

Position Based Dynamics

A fast yet physically plausible method for deformable body simulation

Tiantian Liu

GAMES Webinar

03/28/2019

Microsoft®
Research
微软亚洲研究院

Position Based Dynamics

A **fast** yet **physically plausible** method for deformable body simulation

Tiantian Liu

GAMES Webinar

03/28/2019

Microsoft®
Research
微软亚洲研究院

Intended Takeaway from this Talk...

- For Rookies...
 - Basic idea of a deformable body simulation pipeline
 - What is Position Based Dynamics (PBD)
 - How to implement the basic building blocks of PBD
- For Veterans...
 - A physically correct understanding of PBD
 - Insights and potential improvements of PBD

Major References of this Talk

3rd Workshop in Virtual Reality Interactions and Physical Simulation "VRPHYS" (2006)
C. Mendonça, I. Nazaro (Editors)

Position Based Dynamics

Mathias Müller Bruno Heidelberger Marcus Hennig John Ratcliff

AGEA

Abstract

The most popular approaches for the simulation of dynamic systems in computer graphics are force based. Internal and external forces are accumulated from which accelerations are computed based on Newton's second law of motion. A time integration method is then used to update the velocities and finally the positions of the object. A few simulation methods (most rigid body simulators) use impulse based dynamics and directly manipulate velocities. In this paper we present an approach which omits the velocity layer as well and immediately works on the positions. The main advantage of a position based approach is its consistency. Overlapping problems of explicit integration schemes in force based systems can be avoided. In addition, collision constraints can be handled easily and penetrations can be resolved completely by projecting points to valid locations. We have used the approach to build a real time cloth simulator which is part of a physics software library for games. This application demonstrates the strengths and benefits of the method.

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

1. Introduction

Research in the field of physically based animation in computer graphics is concerned with finding new methods for the simulation of physical phenomena such as the dynamics of rigid bodies, deformable objects or fluid flow. In contrast to computational systems where the main focus is on accuracy, the main issues here are stability, robustness and speed while the results should remain visually plausible. Therefore, existing methods from computational sciences can not be adopted one to one. In fact, the main justification for doing research on physically based simulation in computer graphics is to come up with specialized methods, tailored to the particular needs in the field. The method we present falls into this category.

The traditional approach to simulating dynamic objects has been to work with forces. At the beginning of each time step, internal and external forces are accumulated. Examples of internal forces are elastic forces in deformable objects or viscosity and pressure forces in fluids. Gravity and collision forces are examples of external forces. Newton's second law of motion relates forces to accelerations via the mass. So instead

of the density or lumped masses of vertices, the forces are transformed into accelerations. Any time integration scheme can then be used to first compute the velocities from the accelerations and then the positions from the velocities. Such approaches use impulses instead of forces to control the motion. Because impulses directly change velocities, level of integration can be skipped.

In computer graphics and especially in computer game it is often desirable to have direct control over position objects or vertices of a mesh. The user might want to attach a vertex to a kinematic object or make sure the vertex always stays outside a colliding object. The method we propose works directly on positions which makes such manipulations easy. In addition, with the position based approach possible to control the integration directly thereby avoid overshooting and energy gain problems in connection with explicit integration. So the main features and advantage position based dynamics are:

- Position based simulation gives control over explicit integration and removes the typical instability problems.

XPBD: Position-Based Simulation of Compliant Constrained Dynamics

Miles Macklin Matthias Müller Ntaspang Chentanez

NVIDIA

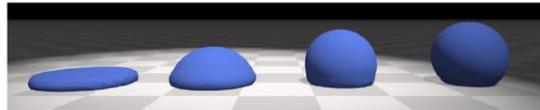


Figure 1: In this example, we see the effect of changing the relative stiffness of volume conservation and stretch and shear constraints on a deformable body. Unlike traditional PBD, our method allows users to control the stiffness of deformable bodies in a time step and iteration count independent manner,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

Categories and Subject Descriptors: I.3.7 [Computer Graphics]: Interactive Simulation; I.3.7 [Computer Graphics]: Real-time simulation; I.3.7 [Computer Graphics]: Animation

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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. © 2016 Copyright held by the owner(s). Publication rights licensed to ACM. SIGGRAPH '16, October 10–12, 2016, Burlingame, CA, USA. ISBN: 978-1-4503-4092-7/16/00. DOI: http://dx.doi.org/10.1145/294258.294272

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 nearby 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 that 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 problem 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.

Efficient Simulation of Inextensible Cloth

Rony Goldenthal^{1,2} David Harman¹ Raanan Fattal¹ Michel Bercovier² Eitan Grinspan¹
¹Columbia University ²The Hebrew University of Jerusalem ³University of California, Berkeley

Abstract

Many textiles do not noticeably stretch under their own weight. Unfortunately, for better performance many cloth solvers disregard this fact. We propose a method to obtain very low strains along the warp and weft direction using Constrained Lagrangian Mechanics and a novel fast projection method. The resulting algorithm acts as a velocity filter that easily integrates into existing simulation code.



Figure 1: Importance of capturing inextensibility. For efficiency many simulation methods allow 10% or more strain, whereas many fabrics do not visibly stretch. A 1m² patch pinned at two corners 1m apart, is allowed to relax under gravity. We compare (left to right) three simulations of progressively smaller permissible strain with an actual denim patch.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.5 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: Physically-based Modeling, Cloth simulation, Constrained Lagrangian Mechanics, Constraints, Stretching, Inextensibility, Isometry

1 Introduction

Our eyes are very sensitive to the behavior of fabrics, to the extent that we can identify the kind of fabric simply from its shape and motion [Goffills and Kulkarni 2002]. One important fact is that most fabrics do not stretch under their own weight. Unfortunately, for many popular cloth solvers, a reduction of permissible stretching is synonymous with degradation in performance: for tractable simulation times one may settle for an unrealistic 10% or more strain (compare 1% and 10%, Figure 1). Our work alleviates this problem by introducing a numerical solver that excels at time-stepping quasi-inextensible surfaces (stretching below 1%).

The solver builds on a framework of Constrained Lagrangian Mechanics (CLM) [Marsden 1999]. Warp and weft, the perpendicular sets of strands that make up a textile, are prohibited from stretching by enforcing constraint equations, not by integrating spring forces. We present numerical evidence supporting the observation that a constraint-based method is inherently well-suited to operate in the quasi-inextensible regime. In contrast, for this regime spring-based methods are known to experience a range of difficulties, leading to the adoption of various strain limiting [Pruess 1995] and strain rate-limiting algorithms.

We are motivated by the work of Bridson et al. [2002], who viewed strain limiting as one of multiple velocity filtering passes (another being collision handling). The velocity filter paradigm enables the design of modular systems with mix-and-match flexibility.

Contributions We propose a novel CLM formulation that is implicit on the constraint gradient (§4.1). We prove that the implicit method's nonlinear equations correspond to a minimization problem (§4.2); this result motivates a fast projection method for enforcing inextensibility (§4.3). We describe an implementation of fast projection as a simple and efficient velocity filter, as part of a framework that decouples time-stepping, inextensibility, and collision passes (§4.4). Consequently, this fast projection method easily incorporates with a code's existing bending, damping, and collision models, to yield accelerated performance (§5).

Before discussing these contributions, we summarize the relevant literature (§2) and describe the basic discrete cloth model (§3).

2 Related Work

For brevity, we review work on stretch resistance; for broad surveys on cloth simulation see [Houze and Breen 2000; Choi and Ko 2005].

The most general approach is to treat cloth as an elastic material [Terzopoulos et al. 1987; Breen et al. 1994; Ebelhardt et al. 1996; Baraff and Witkin 1998; Choi and Ko 2002]. To reduce visible stretching, elastic models typically adopt large elastic moduli or stiff springs, degrading numerical stability [Hsu et al. 2003].

To address the stiffness of the resulting differential equations, Baraff and Witkin [1998] proposed implicit integration, allowing for large, stable time-steps; adaptive time-stepping was required to prevent collision handling. Ebelhardt [2000] and Boxerman et al. [2003] adopted implicit-explicit (IMEX) formulations, which treat only a subset of forces implicitly. Our method is closely related to the IMEX approach, in the sense that stretching forces are singled out for special treatment.

These works, and many of their sequels, improved performance by allowing some perceptible stretch of the fabric. In the quasi-inextensible regime, however, implicit methods encounter numerical limitations [Volino and Magnien-Thalmann 2001; Boxerman 2003; Haath et al. 2003]; the condition number of the implicit system grows with the elastic material stiffness, forcing iterative solvers to perform many iterations, additionally, time-stepping algorithms such as Backward Euler and IMEX² introduce undesirable numerical damping when the system is stiff [Boxerman 2003].

Stable Constrained Dynamics

Maxime Tournier^{4,1,2} Matthieu Neume^{1,3} Benjamin Gilles^{2,1} François Faure^{5,3,1}
¹INRIA ²LIRMM-CNRS ³LJK-CNRS ⁴RKEN ISI-BTCC ⁵Univ. 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. Base: stiff 3D plane; 1:2 inextensible string, rigid arms; 3: Translucent: soft lateral springs, inextensible textile; 4: Knee: complex assembly of rigid bodies and stiff unilateral springs; 5: Rigid: 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 with much slower non-linear solvers before. We show how to solve several orders of magnitude speed-ups. Moreover, it allows 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.

© The Eurographics Association 2006.

[Müller et al. 2007]

[Macklin et al. 2016]

[Goldenthal et al. 2007]

[Tournier et al. 2015]

Computer Graphics (Past and Now)



