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

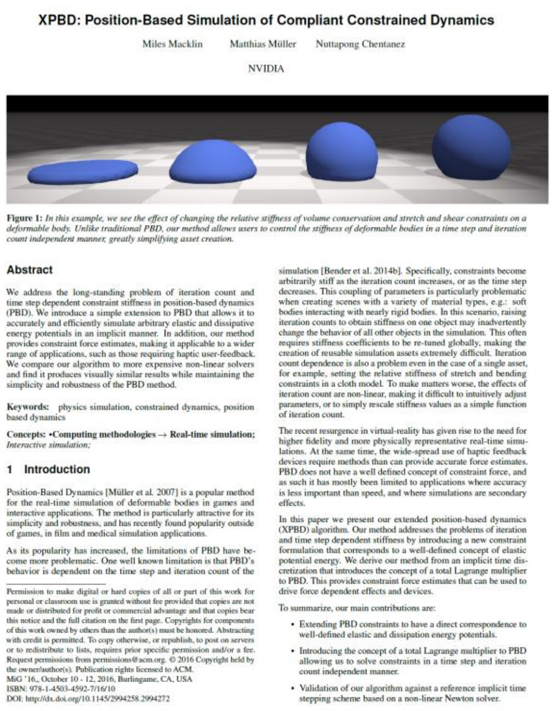
03/28/2019

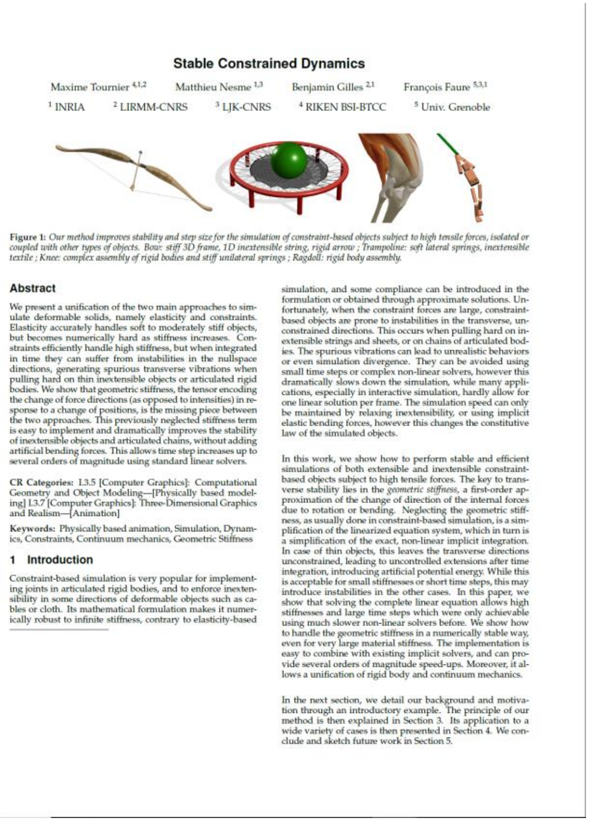
Microsoft Research
微软亚洲研究院

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

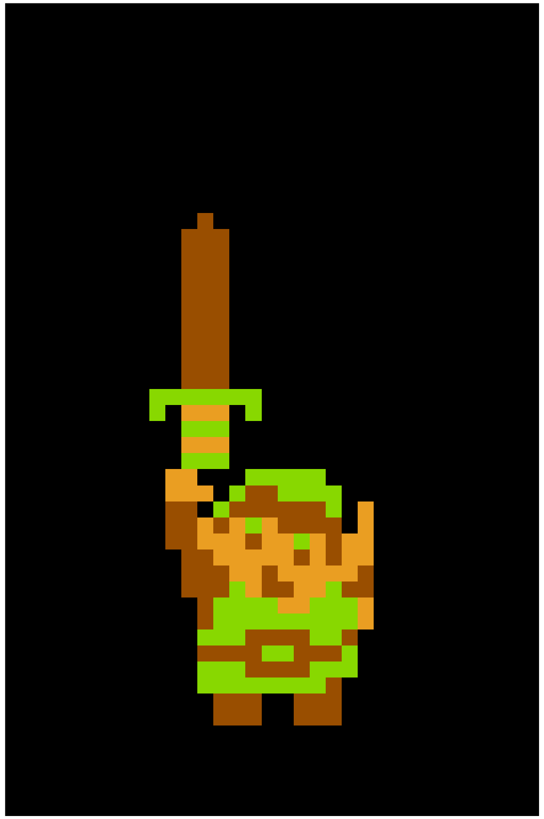
Major References of this Talk





