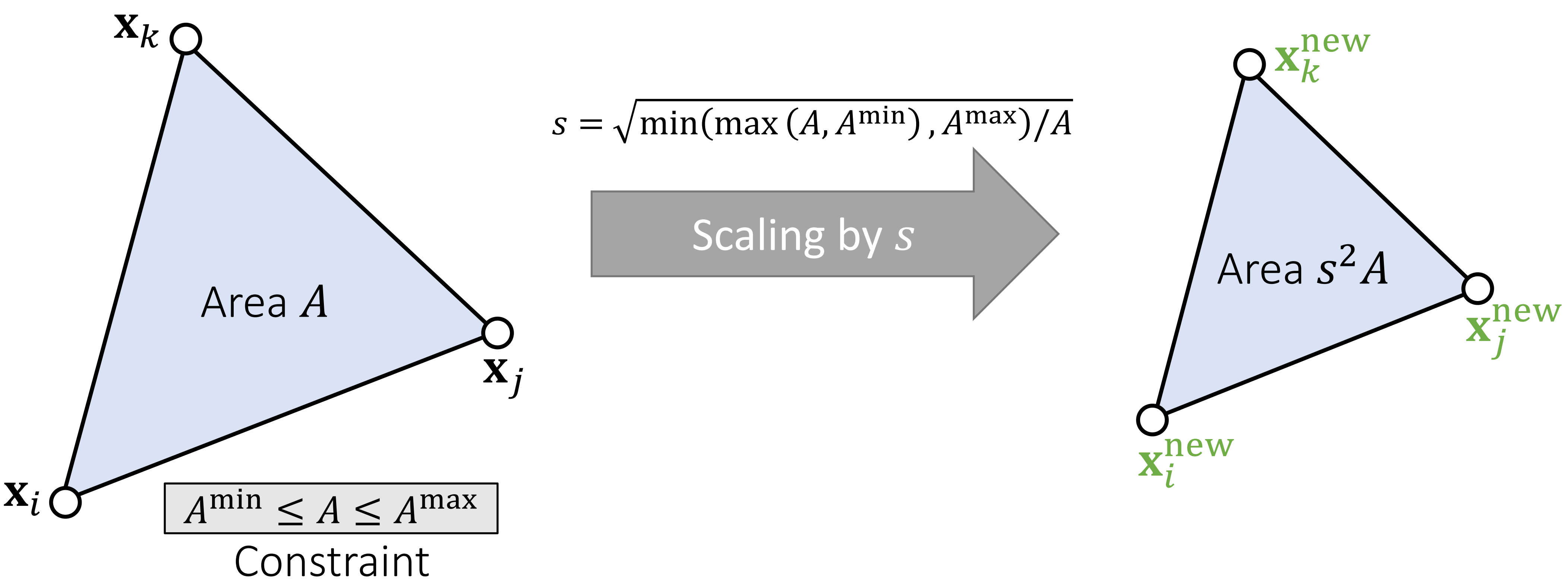
PBD: $\sigma_0 \equiv 1$; No limit: $\sigma^{\min}, \sigma^{\max} \leftarrow \infty$

Triangle Area Limit

We can limit the triangle area as well. To do so, we define a scaling factor.

$$\{\mathbf{x}_i^{\text{new}}, \mathbf{x}_i^{\text{new}}, \mathbf{x}_k^{\text{new}}\} = \text{argmin}\frac{1}{2}\left\{m_i\|\mathbf{x}_i^{\text{new}} - \mathbf{x}_i\|^2 + m_j\left\|\mathbf{x}_j^{\text{new}} - \mathbf{x}_j\right\|^2 + m_j\|\mathbf{x}_k^{\text{new}} - \mathbf{x}_k\|^2\right\}$$

such that the constraint is satisfied.

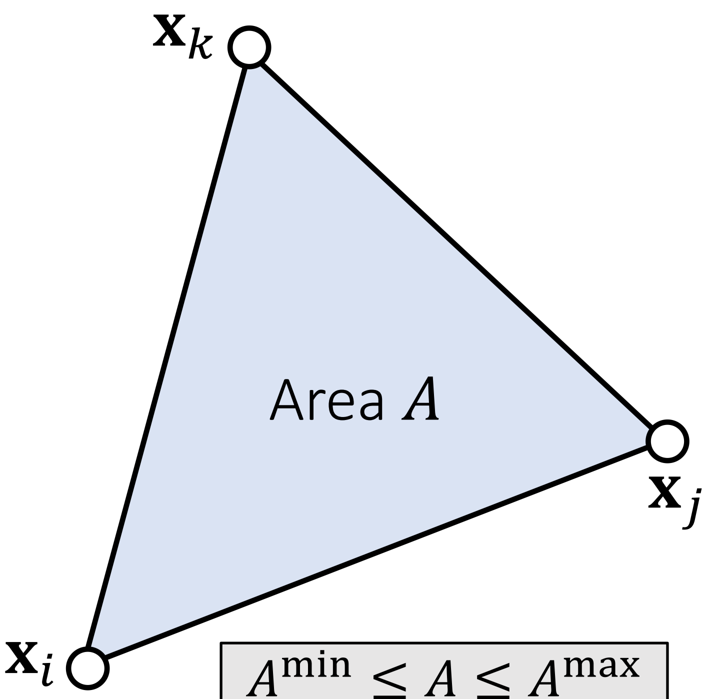


Triangle Area Limit

To limit the area, we use the fact that the mass center doesn't move.

$$\{\mathbf{x}_i^{\text{new}}, \mathbf{x}_i^{\text{new}}, \mathbf{x}_k^{\text{new}}\} = \operatorname{argmin} \frac{1}{2}\left\{m_i\|\mathbf{x}_i^{\text{new}} - \mathbf{x}_i\|^2 + m_j\left\|\mathbf{x}_j^{\text{new}} - \mathbf{x}_i\right\|^2 + m_j\left\|\mathbf{x}_j^{\text{new}} - \mathbf{x}_i\right\|^2\right\}$$

such that the constraint is satisfied.