1986

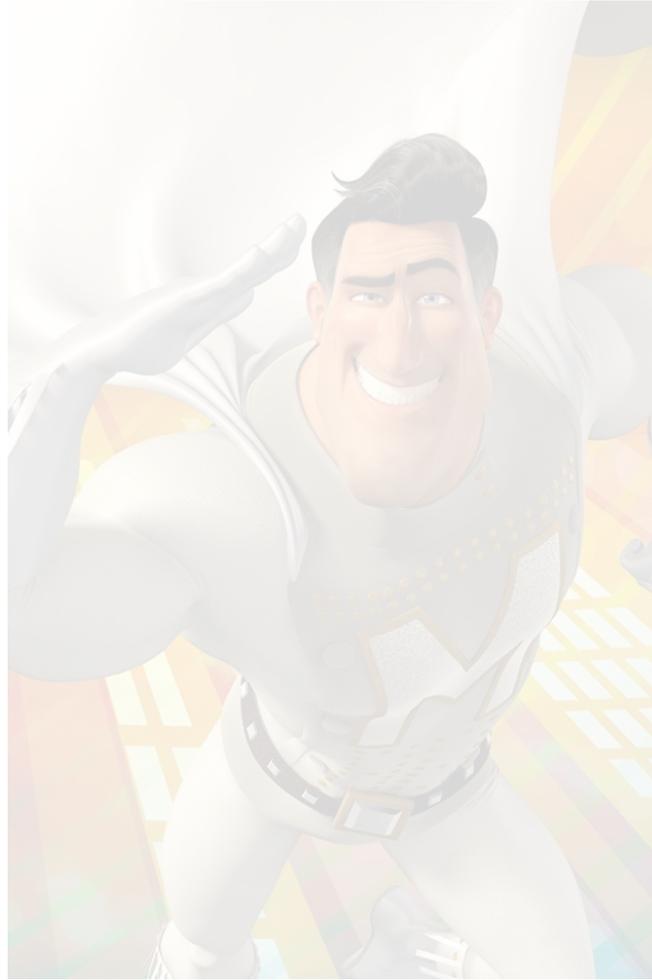


2017

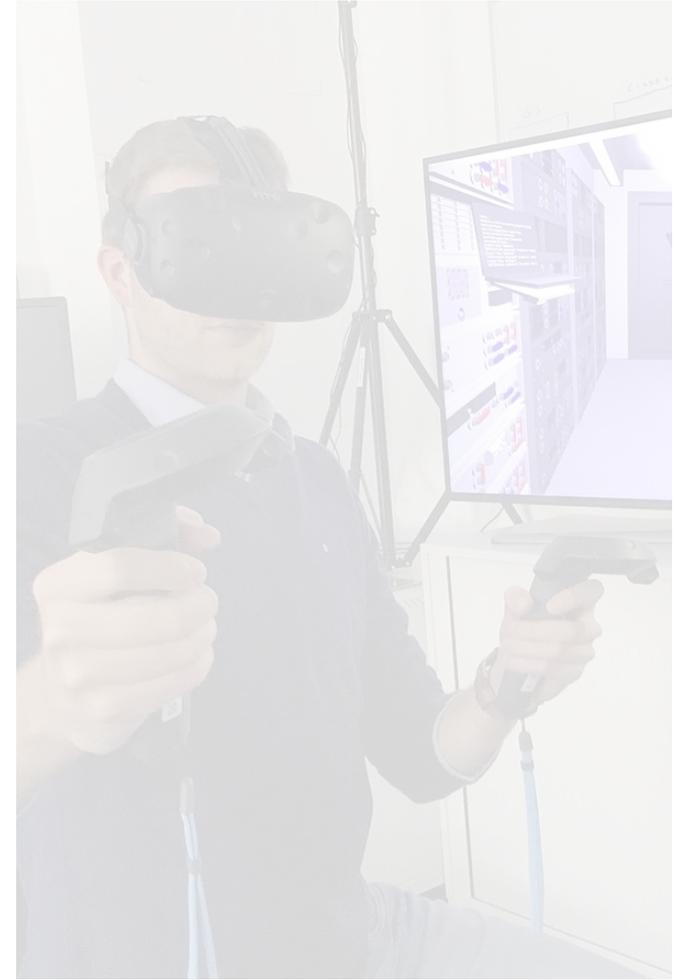
Deformable Body Simulation



Game



Movie/Animation



AR/VR

Deformable Body Simulation



Game



Movie/Animation

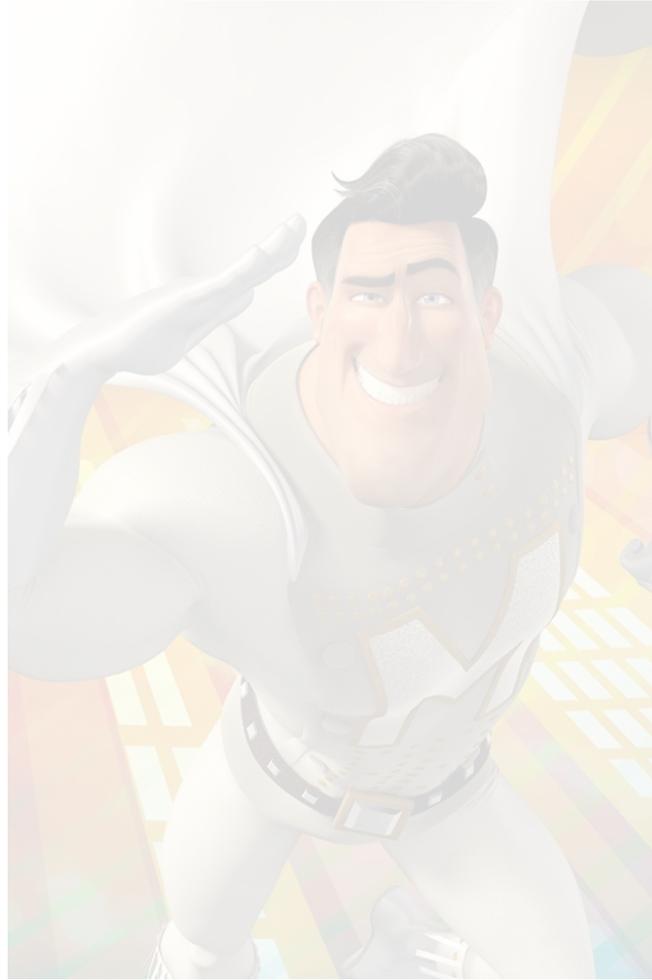


AR/VR

Deformable Body Simulation



Game



Movie/Animation

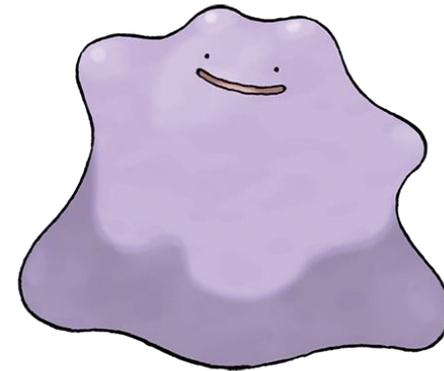


AR/VR

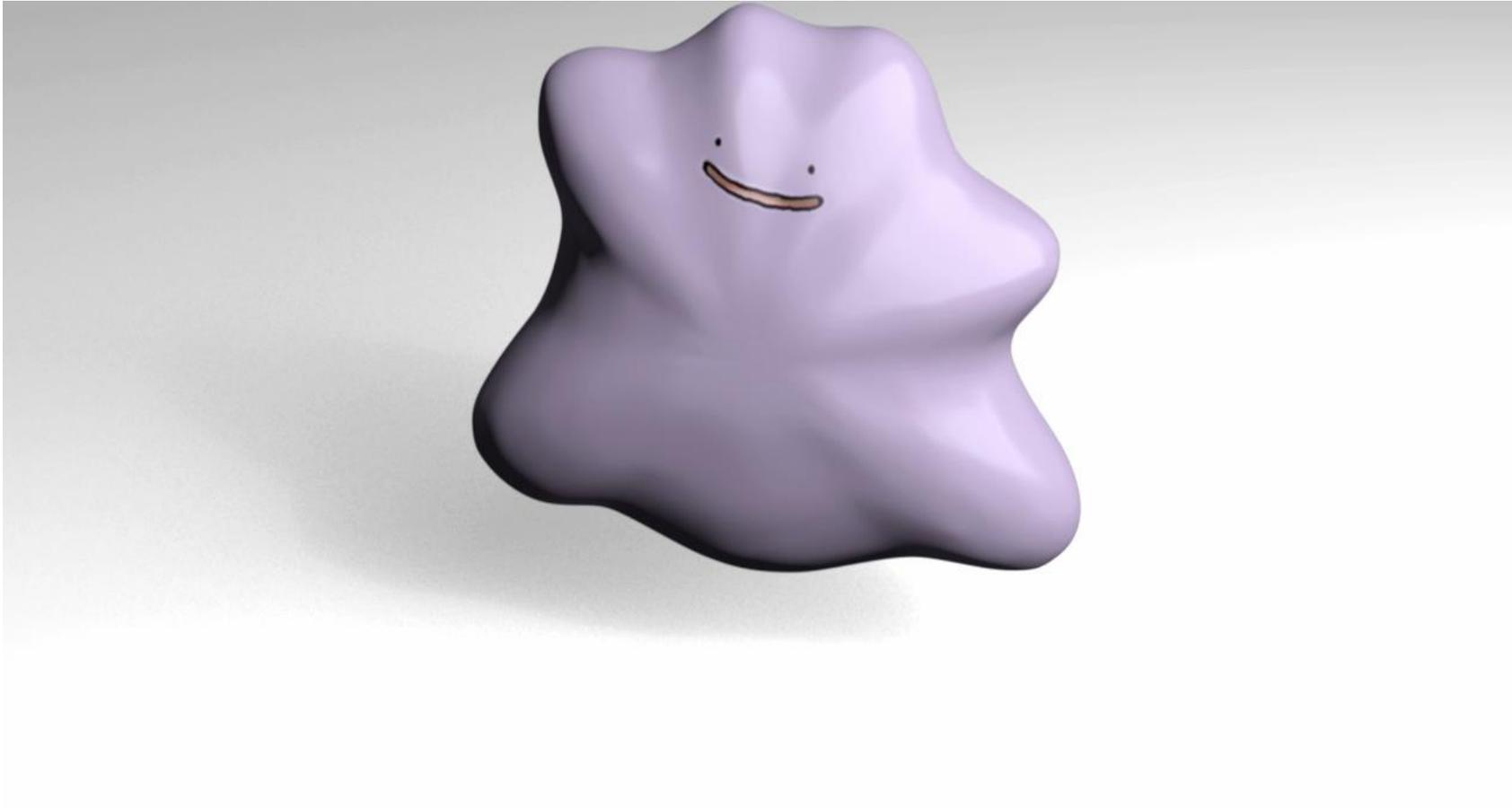
Rigid Body

v.s.

Deformable Body

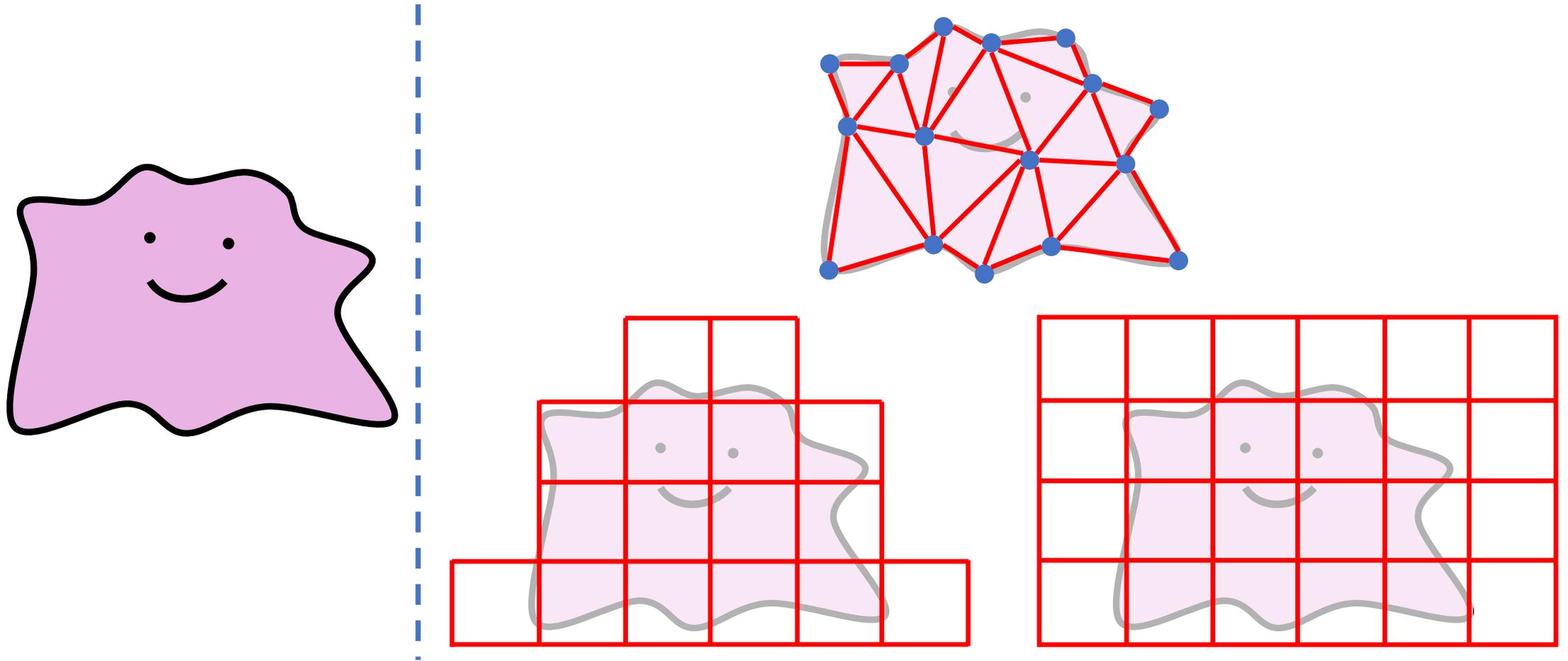


Deformable Body Simulation

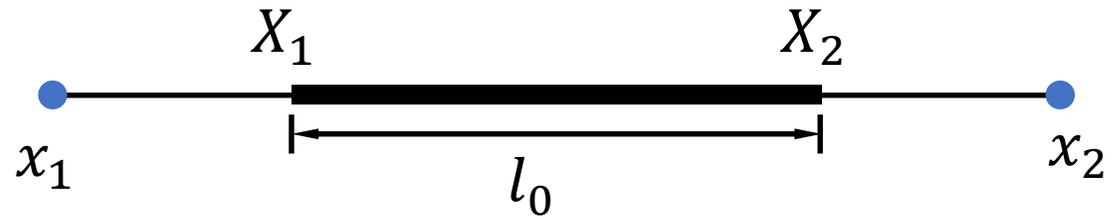
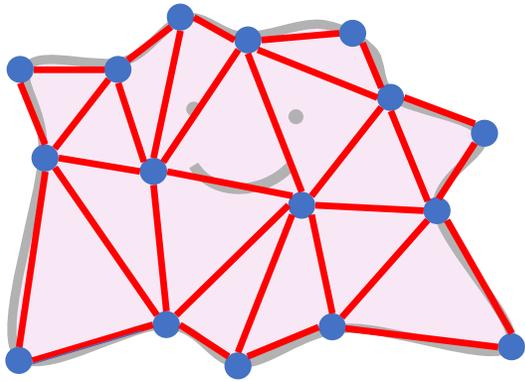


[Liu et al. 2017]

Spatial Discretization

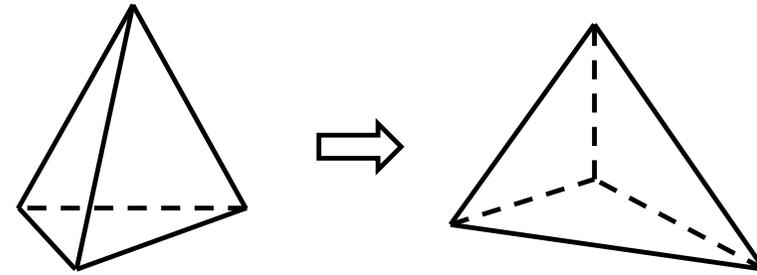
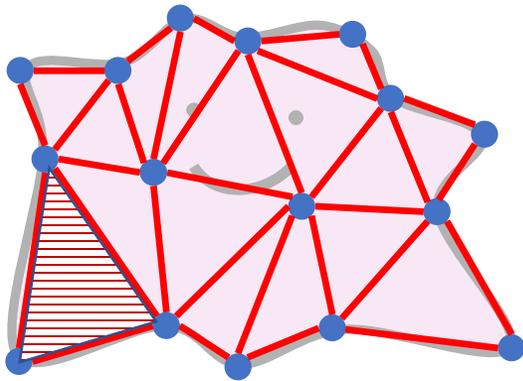


Representation of a Deformable Body



$$E(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{2} k (\|\mathbf{x}_1 - \mathbf{x}_2\| - l_0)^2$$

Representation of a Deformable Body

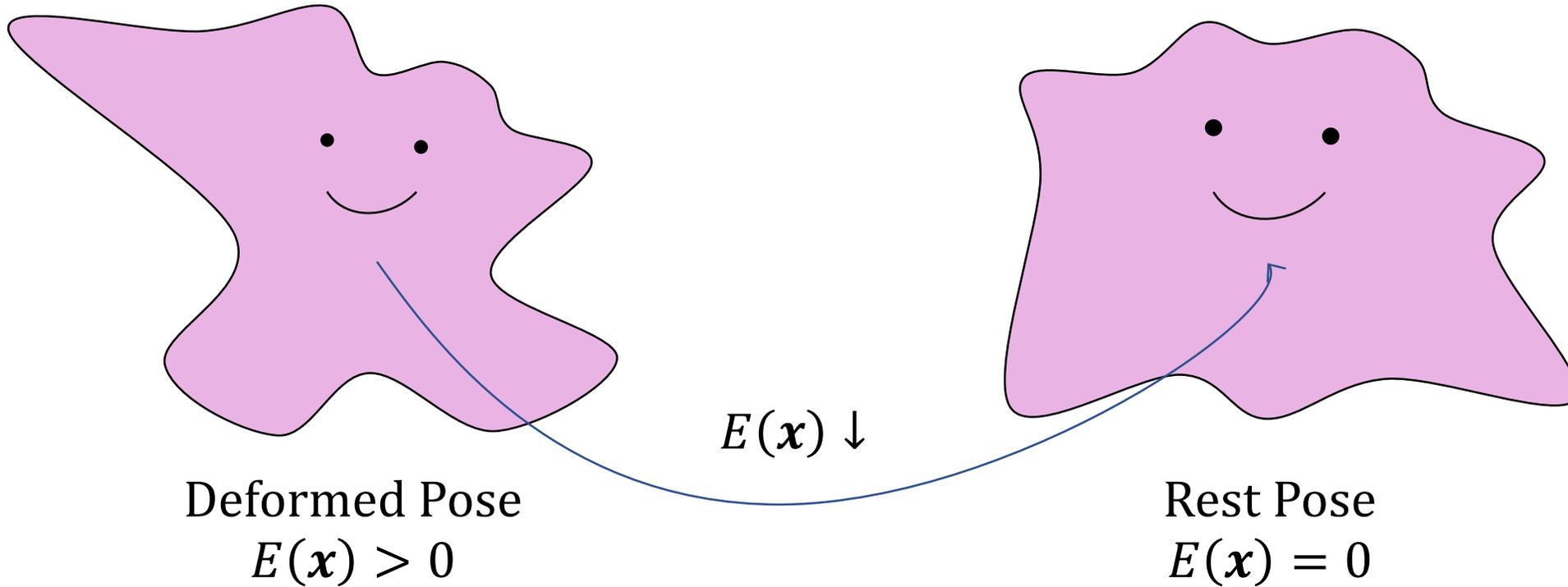


$$\Psi(\mathbf{F}(\mathbf{x})) = \mu \|\boldsymbol{\epsilon}(\mathbf{F})\|_F^2 + \frac{1}{2} \lambda \text{tr}^2(\boldsymbol{\epsilon}(\mathbf{F}))$$

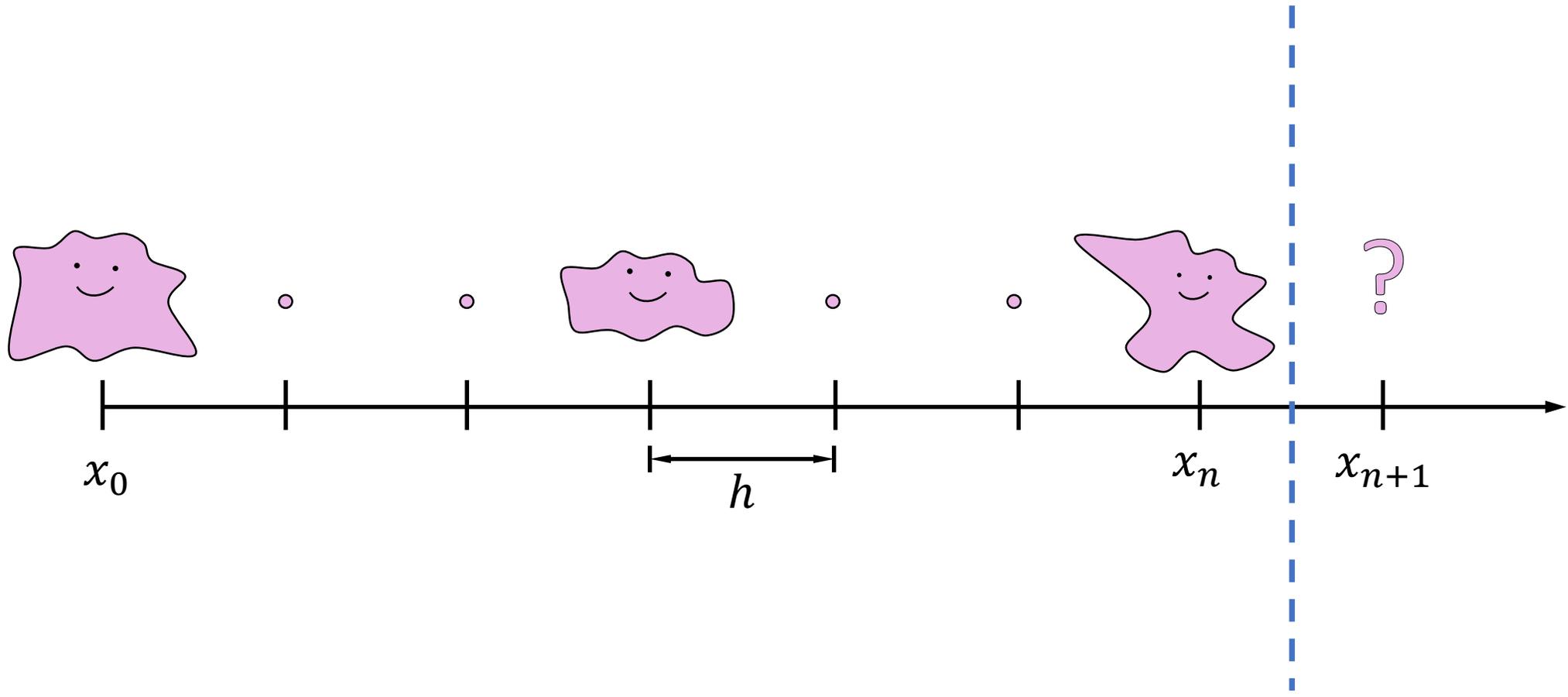
Energy Density
Deformation Gradient

Strain Tensor

Elastic Energy



Temporal Discretization



Newton's 2nd Law of Motion

- $x_{n+1} = x_n + \int_{t_n}^{t_n+h} v(t) dt$
- $v_{n+1} = v_n + \int_{t_n}^{t_n+h} \underbrace{M^{-1}(f_{int}(x(t)) + f_{ext})}_{a(t)} dt$

Time integration: Implicit Euler

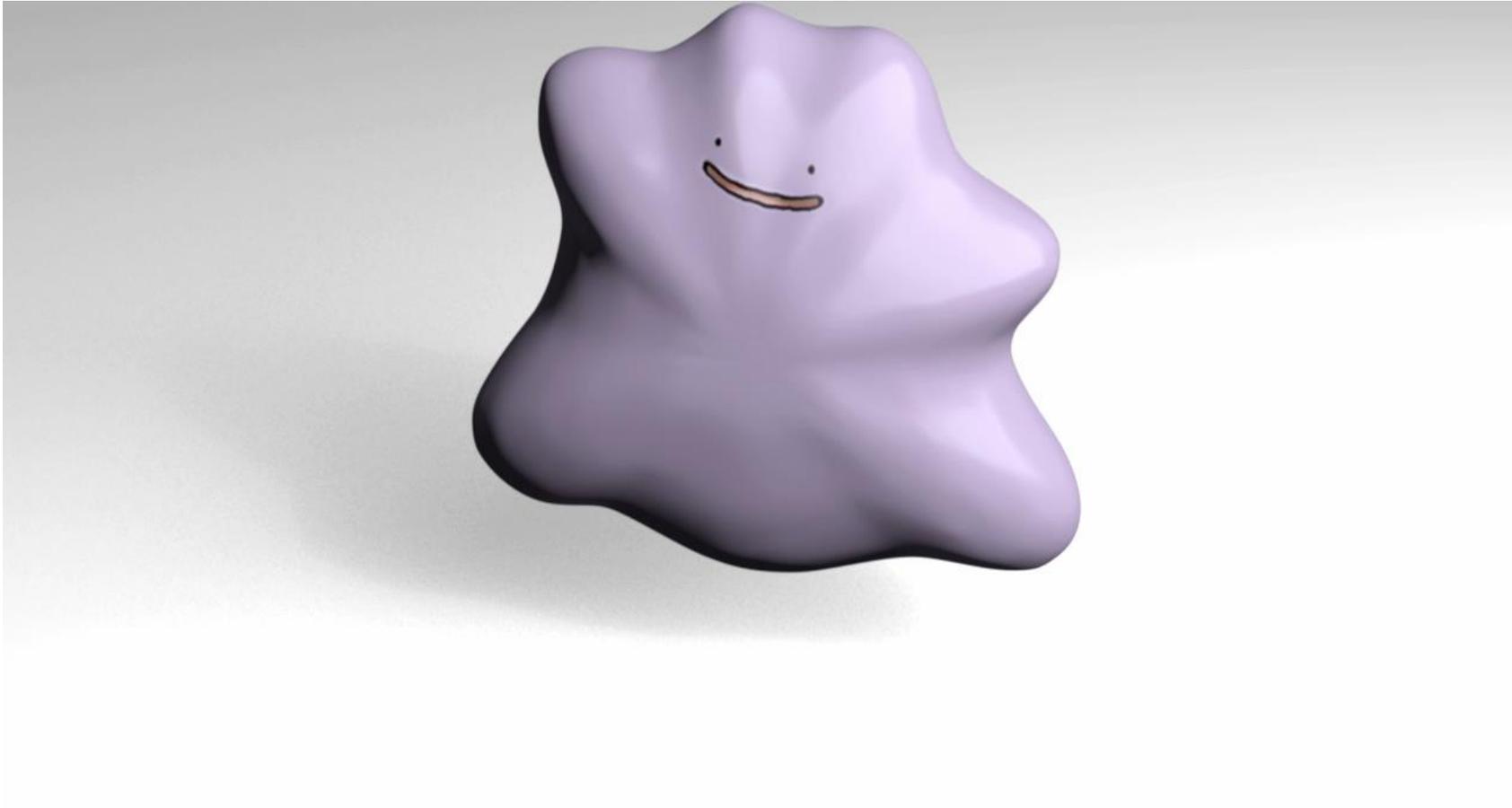
- $x_{n+1} = x_n + hv_{n+1}$
- $v_{n+1} = v_n + hM^{-1}(f_{int}(x_{n+1}) + f_{ext})$
- $\underline{x_{n+1}} = x_n + \underbrace{hv_n + h^2M^{-1}f_{ext}}_y + h^2M^{-1}f_{int}(\underline{x_{n+1}})$

Variational Implicit Euler

- $x_{n+1} = \operatorname{argmin}_x \underbrace{\frac{1}{2h^2} \|x - y\|_M}_{\text{inertia}} + \underbrace{E(x)}_{\text{elasticity}}$

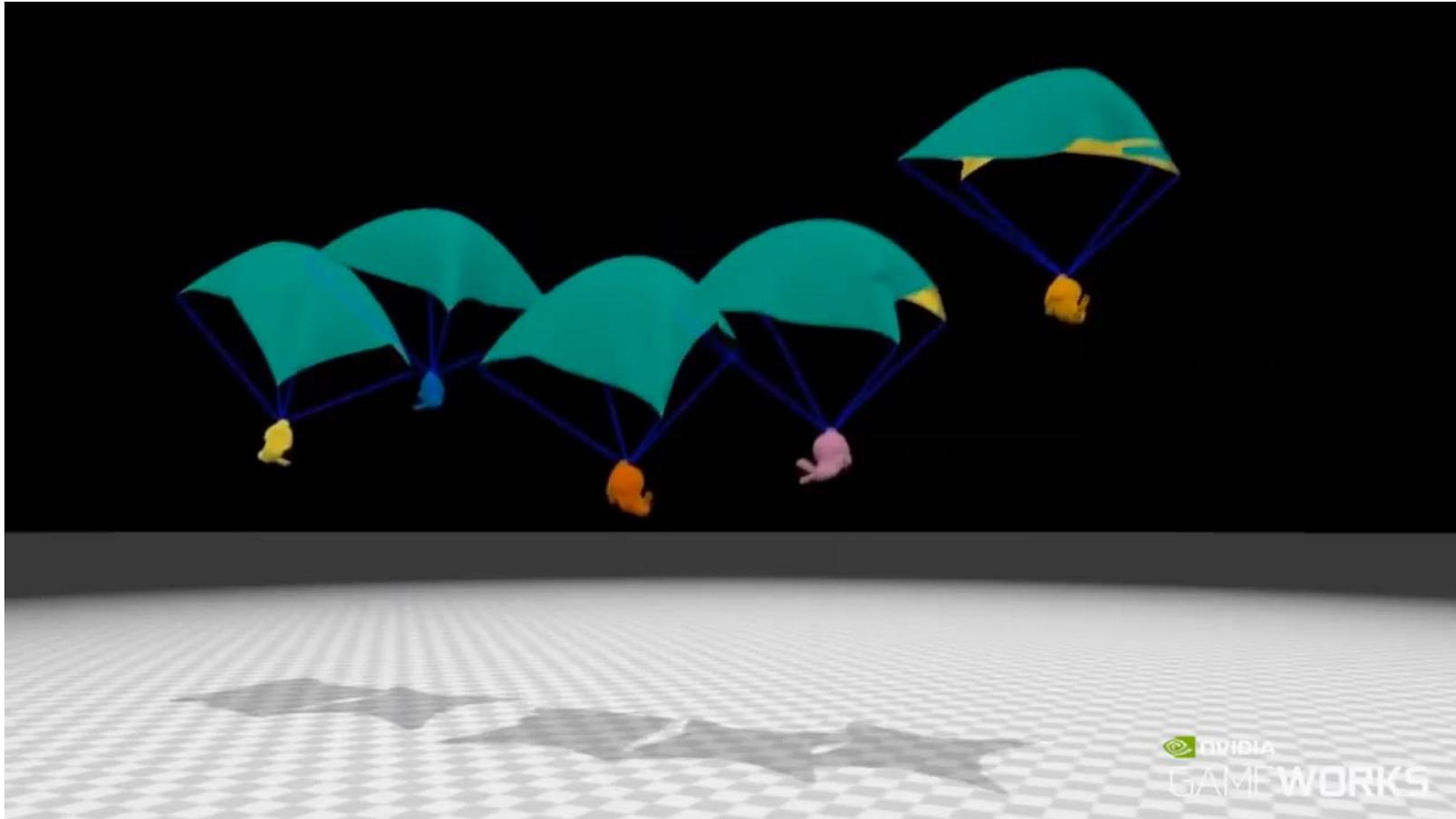
- Pick your favorite optimization tool to solve
 - Gradient Descent / Newton / Quasi-Newton etc...

Deformable Body Simulation



[Liu et al. 2017]