$$\boxed{A^{\min} \le A \le A^{\max}}$$

Constraint

$$\mathbf{x}^{\text{new}} \longleftarrow \text{Projection}(\mathbf{x})$$

$$A \longleftarrow \frac{1}{2}\left\|\left(\mathbf{x}_j - \mathbf{x}_i\right) \times \left(\mathbf{x}_k - \mathbf{x}_i\right)\right\|$$

$$s \longleftarrow \sqrt{\min(\max\left(A, A^{\min}\right), A^{\max})/A}$$

$$\mathbf{c} \longleftarrow \frac{1}{m_i + m_j + m_k}\left(m_i\mathbf{x}_i + m_j\mathbf{x}_j + m_k\mathbf{x}_k\right)$$

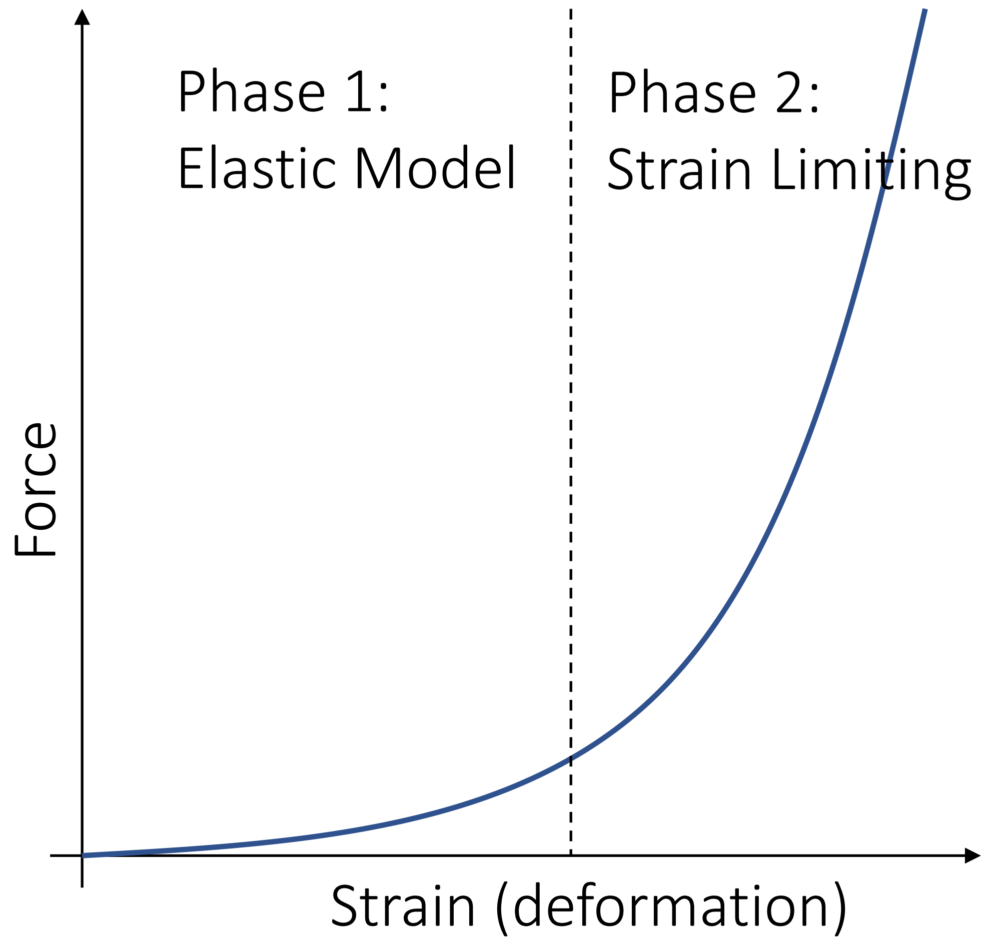
$$\mathbf{x}_i^{\text{new}} \longleftarrow \mathbf{c} + s(\mathbf{x}_i - \mathbf{c})$$

$$\mathbf{x}_j^{\text{new}} \longleftarrow \mathbf{c} + s\left(\mathbf{x}_j - \mathbf{c}\right)$$

$$\mathbf{x}_k^{\text{new}} \longleftarrow \mathbf{c} + s(\mathbf{x}_k - \mathbf{c})$$

Strain Limiting in Simulation

- Strain limiting is widely used in physics-based simulation, typically for avoiding instability and artifacts due to large deformation.
- Strain limiting is useful for nonlinear effects, in a biphasic way.
- Strain limiting also helps address the locking issue.



After-Class Reading (optional)

Provot. 1995. *Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior.* Graphics Interface.

Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior

Xavier Provot
Institut National de Recherche en Informatique et Automatique (INRIA)
B.P. 105
78153 Le Chesnay Cedex
France

Xavier.Provot@inria.fr
http://www-rocq.inria.fr/syntim/research/provot/

Abstract

This paper describes a physically-based model for animating cloth objects, derived from elastically deformable models, and improved in order to take into account the non-elastic properties of woven fabrics. A cloth object is first approximated to a deformable surface composed of a network of masses and springs, the movement of which is evaluated using the numerical integration of the fundamental law of dynamics. We show that when a concentration of high stresses occurs in a small region of the surface, the local deformation becomes unrealistic compared to real deformations of textiles. With such an elastic model, the only solution to decrease these deformations has been so far to increase the stiffness of the deformed springs, but we show that it dramatically increases the cost of the algorithm. We present therefore a new method to adapt our model to the particularly stiff properties of textiles, inspired from dynamic inverse procedures.