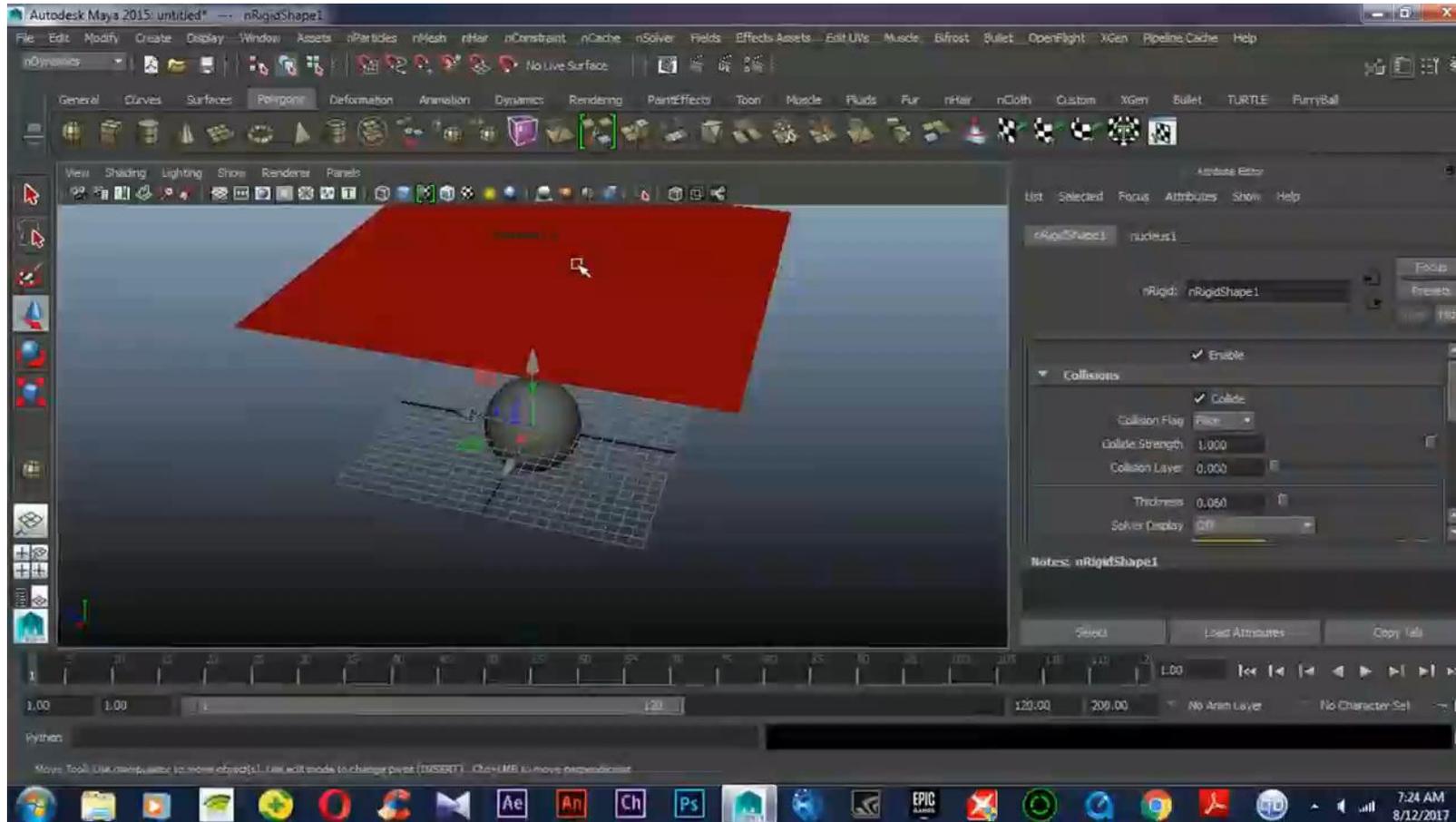
State-of-the-Art Real-time Simulators



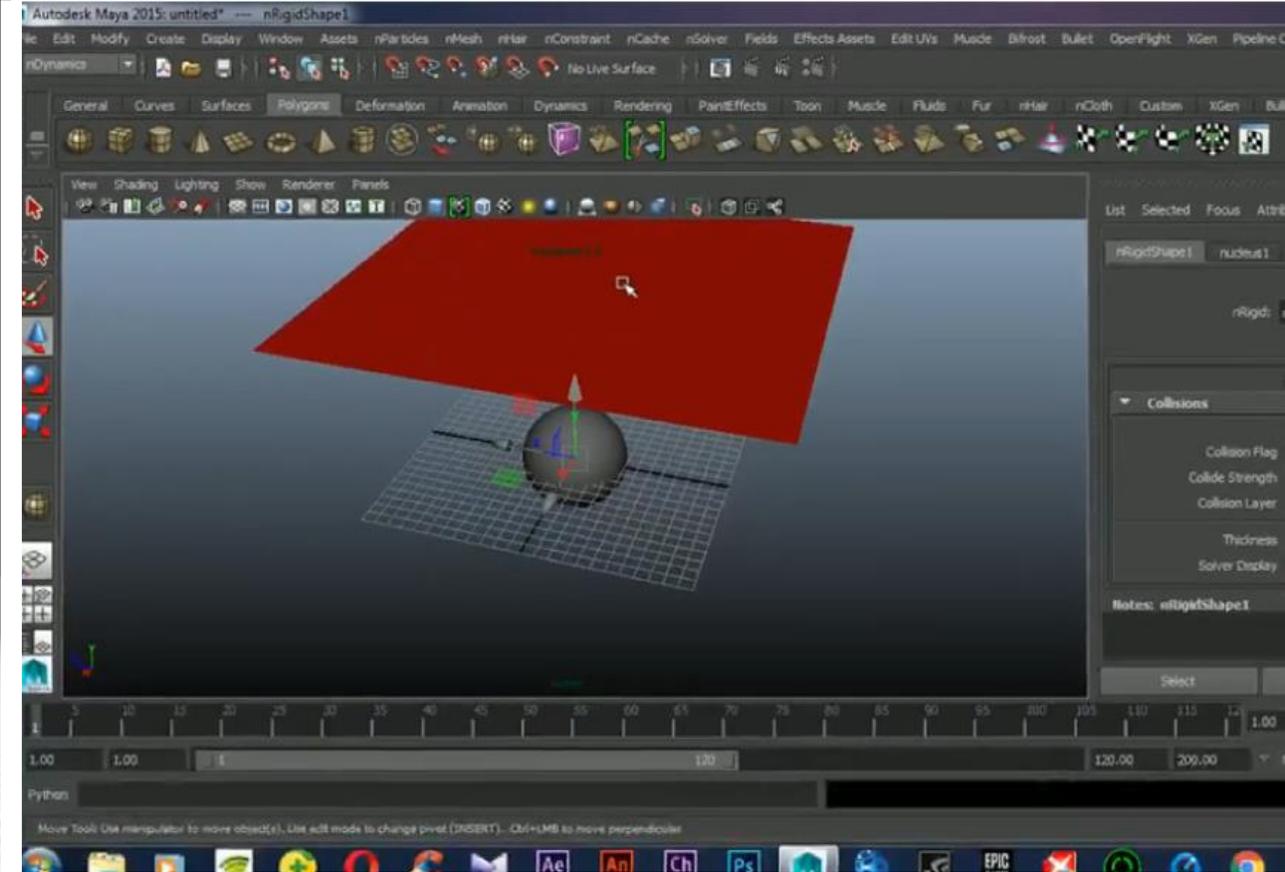
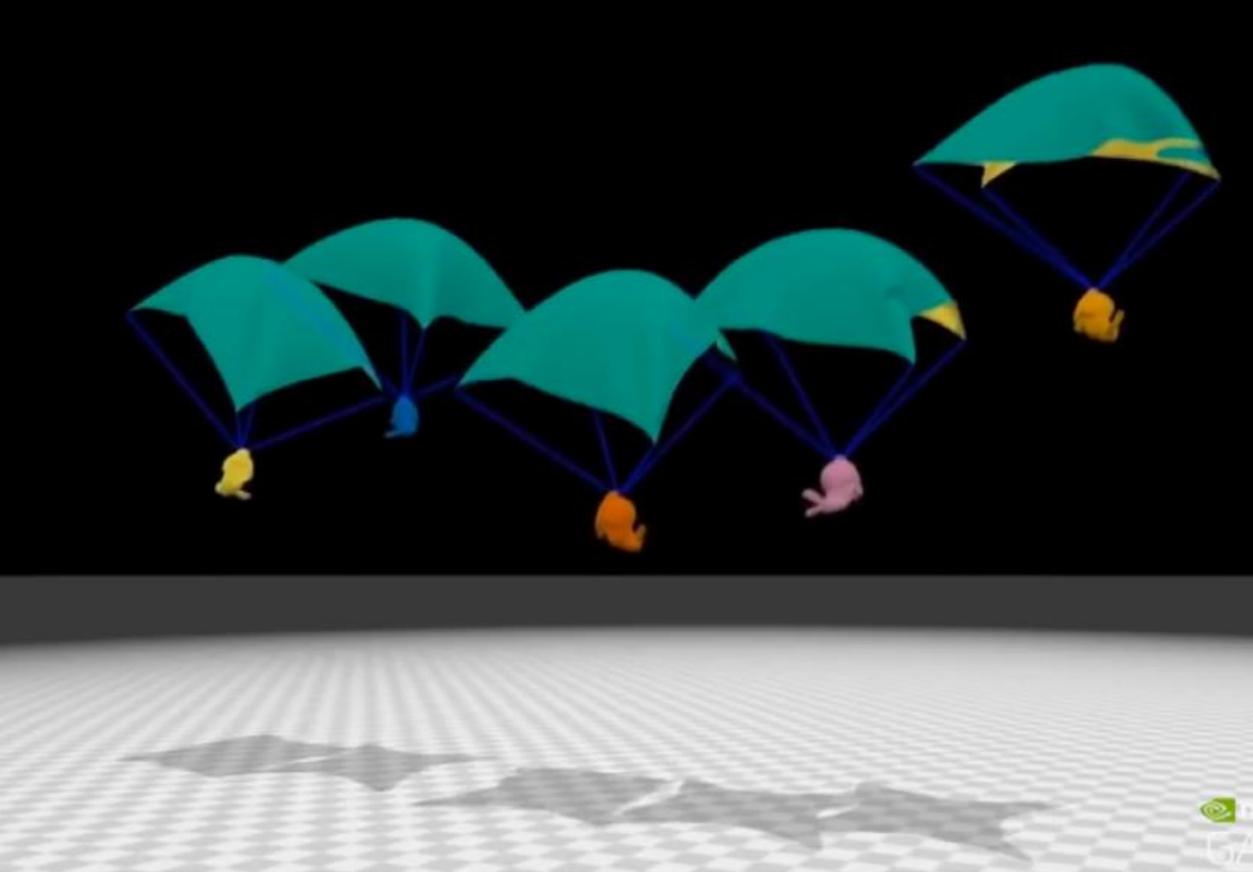
NVIDIA Flex

[Video courtesy of NVIDIA]

State-of-the-Art Real-time Simulators

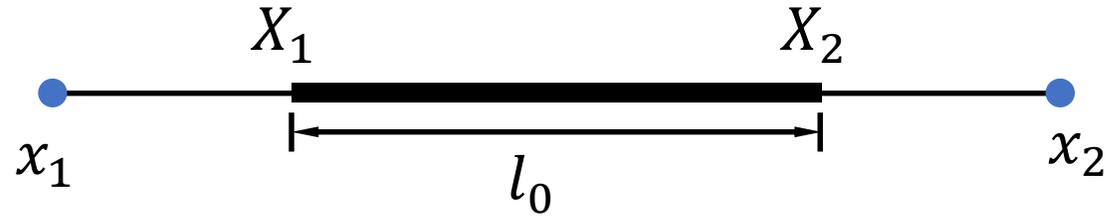
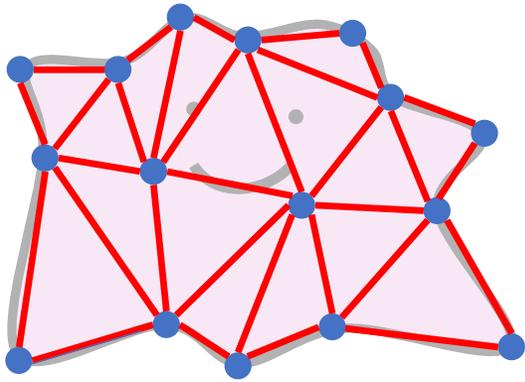


Maya nDynamics [Video courtesy of Pixclue Studios]



Position Based Dynamics

Position Based Dynamics



$$c(\mathbf{x}_1, \mathbf{x}_2) = \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|}{l_0} - 1$$

Goal: $c(\mathbf{x}_1, \mathbf{x}_2) = 0$

PBD: Pipeline

```
(5)  forall vertices  $i$  do  $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$ 
(6)  dampVelocities( $\mathbf{v}_1, \dots, \mathbf{v}_N$ )
(7)  forall vertices  $i$  do  $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$ 
(8)  forall vertices  $i$  do generateCollisionConstraints( $\mathbf{x}_i \rightarrow \mathbf{p}_i$ )
(9)  loop solverIterations times
(10)   projectConstraints( $C_1, \dots, C_{M+M_{coll}}, \mathbf{p}_1, \dots, \mathbf{p}_N$ )
(11) endloop
(12) forall vertices  $i$ 
(13)    $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i) / \Delta t$ 
(14)    $\mathbf{x}_i \leftarrow \mathbf{p}_i$ 
(15) endfor
(16) velocityUpdate( $\mathbf{v}_1, \dots, \mathbf{v}_N$ )
```

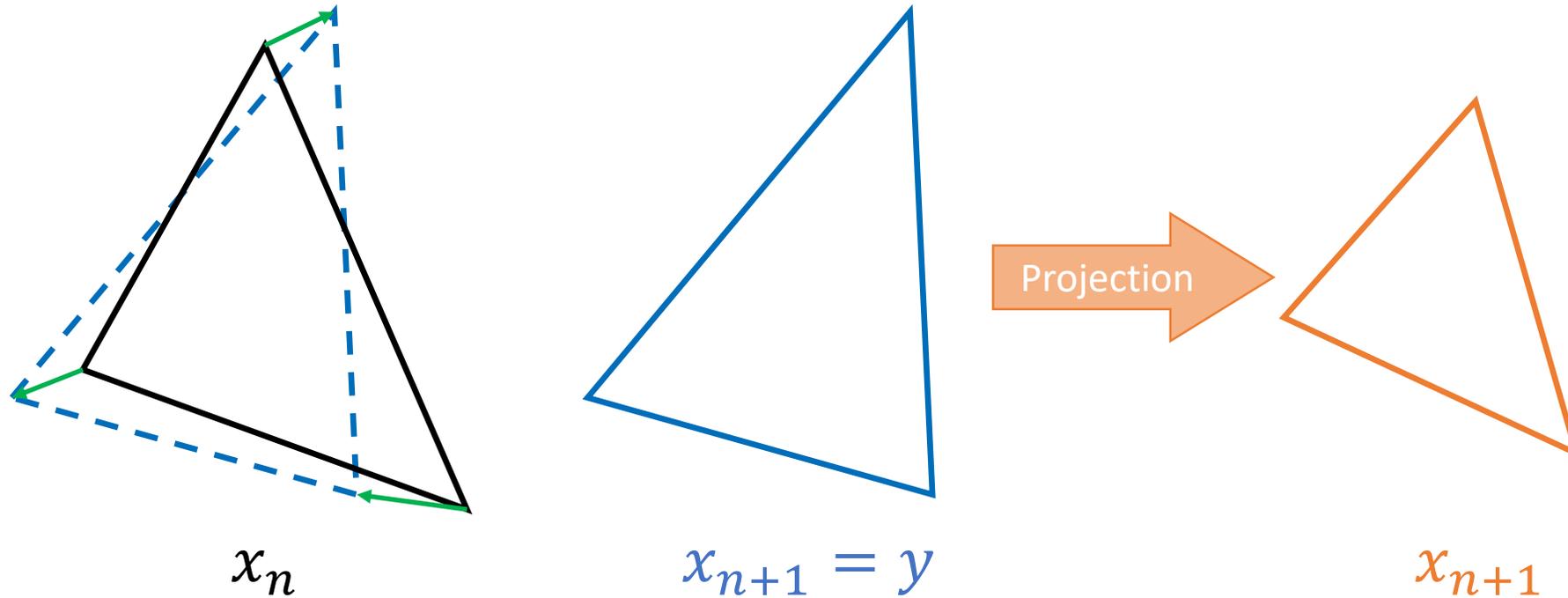
Init:

$$x_{n+1} = x_n + h v_n + h^2 M^{-1} f_{ext}$$

Project:

Move x_{n+1} to satisfy each constraint

PBD: Pipeline (Illustrated version)



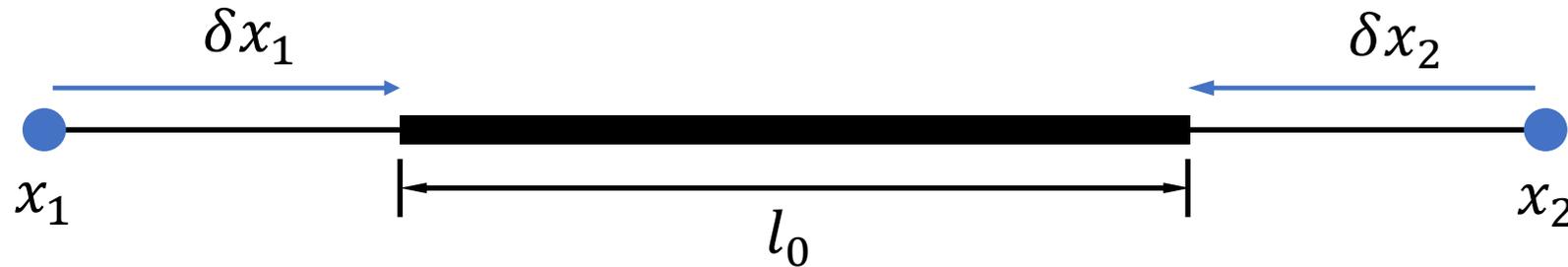
PBD: Projection

- Find a projection direction δx to:
 - Satisfy $c(x + \delta x) \approx c(x) + \nabla c(x)\delta x = 0$
 - Conserve linear momentum: $\sum m_i \delta x_i = 0$
 - Conserve angular momentum: $\sum m_i x_i \times \delta x_i = 0$

PBD: Projection (Cont'd)

- Construct δx_j for the j-th constraint as:
 - $\delta x_j = -M_j^{-1} \nabla c_j^T \delta \lambda_j$
- Compute the step size $\delta \lambda_j$ using $c_j(x) + \nabla c_j \delta x_j = 0$
 - $\delta \lambda_j = \frac{c_j}{\nabla c_j M_j^{-1} \nabla c_j^T}$

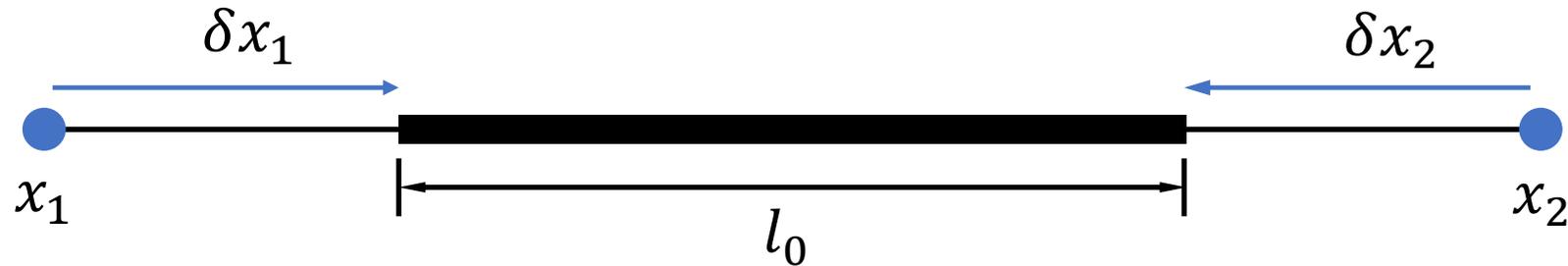
PBD Projection Example



$$c = \frac{\|x_1 - x_2\|}{l_0} - 1 \quad \nabla c = \left[\frac{1}{l_0} \frac{(x_1 - x_2)^T}{\|x_1 - x_2\|}, -\frac{1}{l_0} \frac{(x_1 - x_2)^T}{\|x_1 - x_2\|} \right] \quad \mathbf{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \otimes \mathbf{I}$$

$$\delta \lambda = \frac{c}{\nabla c \mathbf{M}^{-1} \nabla c} = \frac{l_0}{m_1^{-1} + m_2^{-1}} (\|x_1 - x_2\| - l_0)$$

PBD Projection Example

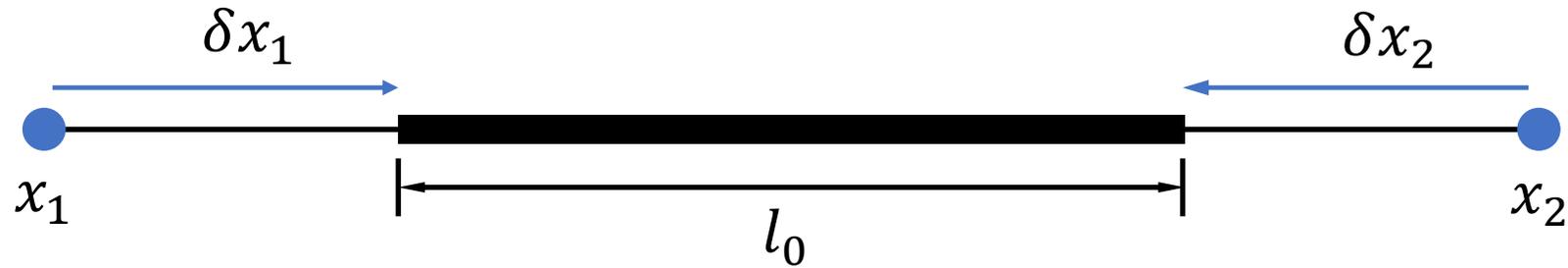


$$\delta x = -\mathbf{M}^{-1} \nabla c^T \delta \lambda$$

$$\delta x_1 = -\frac{l_0}{m_1^{-1} + m_2^{-1}} (\|x_1 - x_2\| - l_0) \frac{m_1^{-1}}{l_0} \frac{x_1 - x_2}{\|x_1 - x_2\|}$$

$$\delta x_2 = -\frac{l_0}{m_1^{-1} + m_2^{-1}} (\|x_1 - x_2\| - l_0) \frac{m_2^{-1}}{l_0} \left[-\frac{x_1 - x_2}{\|x_1 - x_2\|} \right]$$

PBD Projection Example



$$\delta x_1 = -\frac{m_1^{-1}}{m_1^{-1} + m_2^{-1}} (\|x_1 - x_2\| - l_0) \frac{x_1 - x_2}{\|x_1 - x_2\|}$$

$$\delta x_2 = +\frac{m_2^{-1}}{m_1^{-1} + m_2^{-1}} (\|x_1 - x_2\| - l_0) \frac{x_1 - x_2}{\|x_1 - x_2\|}$$

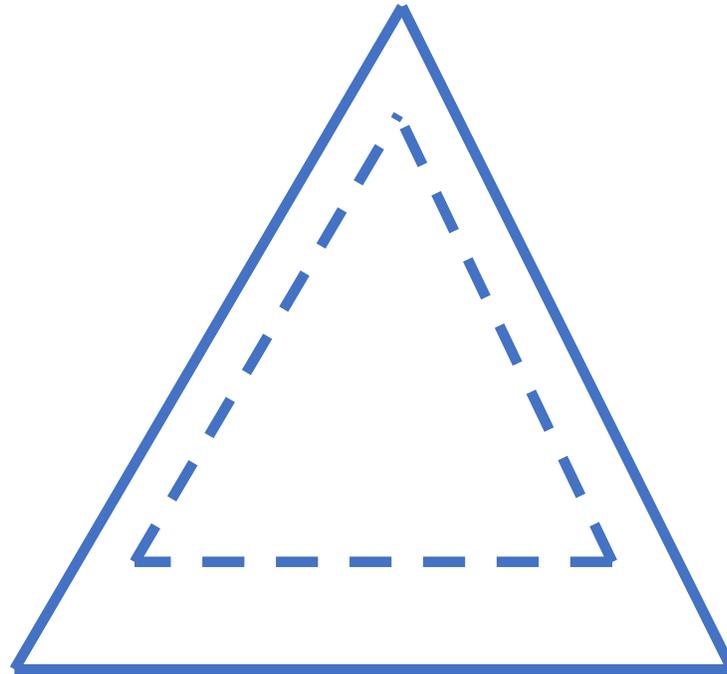
PBD: Pipeline

- (5) **forall** vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
- (6) dampVelocities($\mathbf{v}_1, \dots, \mathbf{v}_N$)
- (7) **forall** vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
- (8) **forall** vertices i **do** generateCollisionConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
- (9) **loop** solverIterations **times**
- (10) projectConstraints($C_1, \dots, C_{M+M_{coll}}, \mathbf{p}_1, \dots, \mathbf{p}_N$)
- (11) **endloop**
- (12) **forall** vertices i
- (13) $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i) / \Delta t$
- (14) $\mathbf{x}_i \leftarrow \mathbf{p}_i$
- (15) **endfor**
- (16) velocityUpdate($\mathbf{v}_1, \dots, \mathbf{v}_N$)

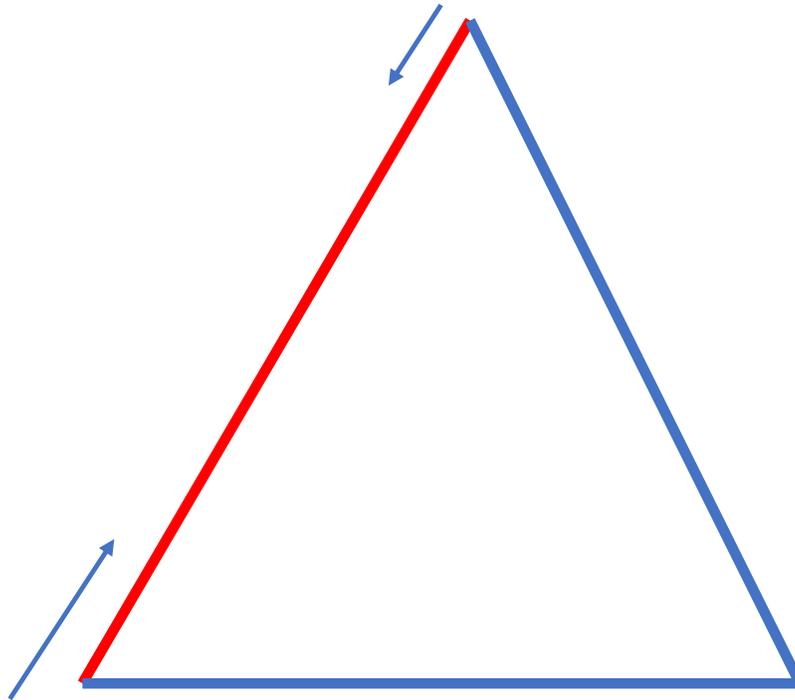
Project:

Move \mathbf{x}_{n+1} to satisfy each constraint

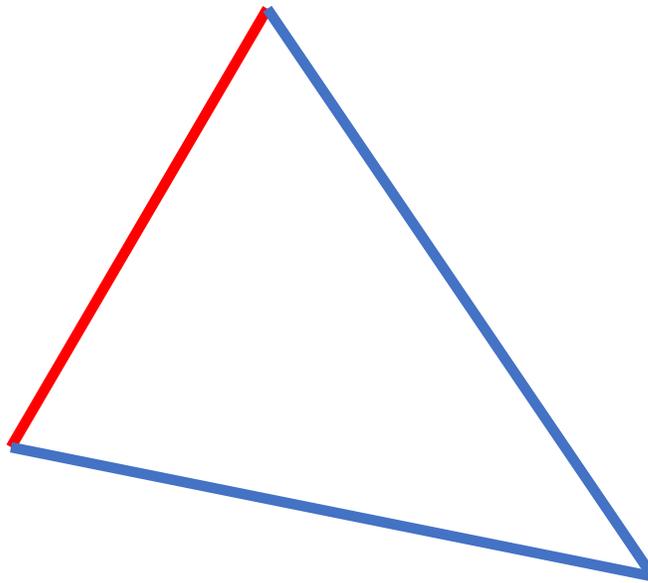
Iteration Strategy: Gauss-Seidel v.s. Jacobi



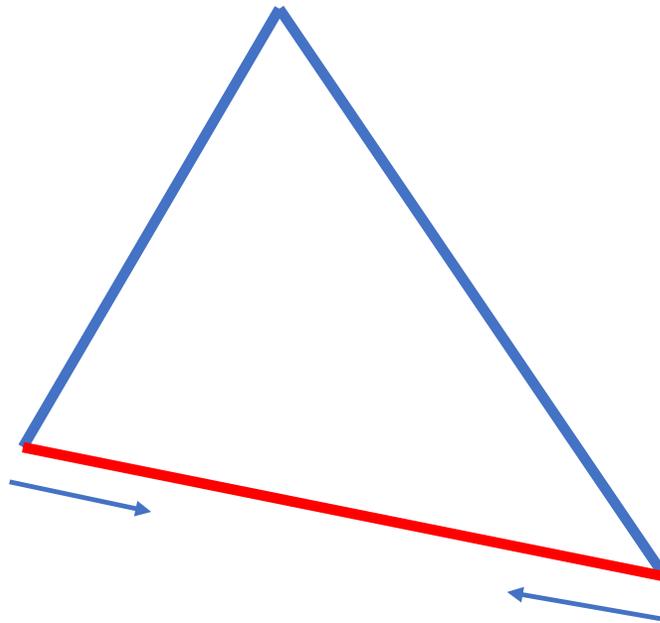
Gauss-Seidel



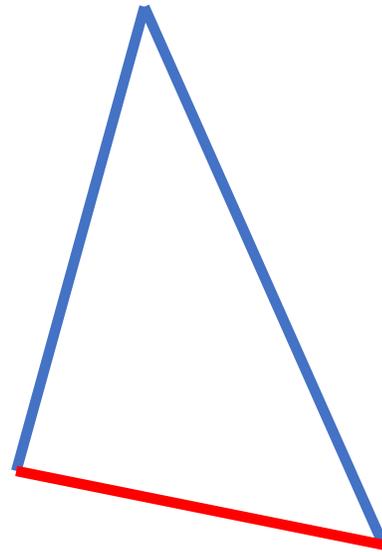
Gauss-Seidel



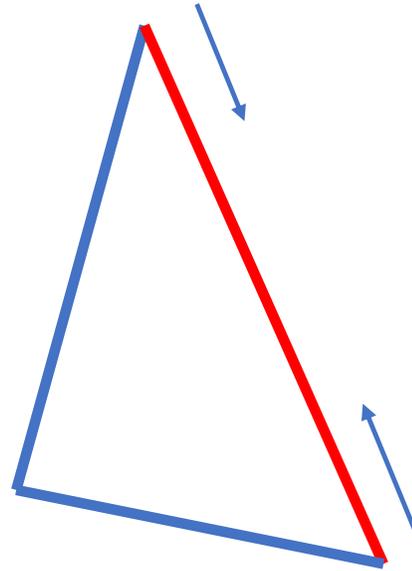
Gauss-Seidel



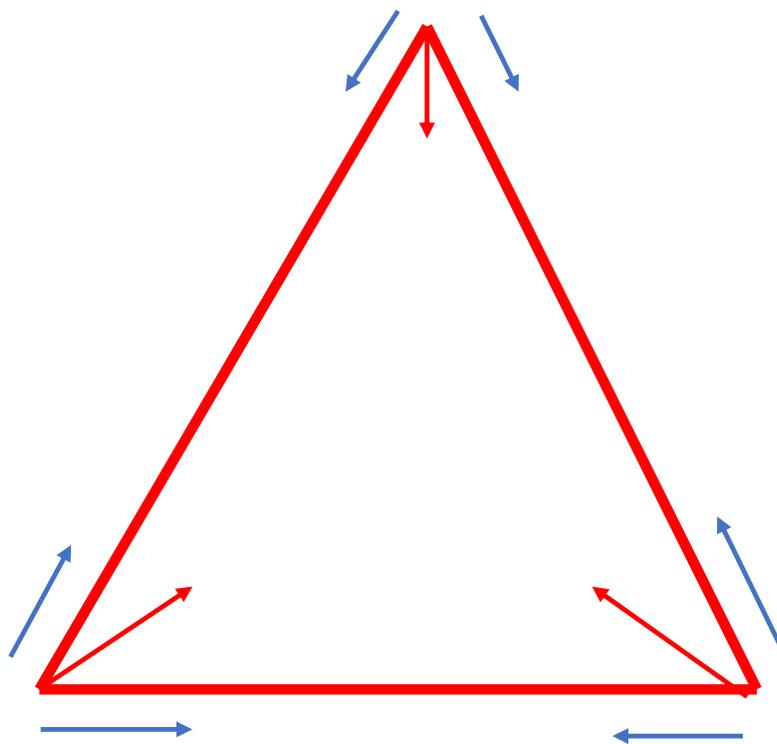
Gauss-Seidel



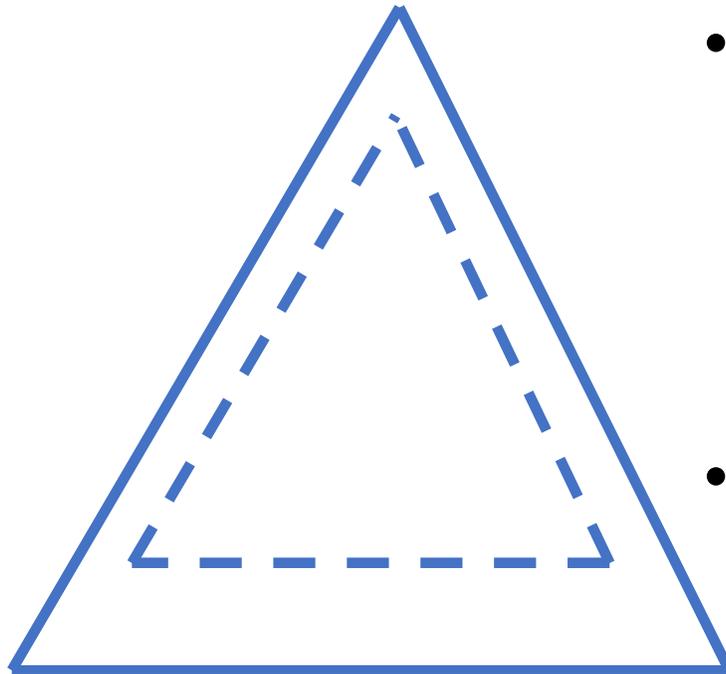
Gauss-Seidel



Jacobi



Gauss-Seidel v.s. Jacobi



- Gauss-Seidel

- + Converges faster
- Hard to parallelize
- May break the symmetry

<-- Colored GS [Fratarcangeli et al. 2016]

<-- Symmetric GS

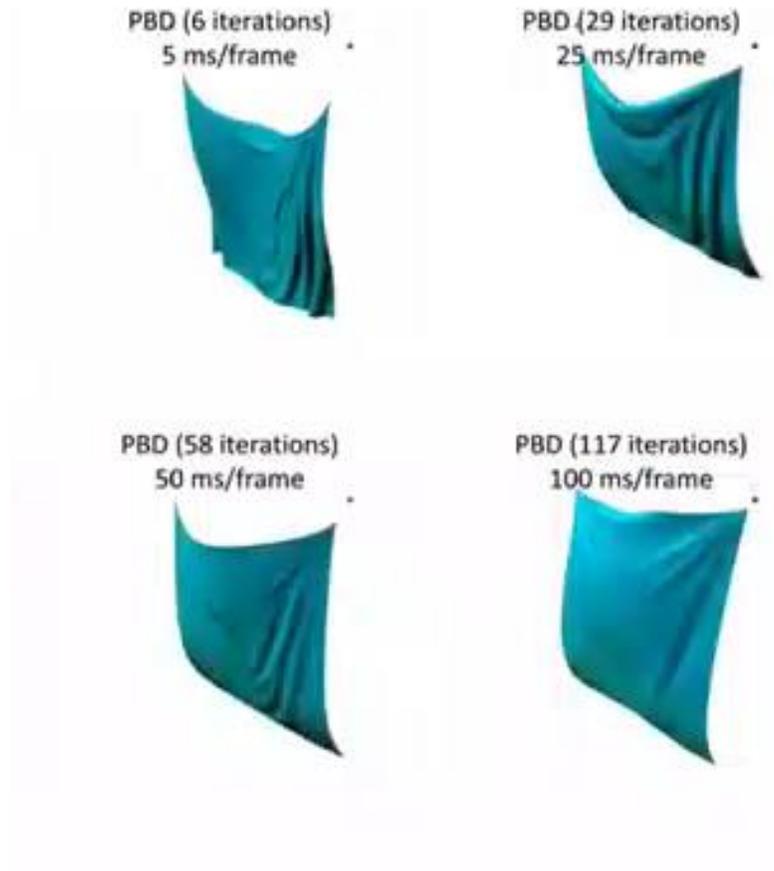
- Jacobi

- + Easy to parallelize
- Converges slower
- Less stable

<-- Chebyshev Acceleration [Wang 2015]

<-- Under Relaxation

Problems



Conclusion

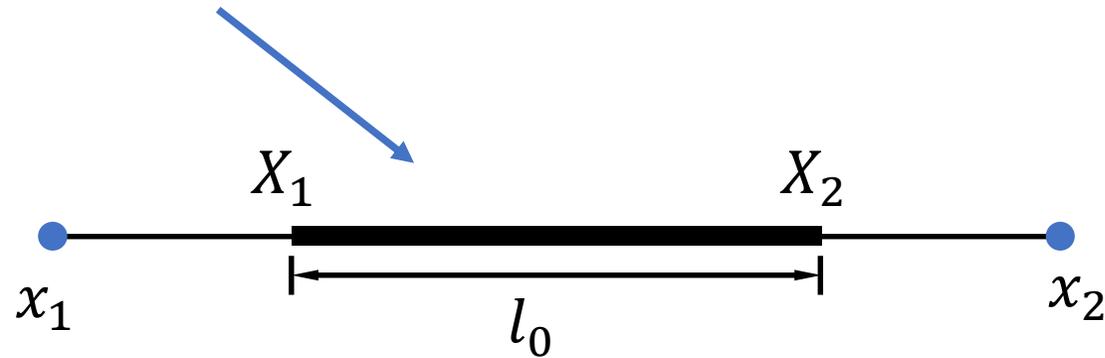
- Position Based Dynamics is:
 - Fast
 - Simple to implement
 - Stable (for most cases)
- It was also considered:
 - Non-physically-based (stiffness related to iteration count)
 - Hard to control

Variational Implicit Euler

- $x_{n+1} = \operatorname{argmin}_x \underbrace{\frac{1}{2h^2} \|x - y\|_M}_{\text{inertia}} + \underbrace{E(x)}_{\text{elasticity}}$
- What if $E(x)$ is (almost) infinitely stiff?