Résumé

Cet article décrit un modèle physique d'animation des tissus, variante des modèles élastiques déformables, et amélioré de façon à prendre en compte les propriétés non élastiques des textiles. Nous modélisons tout d'abord une pièce de tissu par une surface déformable, constituée d'un réseau de masses et de ressorts. Son mouvement est évalué grâce à l'intégration numérique de la loi fondamentale de la dynamique. Nous montrons que lorsqu'une forte concentration de contraintes apparaît à certains endroits de la surface, la déformation locale y devient irréaliste comparée aux déformations rencontrées dans les tissus réels. Avec un tel modèle élastique, la seule solution permettant d'atténuer cette déformation était jusqu'à présent d'augmenter la raideur des ressorts déformés, mais nous montrons que ceci faisait croître dramatiquement le coût de l'algorithme. Nous présentons donc ici une nouvelle méthode permettant d'adapter notre modèle aux propriétés particulièrement rigides des textiles, inspirée des procédures de dynamique inverse.

Keywords: Physically-based models, deformable surfaces, cloth animation, rigid behavior.

1 Introduction

1.1 Background

Woven fabrics have been widely studied in computer graphics in order to find appropriate models describing their particular properties, namely their static behavior (e.g. drape) and their dynamic behavior (e.g. buckling propagation). Early studies can be found in [1, 2, 3, 4], but regarding cloth animation with which we were mostly concerned, physically-based models have proved to be both the most efficient and realistic. Among the physically-based models used in cloth animation, *elastically deformable models* have been used successfully in order to give a representation of the behavior of various cloth objects such as flags, tablecloths, or even garments dressing synthetic actors [5, 6, 7].

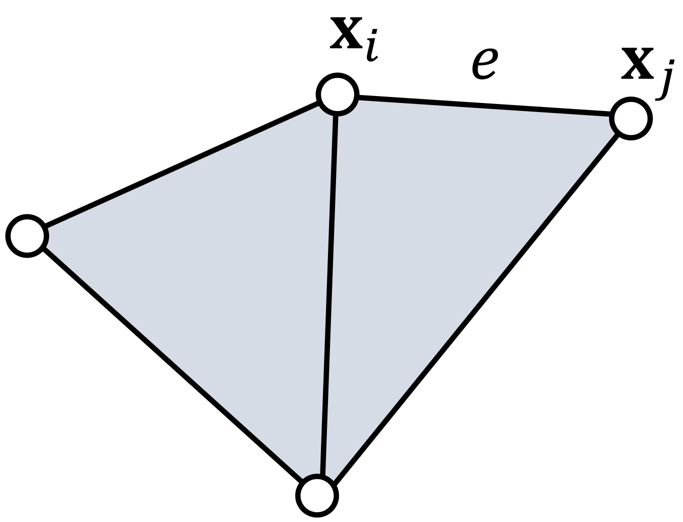
1.2 Realism

However, one of the problems encountered in this kind of modelization is that woven fabrics are far from having ideal elastic properties. This is why, under certain conditions and stresses, these elastic models behave more like sheets of rubber, or gum, than like textiles. This behavior occurs especially when the elastic model is subject to high constraints, and therefore to *high "super-elastic" deformation rates.* Such high constraints do not appear in

Projective Dynamics

Projective Dynamics

Instead of blending projections in a Jacobi or Gauss-Seidel fashion as in PBD, projective dynamics uses projection to define a quadratic energy.



$$E(\mathbf{x}) = \sum_{e=\{i,j\}} \frac{1}{2} \left\| (\mathbf{x}_i - \mathbf{x}_j) - (\mathbf{x}_{e,i}^{\mathrm{new}} - \mathbf{x}_{e,j}^{\mathrm{new}}) \right\|^2$$

$$\{\mathbf{x}_{e,i}^{\mathrm{new}}, \mathbf{x}_{e,j}^{\mathrm{new}}\} = \mathrm{Projection}_e(\mathbf{x}_i, \mathbf{x}_j) \text{ for every edge } e$$

$$\mathbf{f}_i = -\nabla_i E(\mathbf{x}) = -k \sum_{e: i \in e} (\mathbf{x}_i - \mathbf{x}_j) - (\mathbf{x}_{e,i}^{\mathrm{new}} - \mathbf{x}_{e,j}^{\mathrm{new}})$$

$$= -k \sum_{e: i \in e} (\|x_i - x_j\| - L_e) \frac{x_i - x_j}{\|x_i - x_j\|}$$

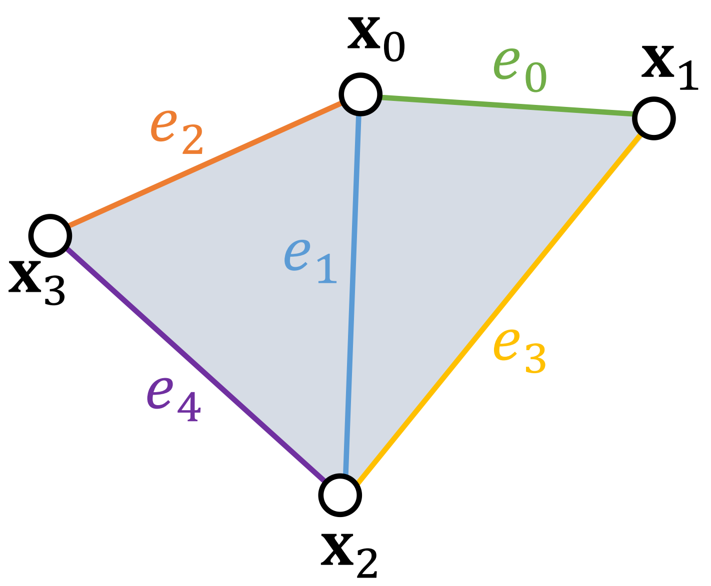
identical to spring force

$$E(x) = \sum_{e=\{i,j\}} \frac{k}{2} (\|x_i - x_j\| - L_e)^2$$

identical to spring energy

Projective Dynamics – Explained

Instead of blending projections in a Jacobi or Gauss-Seidel fashion as in PBD, projective dynamics uses <u>projection</u> to define a <u>quadratic</u> energy.