Constraint-based Variational Implicit Euler

- $\min_x \frac{1}{2h^2} \|x - y\|_M^2 \text{ s.t. } c(x) = 0$



$$c(x_1, x_2) = \frac{\|x_1 - x_2\|}{l_0} - 1 = 0$$

Constraint-based Variational Implicit Euler

- $\min_x \frac{1}{2h^2} \|x - y\|_M^2 \text{ s.t. } c(x) = 0$

- Optimality Condition:

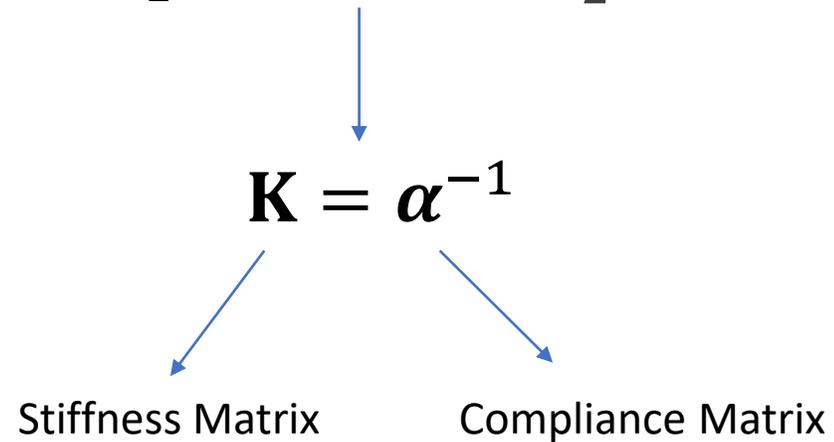
- $\frac{M}{h^2} (x - y) + \nabla_x c(x)^T \lambda = 0$

- $c(x) = 0$

Compliant Constraints

- Re-define energy using constraints:

- $E(x) = \sum_{e_i} \mu_i c_i(x)^2 = \frac{1}{2} c(x)^T \mathbf{K} c(x) = \frac{1}{2} c(x)^T \boldsymbol{\alpha}^{-1} c(x)$



Variational Implicit Euler with compliant constraints

- $\min_x \frac{1}{2h^2} \|x - y\|_M^2 + \frac{1}{2} c(x)^T \boldsymbol{\alpha}^{-1} c(x)$
- Optimality Condition
 - $\frac{M}{h^2} (x - y) + \nabla_x c(x)^T \boldsymbol{\alpha}^{-1} c(x) = 0$
- Optimality Condition with $\lambda = \boldsymbol{\alpha}^{-1} c(x)$
 - $\frac{M}{h^2} (x - y) + \nabla_x c(x)^T \lambda = 0$
 - $c(x) - \boldsymbol{\alpha} \lambda = 0$

Numerical Solutions

- To achieve the exact solution
 - Newton-Raphson
- To achieve an approximated solution
 - Step-and-Project [Goldenthal et al. 2007]
 - Semi-Implicit Euler [Tournier et al. 2015]
 - Position Based Dynamics [Müller et al. 2007, Macklin et al. 2016]

$$\begin{aligned}\frac{M}{h^2}(x - y) + \nabla_x c(x)^T \lambda &= 0 \\ c(x) - \alpha \lambda &= 0\end{aligned}$$

Newton-Raphson method

$$\begin{aligned}\frac{M}{h^2}(x - y) + \nabla_x c(x)^T \lambda &= 0 \\ c(x) - \alpha \lambda &= 0\end{aligned}$$

- For a given state $x^{(k)}$ and $\lambda^{(k)}$
- Compute Newton-Raphson direction δx and $\delta \lambda$ using:

$$\begin{bmatrix} \frac{M}{h^2} + \sum_{j=1}^m \lambda_j^{(k)} \nabla^2 c_j & \nabla c^T \\ \nabla c & -\alpha \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \frac{M}{h^2}(x^{(k)} - y) + \nabla_x c^T \lambda^{(k)} \\ c(x^{(k)}) - \alpha \lambda^{(k)} \end{bmatrix}$$

Hard Constraints

- $\alpha = 0$

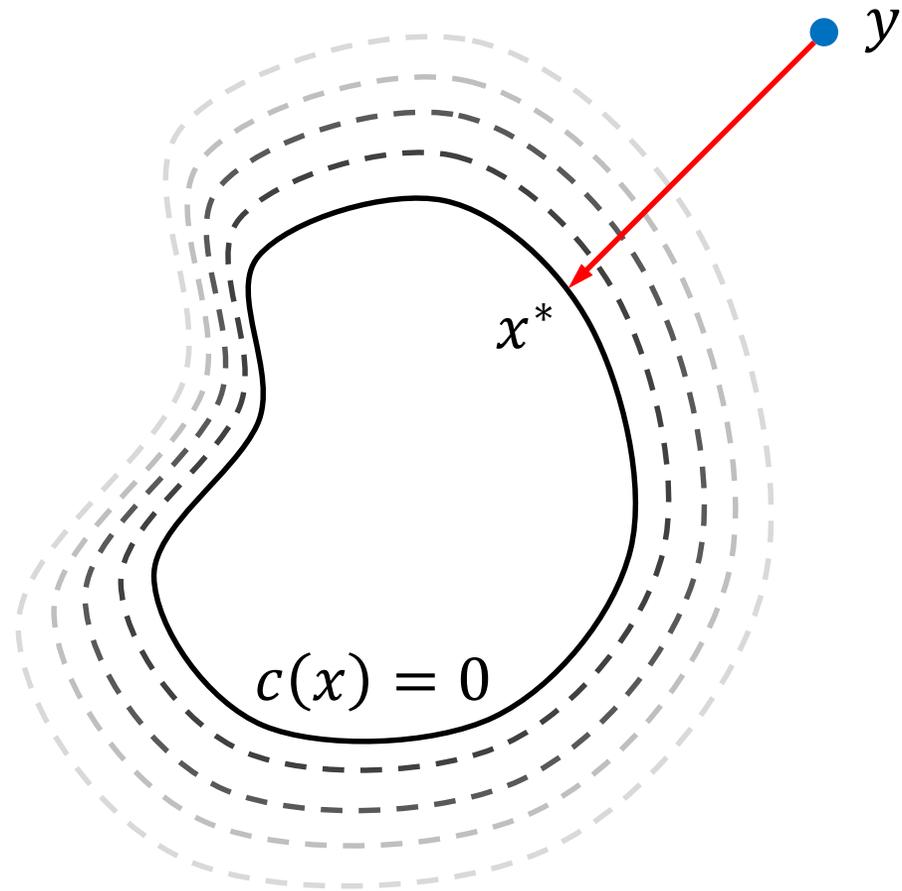
$$\begin{aligned}\frac{M}{h^2}(x - y) + \nabla_x c(x)^T \lambda &= 0 \\ c(x) - \alpha \lambda &= 0\end{aligned}$$

$$\begin{aligned}\frac{M}{h^2}(x - y) + \nabla_x c(x)^T \lambda &= 0 \\ c(x) &= 0\end{aligned}$$



$$\begin{aligned}\min_x \frac{1}{2h^2} \|x - y\|_M^2 \\ \text{s.t. } c(x) = 0\end{aligned}$$

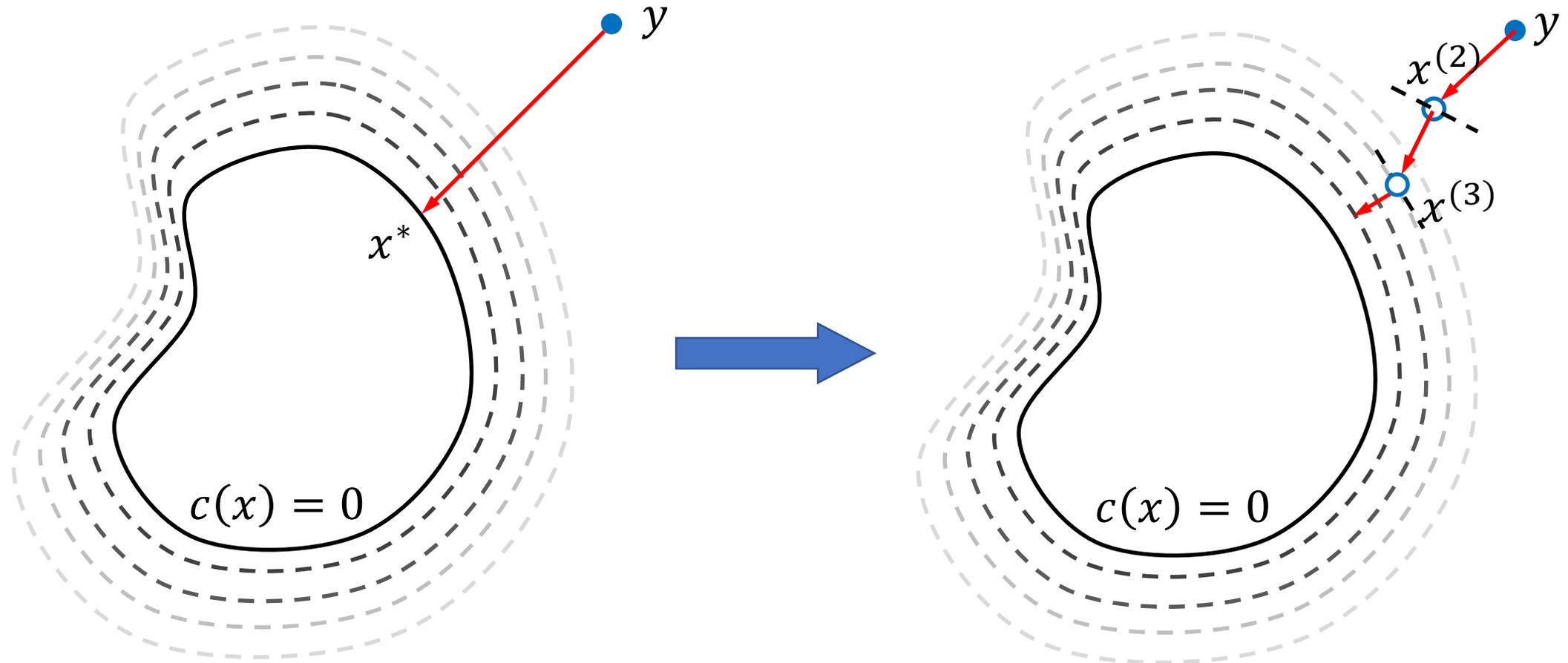
Hard Constraints: Geometric Interpretation



$$\begin{aligned} \min_x & \frac{1}{2h^2} \|x - y\|_M^2 \\ \text{s.t. } & c(x) = 0 \end{aligned}$$

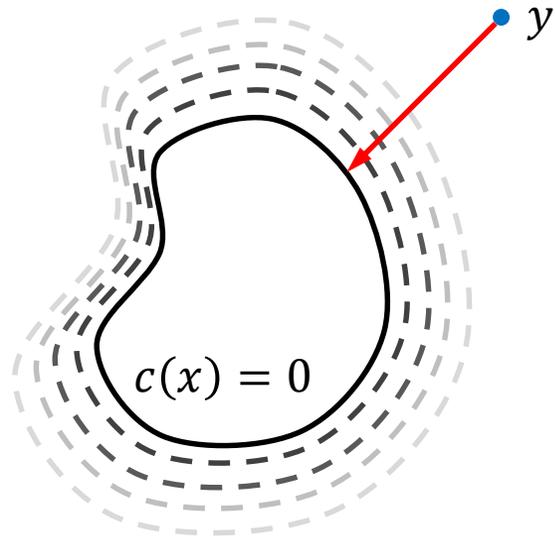
Step and Project (SAP)

[Goldenthal et al. 2007]

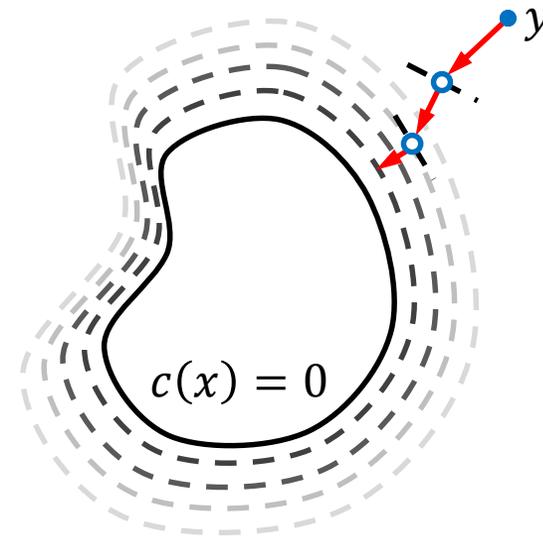


Step and Project (SAP)

[Goldenthal et al. 2007]



$$\min_x \frac{1}{2h^2} \|x - y\|_M^2$$
$$s.t. c(x) = 0$$



$$x_{k+1} = \min_x \frac{1}{2h^2} \|x - x^{(k)}\|_M^2$$
$$s.t. c(x^{(k)}) + \nabla_x c(x^{(k)})(x - x^{(k)}) = 0$$

SAP: Update

$$\begin{aligned}\frac{M}{h^2} (x - x^{(k)}) + \nabla_x c(x^{(k)})^T \lambda &= 0 \\ c(x^{(k)}) + \nabla_x c(x^{(k)}) (x - x^{(k)}) &= 0\end{aligned}$$

Initialize $x^{(k+1)}$ and $\lambda^{(k+1)}$
with $x^{(k)}$ and 0

$$\begin{bmatrix} M/h^2 & \nabla c^T \\ \nabla c & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} 0 \\ c(x^{(k)}) \end{bmatrix}$$

Schur Complement of 0: $-\nabla c (M/h^2)^{-1} \nabla c^T$

$$[-\nabla c (M/h^2)^{-1} \nabla c^T] \delta \lambda = -c(x^{(k)})$$

SAP: Update

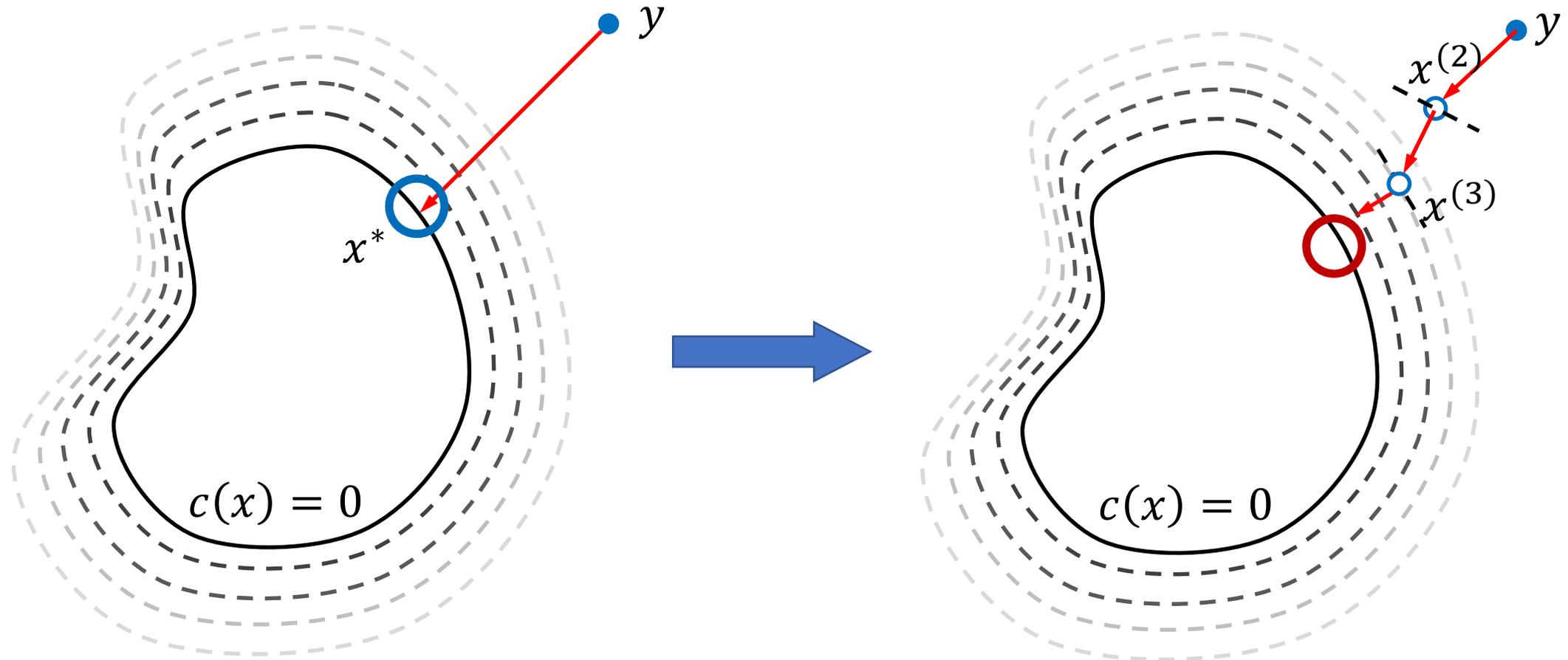
$$\begin{aligned}\frac{M}{h^2} (x - x^{(k)}) + \nabla_x c(x^{(k)})^T \lambda &= 0 \\ c(x^{(k)}) + \nabla_x c(x^{(k)})(x - x^{(k)}) &= 0\end{aligned}$$