$$E(\mathbf{x}) = \boxed{\tfrac{1}{2}\left\|(\mathbf{x}_0-\mathbf{x}_1)-\left(\mathbf{x}_{0,0}^{\text{new}}-\mathbf{x}_{0,1}^{\text{new}}\right)\right\|^2} + \boxed{\tfrac{1}{2}\left\|(\mathbf{x}_0-\mathbf{x}_2)-\left(\mathbf{x}_{1,0}^{\text{new}}-\mathbf{x}_{1,2}^{\text{new}}\right)\right\|^2} + \boxed{\tfrac{1}{2}\left\|(\mathbf{x}_0-\mathbf{x}_3)-\left(\mathbf{x}_{2,0}^{\text{new}}-\mathbf{x}_{2,3}^{\text{new}}\right)\right\|^2} + \boxed{\tfrac{1}{2}\left\|(\mathbf{x}_1-\mathbf{x}_2)-\left(\mathbf{x}_{3,1}^{\text{new}}-\mathbf{x}_{3,2}^{\text{new}}\right)\right\|^2} + \boxed{\tfrac{1}{2}\left\|(\mathbf{x}_2-\mathbf{x}_3)-\left(\mathbf{x}_{4,2}^{\text{new}}-\mathbf{x}_{4,3}^{\text{new}}\right)\right\|^2}$$

$$\mathbf{f}_0 = \boxed{\left(\mathbf{x}_{0,0}^{\text{new}}-\mathbf{x}_{0,1}^{\text{new}}\right)-(\mathbf{x}_0-\mathbf{x}_1)} + \boxed{\left(\mathbf{x}_{1,0}^{\text{new}}-\mathbf{x}_{1,2}^{\text{new}}\right)-(\mathbf{x}_0-\mathbf{x}_2)} + \boxed{\left(\mathbf{x}_{2,0}^{\text{new}}-\mathbf{x}_{2,3}^{\text{new}}\right)-(\mathbf{x}_0-\mathbf{x}_3)}$$

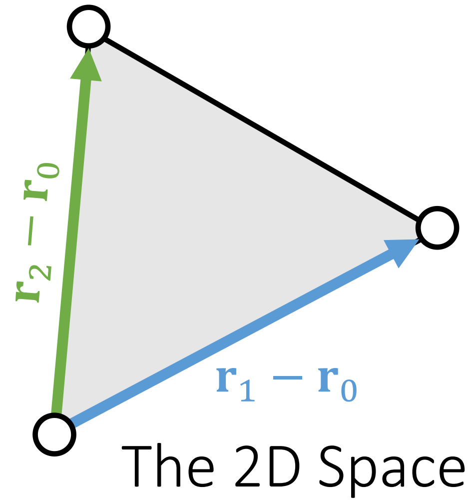
$$\mathbf{H} = \begin{bmatrix} 3\mathbf{I} & -\mathbf{I} & -\mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & 2\mathbf{I} & -\mathbf{I} & \\ -\mathbf{I} & -\mathbf{I} & 3\mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & & -\mathbf{I} & 2\mathbf{I} \end{bmatrix}$$

Projective Dynamics – Shape Matching

Shape matching is also projective dynamics, if we view rotation as projection:

$$E(\mathbf{x}) = \frac{1}{2}\left\| [\mathbf{x}_1 - \mathbf{x}_0 \quad \mathbf{x}_2 - \mathbf{x}_0][\mathbf{r}_1 - \mathbf{r}_0 \quad \mathbf{r}_2 - \mathbf{r}_0]^{-1} - \boxed{\mathbf{R}} \right\|^2$$

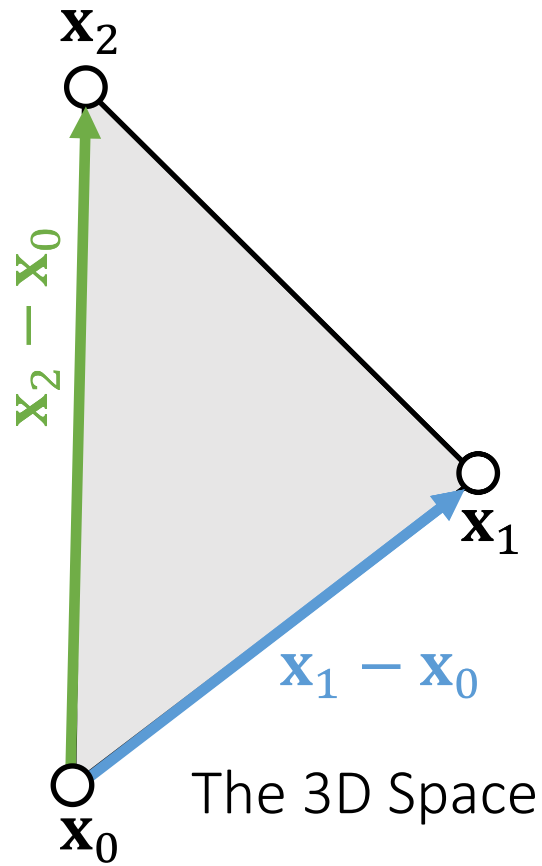
projection



Assuming that $\mathbf{R}$ is constant,

$$\mathbf{f}_0 = -\nabla_0 E(\mathbf{x})$$
$$\mathbf{f}_1 = -\nabla_1 E(\mathbf{x})$$
$$\mathbf{f}_2 = -\nabla_2 E(\mathbf{x})$$

$$\mathbf{H} = \frac{\partial E^2(\mathbf{x})}{\partial \mathbf{x}^2}$$ is a constant!



Simulation by Projective Dynamics

- According to implicit integration and Newton's method, a projective dynamics simulator looks as follows, with matrix $\mathbf{A} = \frac{1}{\Delta t^2}\mathbf{M} + \mathbf{H}$ being constant.

- We can use a direct solver with only one factorization of $\mathbf{A}$.

Initialize $\mathbf{x}^{(0)}$, often as $\mathbf{x}^{[0]}$ or $\mathbf{x}^{[0]} + \Delta t \mathbf{v}^{[0]}$

For $k = 0 \ldots K$

Recalculate projection

Solve $\left(\frac{1}{\Delta t^2}\mathbf{M} + \mathbf{H}\right)\Delta\mathbf{x} = -\frac{1}{\Delta t^2}\mathbf{M}\left(\mathbf{x}^{(k)} - \mathbf{x}^{[0]} - \Delta t \mathbf{v}^{[0]}\right) + \mathbf{f}\left(\mathbf{x}^{(k)}\right)$

$\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} + \Delta\mathbf{x}$

If $||\Delta\mathbf{x}||$ is small then break

$\mathbf{x}^{[1]} \leftarrow \mathbf{x}^{(k+1)}$

$\mathbf{v}^{[1]} \leftarrow \left(\mathbf{x}^{[1]} - \mathbf{x}^{[0]}\right) / \Delta t$

"Newton's Method"

Preconditioned Steepest Descent

- Mathematically, this approach is preconditioned steepest descent, in which:

$$\underbrace{\left(\frac{1}{\Delta t^2}\mathbf{M} + \mathbf{H}\right)}_{\text{preconditioner}} \Delta \mathbf{x} = \underbrace{\boxed{-\frac{1}{\Delta t^2}\mathbf{M}\left(\mathbf{x}^{(k)} - \mathbf{x}^{[0]} - \Delta t \mathbf{v}^{[0]}\right) + \mathbf{f}\left(\mathbf{x}^{(k)}\right)}}_{\text{negative gradient of } F(\mathbf{x})}$$

$$F(\mathbf{x}) = \frac{1}{2\Delta t^2}\left\|\mathbf{x} - \mathbf{x}^{[0]} - \Delta t \mathbf{v}^{[0]}\right\|_{\mathbf{M}}^2 + E(\mathbf{x})$$

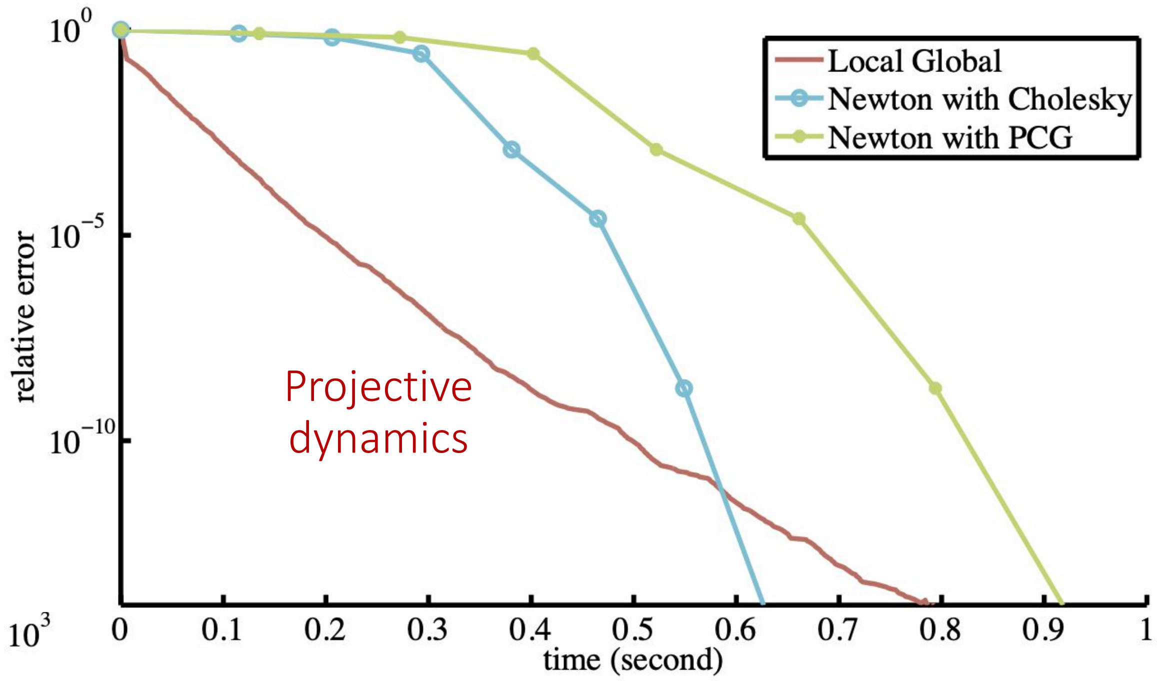
- The performance depends on how well **H** approximates the real Hessian.

内存访问耗时多

Pros and Cons of Projective Dynamics

- By building constraints into energy, the simulation now has a theoretical <u>solution</u> with physical meaning.
- Fast on CPUs with a direct solver. No more factorization!
- Fast convergence in the first few iterations.

- Slow on GPUs. (GPUs don't support direct solver wells.)
- Slow convergence over time, as it fails to consider Hessian caused by projection.
  - Still suffering from high stiffness
- Cannot easily handle constraint changes.
  - Contacts
  - Remeshing due to fracture, etc.