Initialize $x^{(k+1)}$ and $\lambda^{(k+1)}$
with $x^{(k)}$ and 0

$$\begin{bmatrix} M/h^2 & \nabla c^T \\ \nabla c & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} 0 \\ c(x^{(k)}) \end{bmatrix}$$

$$\begin{cases} \delta \lambda = \frac{1}{h^2} (\nabla c M^{-1} \nabla c^T)^{-1} c(x^{(k)}) \\ \delta x = -h^2 M^{-1} \nabla c^T \delta \lambda \end{cases}$$

SAP: Solution Drifting



SAP vs. the Exact Solution



SAP



Exact
Solution

PBD vs. SAP

- For each constraint j :

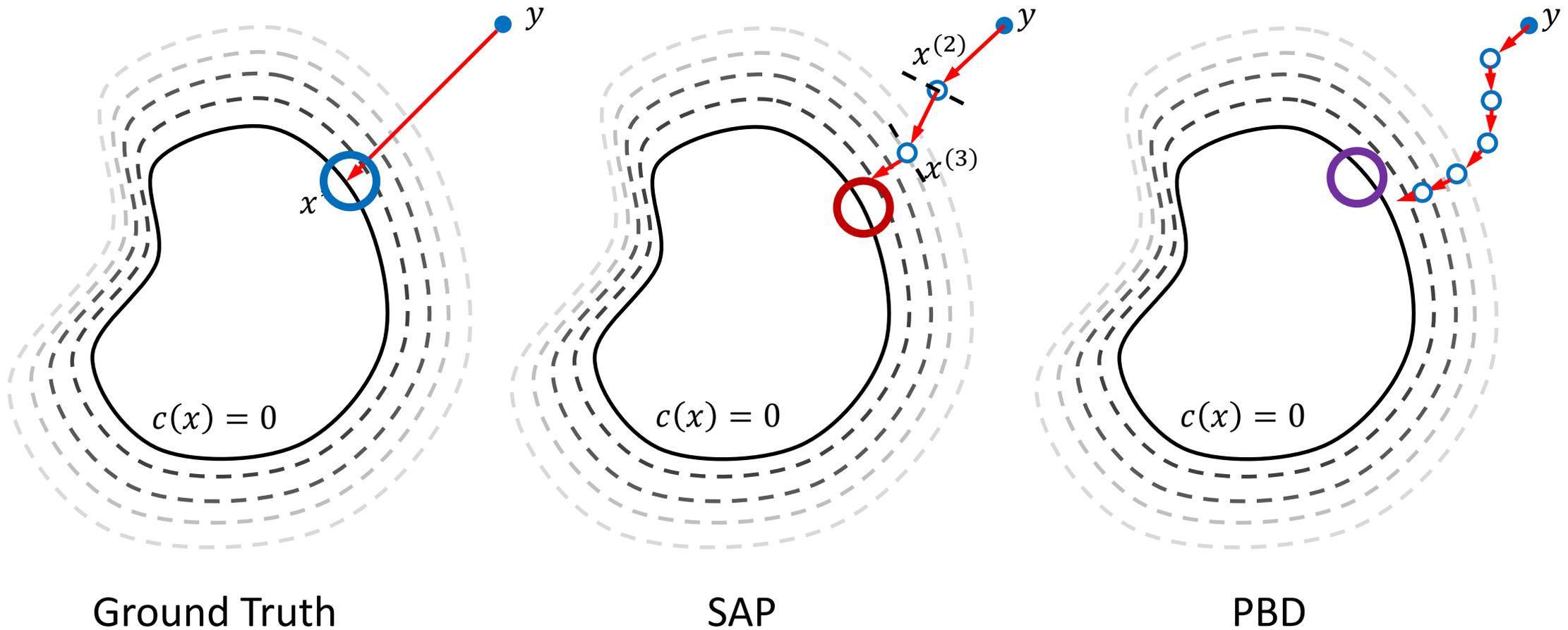
- Compute $\delta\lambda_j$: $\delta\lambda_j = \frac{c_j}{\nabla c_j M_j^{-1} \nabla c_j^T}$
- Compute δx_j : $\delta x_j = -M_j^{-1} \nabla c_j^T \delta\lambda_j$
- Commit: $x = x + \delta x_j$

PBD (G-S)

- $\delta\lambda = \frac{1}{h^2} (\nabla c M^{-1} \nabla c^T)^{-1} c$
- $\delta x = -h^2 M^{-1} \nabla c^T \delta\lambda$
- $x = x + \delta x$

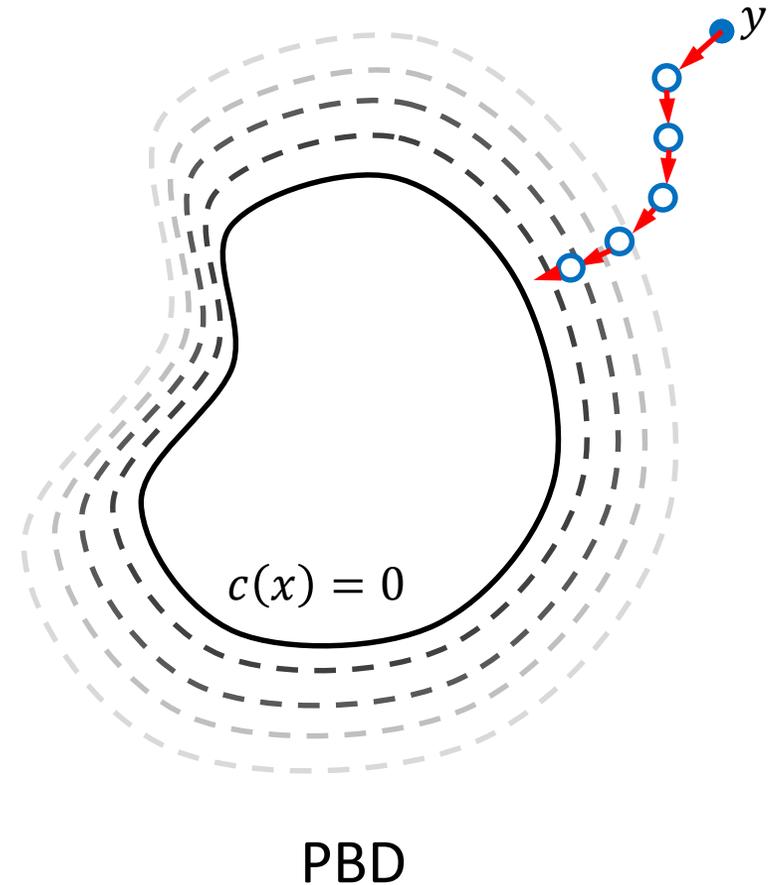
SAP

PBD v.s. SAP: Geometric Interpretation



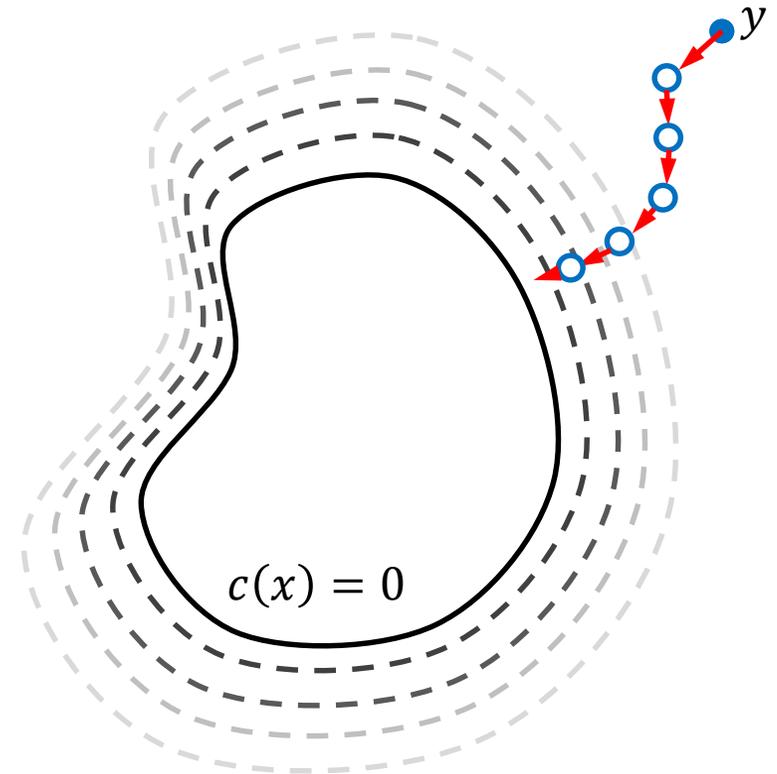
PBD Problems Visualized

- Solution Drifts
- Non-physically-based
- Hard to Control



PBD Problems Visualized

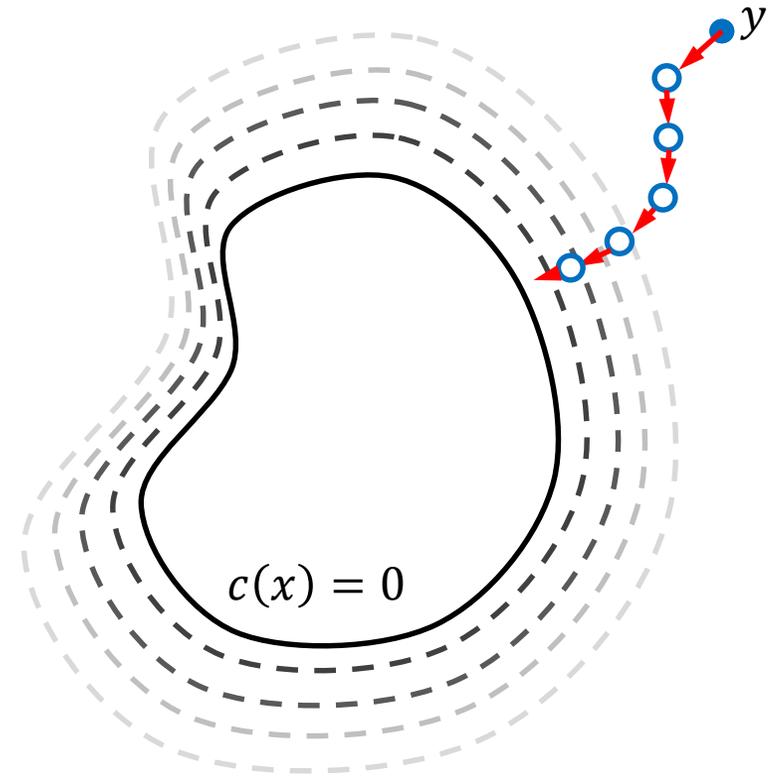
- Solution Drifts
- Non-physically-based
 - Stiffness depends on iteration count
 - $y = x_n + hv_n + h^2 M^{-1} f_{ext}$
- Hard to Control



PBD

PBD Problems Visualized

- Solution Drifts
- Non-physically-based
- Hard to Control
 - $\min_x \frac{1}{2h^2} \|x - y\|_M^2$
 - *s. t.* $c(x) = 0$



PBD

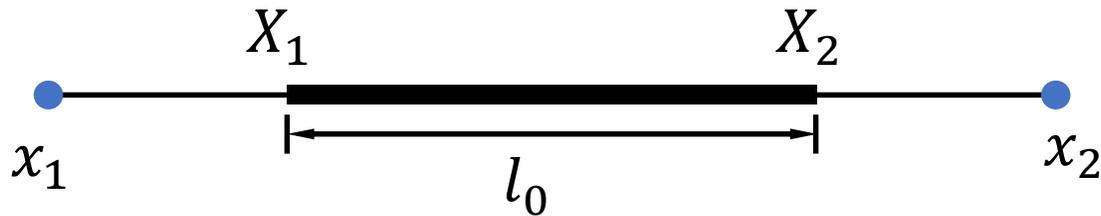
Compliant Constraints

- $\alpha \neq 0, E = \frac{1}{2} \mathbf{c}^T \alpha^{-1} \mathbf{c}$

$$\begin{aligned} \frac{M}{h^2} (x - y) + \nabla_x c(x)^T \lambda &= 0 \\ c(x) - \alpha \lambda &= 0 \end{aligned}$$

Compliant Constraints

- $\alpha \neq \mathbf{0}, E = \frac{1}{2} \mathbf{c}^T \alpha^{-1} \mathbf{c}$
- Mass Spring System



$$E(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{2} k (\|\mathbf{x}_1 - \mathbf{x}_2\| - l_0)^2$$

$$\begin{aligned} \frac{M}{h^2} (\mathbf{x} - \mathbf{y}) + \nabla_{\mathbf{x}} c(\mathbf{x})^T \lambda &= 0 \\ c(\mathbf{x}) - \alpha \lambda &= 0 \end{aligned}$$

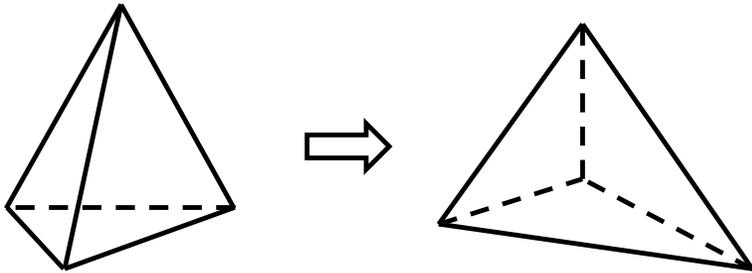
$$c(\mathbf{x}_1, \mathbf{x}_2) = \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|}{l_0} - 1$$

$$k = \frac{2\mu A}{l_0}$$

$$\alpha = [2\mu A l_0]^{-1} = [k l_0^2]^{-1}$$

Compliant Constraints

- $\alpha \neq \mathbf{0}$, $E = \frac{1}{2} \mathbf{c}^T \alpha^{-1} \mathbf{c}$
- FEM (3D case)



$$E(\mathbf{F}(\mathbf{x})) = V(\mu \|\boldsymbol{\epsilon}\|_F^2 + \frac{1}{2} \lambda \text{tr}^2(\boldsymbol{\epsilon}))$$