After-Class Reading

Bouaziz et al. 2014. *Projective Dynamics: Fusing Constraint Projections for Fast Simulation. TOG (SIGGRAPH)*.

Projective Dynamics: Fusing Constraint Projections for Fast Simulation

Sofien Bouaziz* EPFL | Sebastian Martin† VM Research | Tiantian Liu‡ University of Pennsylvania | Ladislav Kavan§ University of Pennsylvania | Mark Pauly¶ EPFL

Figure 1: *We propose a new "projection-based" implicit Euler integrator that supports a large variety of geometric constraints in a single physical simulation framework. In this example, all the elements including building, grass, tree, and clothes (49k DoFs, 43k constraints), are simulated at 3.1ms/iteration using 10 iterations per frame (see also accompanying video).*

Abstract

We present a new method for implicit time integration of physical systems. Our approach builds a bridge between nodal Finite Element methods and Position Based Dynamics, leading to a simple, efficient, robust, yet accurate solver that supports many different types of constraints. We propose specially designed energy potentials that can be solved efficiently using an alternating optimization approach. Inspired by continuum mechanics, we derive a set of continuum-based potentials that can be efficiently incorporated within our solver. We demonstrate the generality and robustness of our approach in many different applications ranging from the simulation of solids, cloths, and shells, to example-based simulation. Comparisons to Newton-based and Position Based Dynamics solvers highlight the benefits of our formulation.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: physics-based animation, implicit Euler method, position based dynamics, continuum mechanics.

Links: DL PDF

*sofien.bouaziz@epfl.ch
†sebimartin@gmail.com
‡ltt1598@gmail.com
§ladislav.kavan@gmail.com
¶mark.pauly@epfl.ch

1 Introduction

Physics-based simulation of deformable material has become an indispensable tool in many areas of computer graphics. Virtual worlds, and more recently character animations, incorporate sophisticated simulations to greatly enhance visual experience, e.g., by simulating muscles, fat, hair, clothing, or vegetation. These models are often based on finite element discretizations of continuum-mechanics formulations, allowing highly *accurate* simulation of complex nonlinear materials.

Besides realism and accuracy, a number of other criteria are also important in computer graphics applications. By *generality* we mean the ability to simulate a large spectrum of behaviors, such as different types of geometries (solids, shells, rods), different material properties, or even art-directable extensions to classic physics-based simulation. *Robustness* refers to the capability to adequately handle difficult configurations, including large deformations, degenerate geometries, and large time steps. Robustness is especially important in real-time applications where there is no "second chance" to re-run a simulation, such as in computer games or medical training simulators. The *simplicity* of a solver is often important for its practical relevance. Building on simple, easily understandable concepts – and the resulting lightweight codebases – eases the maintenance of simulators and makes them adaptable to specific application needs. *Performance* is a critical enabling criterion for realtime applications. However, performance is no less important in offline simulations, where the turnaround time for testing new scenes and simulation parameters should be minimized.

Current continuum mechanics approaches often have unfavorable trade-offs between these criteria for certain computer graphics applications, which led to the development of alternative methods, such as Position Based Dynamics (PBD). Due to its generality, simplicity, robustness, and efficiency, PBD is now implemented in a wide range of high-end products including PhysX, Havok Cloth, Maya nCloth, and Bullet. While predominantly used in realtime applications, PBD is also often used in offline simulation. However, the desirable qualities of PBD come at the cost of limited accuracy, because PBD is not rigorously derived from continuum mechanical principles.

We propose a new implicit integration solver that bridges the gap

Constrained Dynamics

Constrained Dynamics

A critical problem exists: what if constraints/forces are very very stiff? Or infinitely stiff?

$$E(\mathbf{x}) = \sum_e \frac{1}{2}k\left(\|\mathbf{x}_{ei} - \mathbf{x}_{ej}\| - L_e\right)^2 = \frac{1}{2}\boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x})\mathbf{C}^{-1}\boldsymbol{\phi}(\mathbf{x})$$

$$\mathbf{f}(\mathbf{x}) = -\nabla E = -\left(\frac{\partial E}{\partial \boldsymbol{\phi}}\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}}\right)^{\mathrm{T}} = -\mathbf{J}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{\phi} = \mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}$$

$$\boxed{\phi_e(\mathbf{x}) = \|\mathbf{x}_{ei} - \mathbf{x}_{ej}\| - L_e}$$

Compliant constraint

Let N be the number of vertices and E be the number of constraints,

$$\boldsymbol{\phi}(\mathbf{x}) \in \mathbf{R}^E$$

$$\mathbf{C} = \begin{bmatrix} 1/k & & \\ & 1/k & \\ & & \ddots \end{bmatrix} \in \mathbf{R}^{E\times E}$$

Compliant matrix

$$\mathbf{J} = \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}} \in \mathbf{R}^{E\times 3N}$$

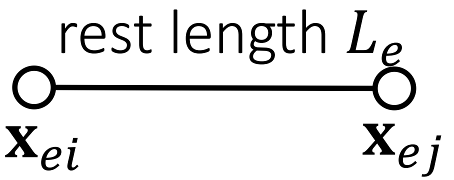
Jacobian

$$\boldsymbol{\lambda} = -\mathbf{C}^{-1}\boldsymbol{\phi} \in \mathbf{R}^E$$

Dual variables (Lagrangian multipliers)

Constrained Dynamics

A critical problem exists: what if constraints/forces are very very stiff? Or infinitely stiff?



$$\boxed{\phi_e(\mathbf{x}) = \left\|\mathbf{x}_{ei} - \mathbf{x}_{ej}\right\| - L_e}$$

Compliant constraint

$$E(\mathbf{x}) = \sum_e \frac{1}{2}k\left(\left\|\mathbf{x}_{ei} - \mathbf{x}_{ej}\right\| - L_e\right)^2 = \frac{1}{2}\boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x})\mathbf{C}^{-1}\boldsymbol{\phi}(\mathbf{x})$$

$$\mathbf{f}(\mathbf{x}) = -\nabla E = -\left(\frac{\partial E}{\partial \boldsymbol{\phi}}\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}}\right)^{\mathrm{T}} = -\mathbf{J}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{\phi} = \mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}$$

By implicit integration, we get:

$$\mathbf{M}\mathbf{v}^{\text{new}} - \Delta t\,\mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}^{\text{new}} = \mathbf{M}\mathbf{v}$$

Meanwhile,

$J = \frac{\partial \phi}{\partial x}$

$$\mathbf{C}\boldsymbol{\lambda}^{\text{new}} = -\boldsymbol{\phi}^{\text{new}} \approx -\boldsymbol{\phi} - \mathbf{J}(\mathbf{x}^{\text{new}} - \mathbf{x}) \approx -\boldsymbol{\phi} - \Delta t\mathbf{J}\,\mathbf{v}^{\text{new}}$$

$$\boxed{\begin{bmatrix} \mathbf{M} & -\Delta t\,\mathbf{J}^{\mathrm{T}} \\ \Delta t\mathbf{J} & \mathbf{C} \end{bmatrix}\begin{bmatrix} \mathbf{v}^{\text{new}} \\ \boldsymbol{\lambda}^{\text{new}} \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{v} \\ -\boldsymbol{\phi} \end{bmatrix}}$$

Constrained Dynamics

- Now we have a system with two sets of variables: the primal variable $\mathbf{x}$ (or $\mathbf{v} = \dot{\mathbf{x}}$) and the dual variable $\boldsymbol{\lambda}$.
- Method 1: We can solve the two variables by a direct solver together, in a primal-dual fashion:

$$\begin{bmatrix} \mathbf{M} & -\Delta t\ \mathbf{J}^{\mathrm{T}} \\ \Delta t\mathbf{J} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{\mathrm{new}} \\ \boldsymbol{\lambda}^{\mathrm{new}} \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{v} \\ -\boldsymbol{\phi} \end{bmatrix}$$

- Method 2: We can reduce the system by Schur complement and solve $\boldsymbol{\lambda}^{\mathrm{new}}$ first.

$$(\Delta t^2 \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}} + \mathbf{C})\boldsymbol{\lambda}^{\mathrm{new}} =- \boldsymbol{\phi} - \Delta t\mathbf{J}\mathbf{v}$$

$$\mathbf{v}^{\mathrm{new}} \longleftarrow \mathbf{v} +- \Delta t\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}^{\mathrm{new}}$$

- Infinite stiffness? $\mathbf{C} \longrightarrow \mathbf{0}$.

Constrained Dynamics

- Articulated Rigid Bodies (ragdoll animation)

Stable Constrained Dynamics

From a mass-spring system, we know spring Hessian (tangent stiffness) is:

$$\mathbf{H}(\mathbf{x}) = \sum_{e=\{i,j\}} \begin{bmatrix} \mathbf{H}_e & -\mathbf{H}_e \\ -\mathbf{H}_e & \mathbf{H}_e \end{bmatrix} \qquad \mathbf{H}_e = \underbrace{\boxed{k\frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|^2}}}_{\text{material stiffness}} + \underbrace{\boxed{k\left(1-\frac{L}{\|\mathbf{x}_{ij}\|}\right)\left(\mathbf{I}-\frac{\mathbf{x}_{ij}\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|^2}\right)}}_{\text{geometric stiffness}}$$

According to constrained dynamics: $\mathbf{f}(\mathbf{x}) = \mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda}$ and $\boldsymbol{\lambda} = -\mathbf{C}^{-1}\boldsymbol{\phi}$, so:

$$\mathbf{H}(\mathbf{x}) = -\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \mathbf{J}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{J} - \frac{\partial \mathbf{J}^{\mathrm{T}}}{\partial \mathbf{x}}\boldsymbol{\lambda} = \sum_{e=\{i,j\}} \boxed{k\mathbf{J}_e^{\mathrm{T}}\mathbf{J}_e} + \sum_{e=\{i,j\}} \boxed{\frac{\partial \mathbf{J}_e^{\mathrm{T}}}{\partial \mathbf{x}}\lambda_e}$$

$$\mathbf{J}_e = \frac{\partial \phi_e}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|} & -\frac{\mathbf{x}_{ij}^{\mathrm{T}}}{\|\mathbf{x}_{ij}\|} \end{bmatrix}$$

Stable Constrained Dynamics

According Lecture 5, Page 16, implicit integration is:

$$\left(\frac{1}{\Delta t^2}\mathbf{M} + \mathbf{H}\left(\mathbf{x}^{[0]}\right)\right)\Delta\mathbf{x} = \frac{1}{\Delta t^2}\mathbf{M}\left(\Delta t\mathbf{v}^{[0]}\right) + \mathbf{f}\left(\mathbf{x}^{[0]}\right)$$

$$\Downarrow$$

$$\left(\mathbf{M} + \Delta t^2\mathbf{H}\left(\mathbf{x}^{[0]}\right)\right)\mathbf{v}^{\text{new}} = \mathbf{M}\mathbf{v}^{[0]} + \Delta t\mathbf{f}\left(\mathbf{x}^{[0]}\right)$$