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_x & 0.5\epsilon_{xy} & 0.5\epsilon_{xz} \\ 0.5\epsilon_{xy} & \epsilon_y & 0.5\epsilon_{yz} \\ 0.5\epsilon_{xz} & 0.5\epsilon_{yz} & \epsilon_z \end{bmatrix}$$

$$\begin{aligned} \frac{M}{h^2} (\mathbf{x} - \mathbf{y}) + \nabla_{\mathbf{x}} c(\mathbf{x})^T \lambda &= 0 \\ c(\mathbf{x}) - \alpha \lambda &= 0 \end{aligned}$$

$$E = \frac{1}{2} \mathbf{c}^T \mathbf{K} \mathbf{c}$$

$$\mathbf{c} = [\epsilon_x \quad \epsilon_y \quad \epsilon_z \quad \epsilon_{xy} \quad \epsilon_{xz} \quad \epsilon_{yz}]^T$$

$$\mathbf{K} = V \begin{bmatrix} \lambda + 2\mu & \lambda & \lambda & & & \\ \lambda & \lambda + 2\mu & \lambda & & & \\ \lambda & \lambda & \lambda + 2\mu & & & \\ & & & \mu & & \\ & & & & \mu & \\ & & & & & \mu \end{bmatrix}$$

$$\boldsymbol{\alpha} = \mathbf{K}^{-1}$$

Solve Compliant Constraints with Newton

$$\begin{bmatrix} \frac{M}{h^2} + \sum_{j=1}^m \lambda_j^{(k)} \nabla^2 c_j & \nabla c^T \\ \nabla c & -\alpha \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \frac{M}{h^2} (x^{(k)} - y) + \nabla c^T \lambda^{(k)} \\ c(x^{(k)}) - \alpha \lambda^{(k)} \end{bmatrix}$$

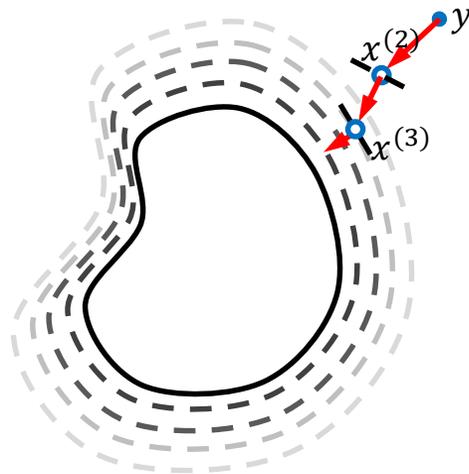
SAP Assumption:

$$y \rightarrow x^{(k)}$$

$$\nabla^2 c = 0$$

SAP Initialization:

$$x \leftarrow x^{(k)}$$



Solve Compliant Constraints with SAP

$$\begin{bmatrix} M & \\ \frac{1}{h^2} & \nabla c^T \\ \nabla c & -\alpha \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla c^T \lambda^{(k)} \\ c(x^{(k)}) - \alpha \lambda^{(k)} \end{bmatrix}$$
$$\left\{ \begin{array}{l} \delta \lambda = \frac{1}{h^2} (\nabla c M^{-1} \nabla c^T + h^2 \alpha)^{-1} (c(x^{(k)}) - \alpha \lambda^{(k)}) \\ \lambda^{(k+1)} = \lambda^{(k)} + \delta \lambda \\ \delta x = -h^2 M^{-1} \nabla c^T \lambda^{(k+1)} \\ x^{(k+1)} = x^{(k)} + \delta x \end{array} \right.$$

Solve Compliant Constraints with XPBD

[Macklin et al. 2016]

$$\begin{bmatrix} M & \\ \frac{1}{h^2} & \nabla c^T \\ \nabla c & -\alpha \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla c^T \lambda^{(k)} \\ c(x^{(k)}) - \alpha \lambda^{(k)} \end{bmatrix}$$

For each constraint: j

Compute: $\delta \lambda_j = \frac{1}{h^2} (\nabla c_j M_j^{-1} \nabla c_j^T + h^2 \alpha_j)^{-1} (c_j(x^{(k)}) - \alpha_j \lambda_j^{(k)})$

Commit: $\lambda_j^{(k+1)} = \lambda_j^{(k)} + \delta \lambda_j$

Compute: $\delta x_j = -h^2 M_j^{-1} \nabla c_j^T \lambda_j^{(k+1)}$

Commit: $x_j^{(k+1)} = x_j^{(k)} + \delta x_j$

PBD v.s. XPBD

- For each constraint j :

- Compute $\delta\lambda_j$: $\delta\lambda_j = \frac{1}{h^2} \frac{c_j}{\nabla c_j M_j^{-1} \nabla c_j^T}$
- Commit: $\lambda_j = \delta\lambda_j$
- Compute δx_j : $\delta x_j = -h^2 M_j^{-1} \nabla c_j^T \lambda_j$
- Commit: $x = x + \delta x_j$

PBD (G-S)

- For each constraint j :

- Compute $\delta\lambda_j$: $\delta\lambda_j = \frac{1}{h^2} \frac{c_j - \alpha_j \lambda_j}{\nabla c_j M_j^{-1} \nabla c_j^T + h^2 \alpha_j}$
- Commit: $\lambda_j = \lambda_j + \delta\lambda_j$
- Compute δx_j : $\delta x_j = -h^2 M_j^{-1} \nabla c_j^T \lambda_j$
- Commit: $x = x + \delta x_j$

XPBD (G-S)

XPBD Result

Inflatable Balloon

Volume, stretch, shear, bending constraints of varying stiffness.

2.5k particles
15k constraints

What is left out?

$$\begin{aligned} \frac{M}{h^2} (x - y) + \nabla_x c(x)^T \lambda &= 0 \\ c(x) - \alpha \lambda &= 0 \end{aligned}$$

$$\begin{bmatrix} \frac{M}{h^2} + \sum_{j=1}^m \lambda_j^{(k)} \nabla^2 c_j & \nabla c^T \\ \nabla c & -\alpha \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \frac{M}{h^2} (x^{(k)} - y) + \nabla c^T \lambda^{(k)} \\ c(x^{(k)}) - \alpha \lambda^{(k)} \end{bmatrix}$$

Schur Complement of the Upper-left Block

$$\begin{bmatrix} \frac{M}{h^2} + \sum_{j=1}^m \lambda_j^{(k)} \nabla^2 c_j & \nabla c^T \\ \nabla c & -\alpha \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \frac{M}{h^2} (x^{(k)} - y) + \nabla c^T \lambda^{(k)} \\ c(x^{(k)}) - \alpha \lambda^{(k)} \end{bmatrix}$$

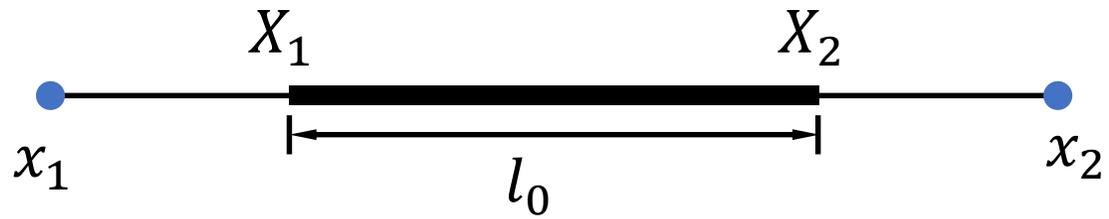
$$\frac{M}{h^2} + \nabla c^T \alpha^{-1} \nabla c + \sum_{j=1}^m \lambda_j^{(k)} \nabla^2 c_j$$

Geometric Stiffness

$$\begin{bmatrix} \frac{M}{h^2} + \sum_{j=1}^m \lambda_j^{(k)} \nabla^2 c_j & \nabla c^T \\ \nabla c & -\alpha \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \frac{M}{h^2} (x^{(k)} - y) + \nabla c^T \lambda^{(k)} \\ c(x^{(k)}) - \alpha \lambda^{(k)} \end{bmatrix}$$

$$\frac{M}{h^2} + \nabla c^T \alpha^{-1} \nabla c + \sum_{j=1}^m \lambda_j^{(k)} \nabla^2 c_j$$

Geometric Stiffness: Example



$$c = \frac{\|x_1 - x_2\|}{l_0} - 1$$

$$E = \frac{1}{2} \alpha^{-1} c^2$$

$$\nabla_{x_1, x_1}^2 E = \nabla c^T \alpha^{-1} \nabla c + (\alpha^{-1} c) \nabla_{x_1, x_1}^2 c$$

Geometric
Stiffness

Stable Constrained Dynamics

[Tournier et al. 2015]

- When to use Geometric Stiffness?
 1. Material highly nonlinear
 2. Strong stiffness

$$\begin{bmatrix} \frac{M}{h^2} + \sum_{j=1}^m \lambda_j^{(k)} \nabla^2 c_j & \nabla c^T \\ \nabla c & -\alpha \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \frac{M}{h^2} (x^{(k)} - y) + \nabla c^T \lambda^{(k)} \\ c(x^{(k)}) - \alpha \lambda^{(k)} \end{bmatrix}$$

Conclusion

$$\begin{aligned}\frac{M}{h^2}(x - y) + \nabla_x c(x)^T \lambda &= 0 \\ c(x) - \alpha \lambda &= 0\end{aligned}$$

	constraint	#iterations/frame	per-iteration cost
[Goldenthal et al. 2007]	hard	medium	medium
[Müller et al. 2007]	hard	high	low
[Tournier et al. 2015]	compliant	one	high
[Macklin et al. 2016]	compliant	high	low

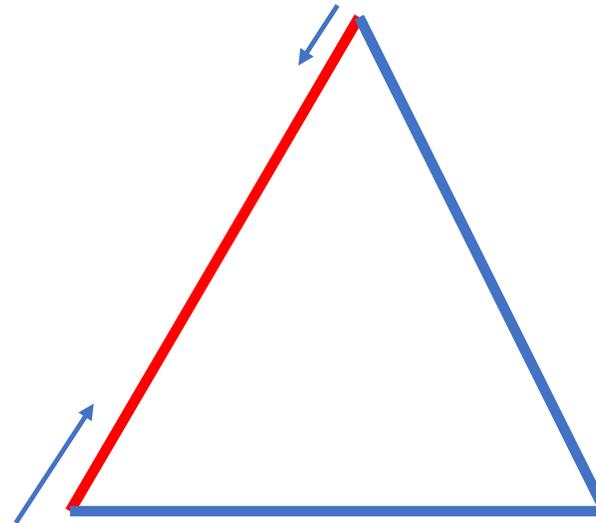
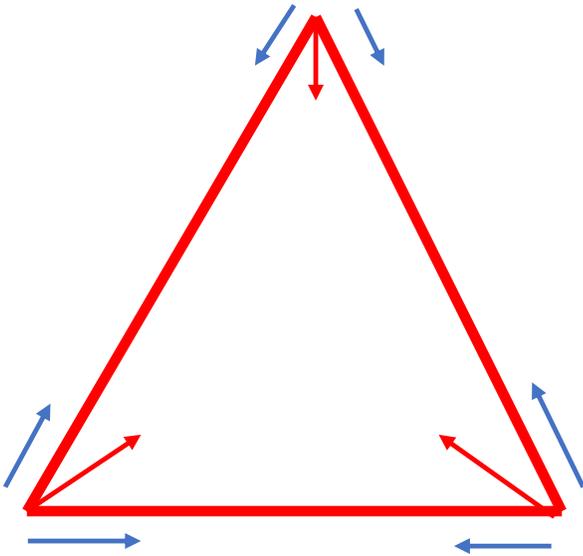
Conclusion: PBD

$$\frac{M}{h^2} (x - y) + \nabla_x c(x)^T \lambda = 0$$
$$c(x) - \alpha \lambda = 0$$

	constraint	#iterations/frame	per-iteration cost
[Goldenthal et al. 2007]	hard	medium	medium
[Müller et al. 2007]	hard	high	low
[Tournier et al. 2015]	compliant	one	high
[Macklin et al. 2016]	compliant	high	low

Conclusion: PBD

- The Performance of (X)PBD is...
 - Similar to one iteration of Jacobi/Gauss-Seidel of SAP



Conclusion: SAP

$$\frac{M}{h^2} (x - y) + \nabla_x c(x)^T \lambda = 0$$
$$c(x) - \alpha \lambda = 0$$

	constraint	#iterations/frame	per-iteration cost
[Goldenthal et al. 2007]	hard	medium	medium
[Müller et al. 2007]	hard	high	low
[Tournier et al. 2015]	compliant	one	high
[Macklin et al. 2016]	compliant	high	low

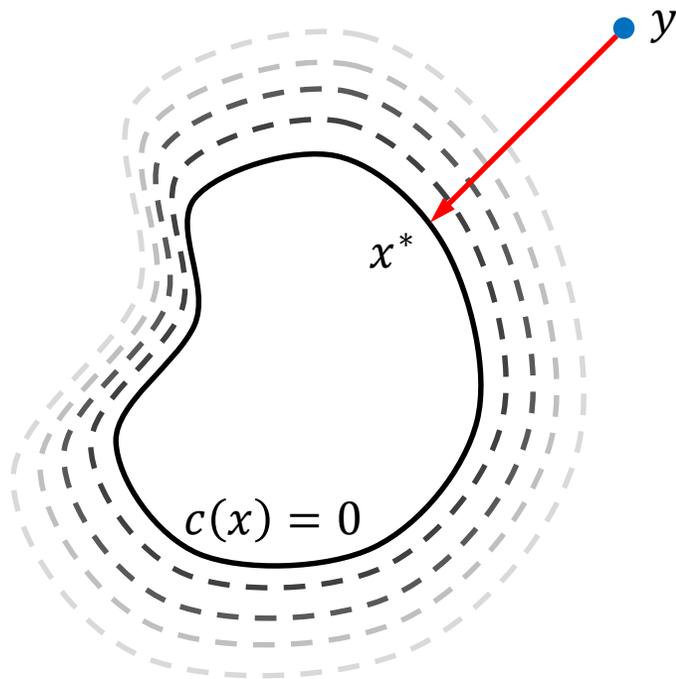
Conclusion: Geometric Stiffness

$$\frac{M}{h^2} (x - y) + \nabla_x c(x)^T \lambda = 0$$
$$c(x) - \alpha \lambda = 0$$

	constraint	#iterations/frame	per-iteration cost
[Goldenthal et al. 2007]	hard	medium	medium
[Müller et al. 2007]	hard	high	low
[Tournier et al. 2015]	compliant	one	high
[Macklin et al. 2016]	compliant	high	low

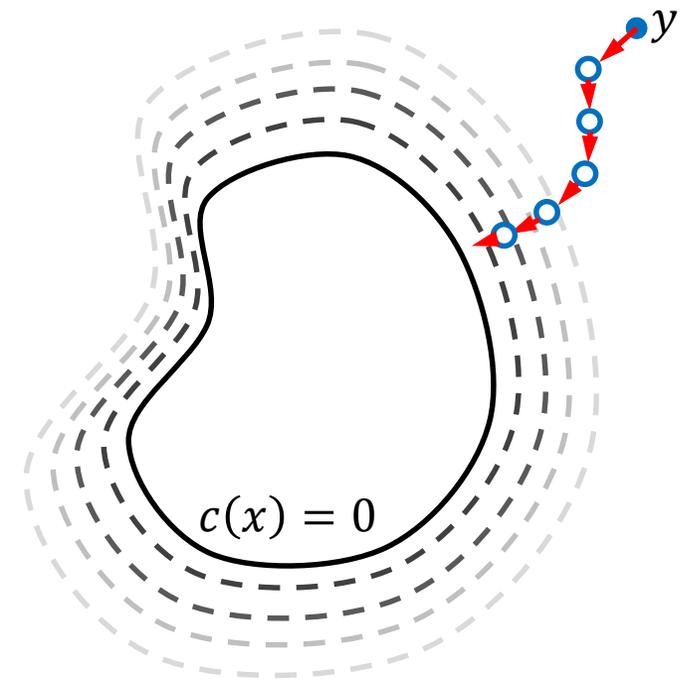
Key Takeaway from this Talk

- (X)PBD is approximately converging to...



Ground Truth

$$\begin{aligned} \frac{M}{h^2} (x - y) + \nabla_x c(x)^T \lambda &= 0 \\ c(x) - \alpha \lambda &= 0 \end{aligned}$$



PBD

Position Based Dynamics

A fast yet physically plausible method for deformable body simulation

Tiantian Liu

GAMES Webinar

03/28/2019

Microsoft
Research
微软亚洲研究院