But implicit integration with constrained dynamics is:

$$\begin{bmatrix} \mathbf{M} & -\Delta t\,\mathbf{J}^{\mathrm{T}} \\ \Delta t\mathbf{J} & \mathbf{C} \end{bmatrix}\begin{bmatrix} \mathbf{v}^{\text{new}} \\ \boldsymbol{\lambda}^{\text{new}} \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{v} \\ -\boldsymbol{\phi} \end{bmatrix}$$

$$\Downarrow$$

$$(\mathbf{M} + \Delta t^2\boxed{\mathbf{J}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{J}})\mathbf{v}^{\text{new}} = \mathbf{M}\mathbf{v} - \Delta t\,\mathbf{J}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{\phi} = \mathbf{M}\mathbf{v} + \Delta t\mathbf{f}$$

material stiffness

Missing geometric stiffness matrix here…

After-Class Reading (optional)

$$\begin{bmatrix} \mathbf{M} \overset{\text{geometric stiffness}}{\boxed{- \Delta t^2 \frac{\partial \mathbf{J}^T}{\partial \mathbf{x}} \boldsymbol{\lambda}}} & -\Delta t\, \mathbf{J}^T \\ \Delta t \mathbf{J} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{\text{new}} \\ \boldsymbol{\lambda}^{\text{new}} \end{bmatrix} = \begin{bmatrix} \mathbf{Mv} \\ -\boldsymbol{\phi} \end{bmatrix}$$

Tournier et al. 2015. *Stable Constrained Dynamics. TOG (SIGGRAPH)*.

Stable Constrained Dynamics

Maxime Tournier [4,1,2] Matthieu Nesme [1,3] Benjamin Gilles [2,1] François Faure [5,3,1]

[1] INRIA [2] LIRMM-CNRS [3] LJK-CNRS [4] RIKEN BSI-BTCC [5] Univ. Grenoble

Figure 1: *Our method improves stability and step size for the simulation of constraint-based objects subject to high tensile forces, isolated or coupled with other types of objects. Bow: stiff 3D frame, 1D inextensible string, rigid arrow ; Trampoline: soft lateral springs, inextensible textile ; Knee: complex assembly of rigid bodies and stiff unilateral springs ; Ragdoll: rigid body assembly.*

Abstract

We present a unification of the two main approaches to simulate deformable solids, namely elasticity and constraints. Elasticity accurately handles soft to moderately stiff objects, but becomes numerically hard as stiffness increases. Constraints efficiently handle high stiffness, but when integrated in time they can suffer from instabilities in the nullspace directions, generating spurious transverse vibrations when pulling hard on thin inextensible objects or articulated rigid bodies. We show that geometric stiffness, the tensor encoding the change of force directions (as opposed to intensities) in response to a change of positions, is the missing piece between the two approaches. This previously neglected stiffness term is easy to implement and dramatically improves the stability of inextensible objects and articulated chains, without adding artificial bending forces. This allows time step increases up to several orders of magnitude using standard linear solvers.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—[Physically based modeling] I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—[Animation]

Keywords: Physically based animation, Simulation, Dynamics, Constraints, Continuum mechanics, Geometric Stiffness

1 Introduction

Constraint-based simulation is very popular for implementing joints in articulated rigid bodies, and to enforce inextensibility in some directions of deformable objects such as cables or cloth. Its mathematical formulation makes it numerically robust to infinite stiffness, contrary to elasticity-based simulation, and some compliance can be introduced in the formulation or obtained through approximate solutions. Unfortunately, when the constraint forces are large, constraint-based objects are prone to instabilities in the transverse, unconstrained directions. This occurs when pulling hard on inextensible strings and sheets, or on chains of articulated bodies. The spurious vibrations can lead to unrealistic behaviors or even simulation divergence. They can be avoided using small time steps or complex non-linear solvers, however this dramatically slows down the simulation, while many applications, especially in interactive simulation, hardly allow for one linear solution per frame. The simulation speed can only be maintained by relaxing inextensibility, or using implicit elastic bending forces, however this changes the constitutive law of the simulated objects.

In this work, we show how to perform stable and efficient simulations of both extensible and inextensible constraint-based objects subject to high tensile forces. The key to transverse stability lies in the *geometric stiffness*, a first-order approximation of the change of direction of the internal forces due to rotation or bending. Neglecting the geometric stiffness, as usually done in constraint-based simulation, is a simplification of the linearized equation system, which in turn is a simplification of the exact, non-linear implicit integration. In case of thin objects, this leaves the transverse directions unconstrained, leading to uncontrolled extensions after time integration, introducing artificial potential energy. While this is acceptable for small stiffnesses or short time steps, this may introduce instabilities in the other cases. In this paper, we show that solving the complete linear equation allows high stiffnesses and large time steps which were only achievable using much slower non-linear solvers before. We show how to handle the geometric stiffness in a numerically stable way, even for very large material stiffness. The implementation is easy to combine with existing implicit solvers, and can provide several orders of magnitude speed-ups. Moreover, it allows a unification of rigid body and continuum mechanics.

In the next section, we detail our background and motivation through an introductory example. The principle of our method is then explained in Section 3. Its application to a wide variety of cases is then presented in Section 4. We conclude and sketch future work in Section 5.

A Summary For the Day

- Position-based dynamics and strain limiting
  - The key is to build a projection function for every constraint.
  - Two approaches for integration: Jacobi and Gauss-Seidel.
  - Fast in low resolutions, but problematic in high resolutions.
  - Not physically correct.
- Projective Dynamics
  - Also uses projection functions, but they are now built into energies.
  - In every iteration, projections are first updated, and then treated as constants in implicit formulation.
  - The matrix in the system becomes constant, can be pre-factorized for fast simulation.
  - Converges fast only in the first few iterations, slow afterwards. CPU friendly.
- Constrained Dynamics
  - Focused on very stiff constraints. Introduces dual variables.
  - Also built upon implicit integration. Two methods: primal-dual, pure dual.
  - Restrictions on the solvers.