[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

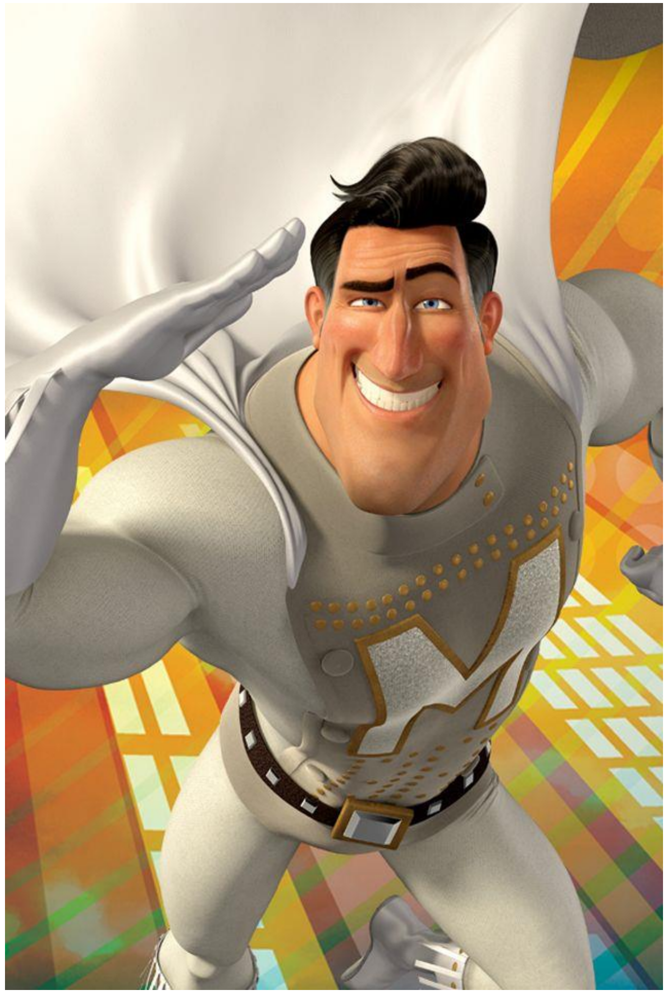
Game

Movie/Animation

AR/VR

Deformable Body Simulation

Game



Movie/Animation

AR/VR

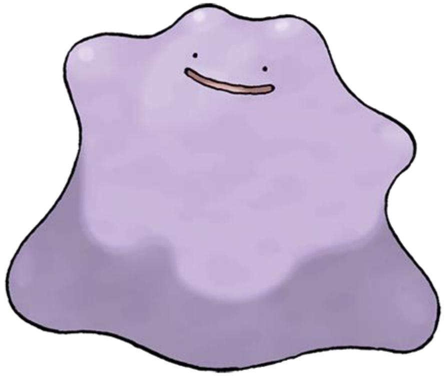
Deformable Body Simulation

Game

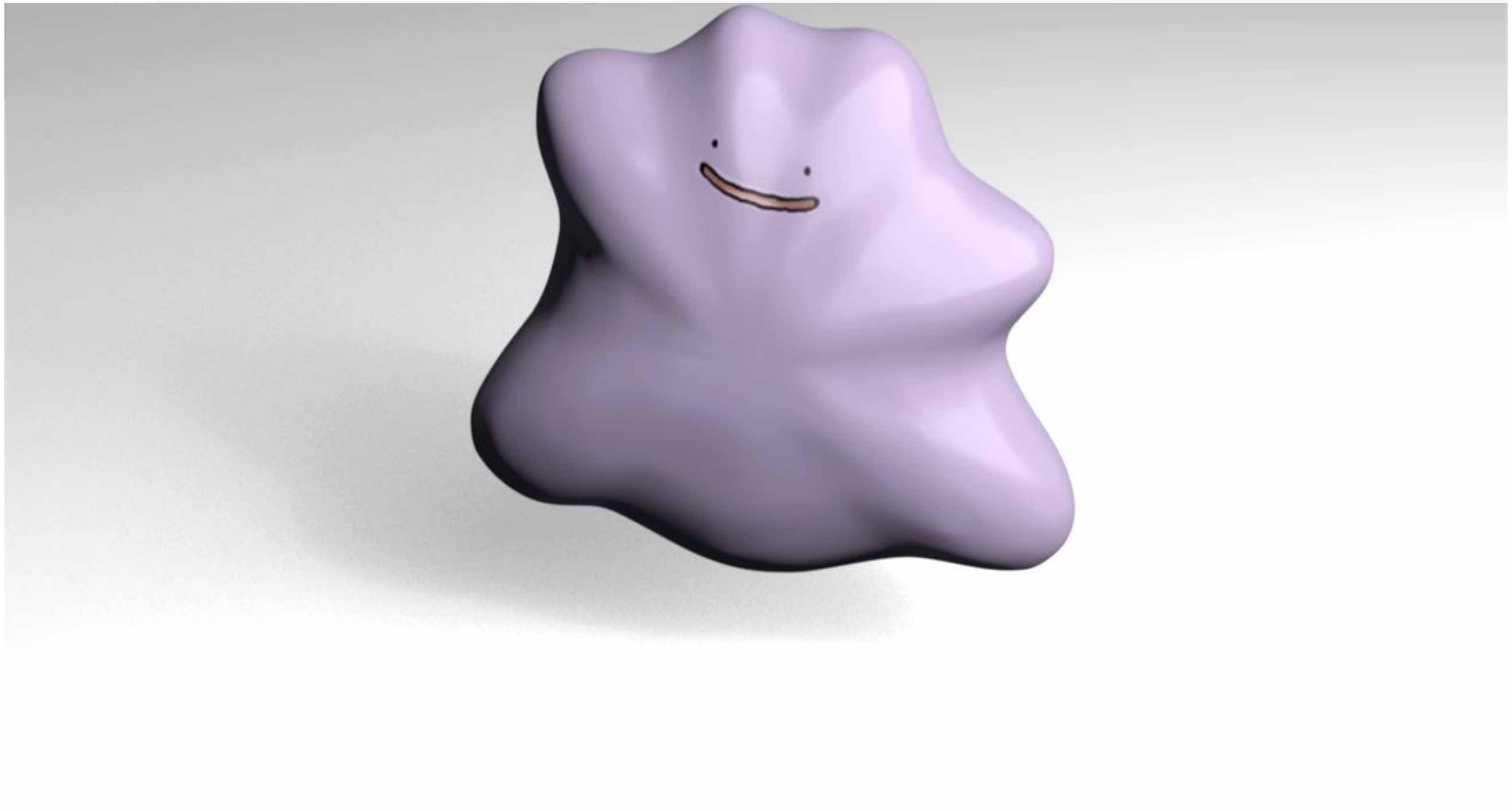
Movie/Animation

AR/VR

Rigid Body v.s. Deformable Body

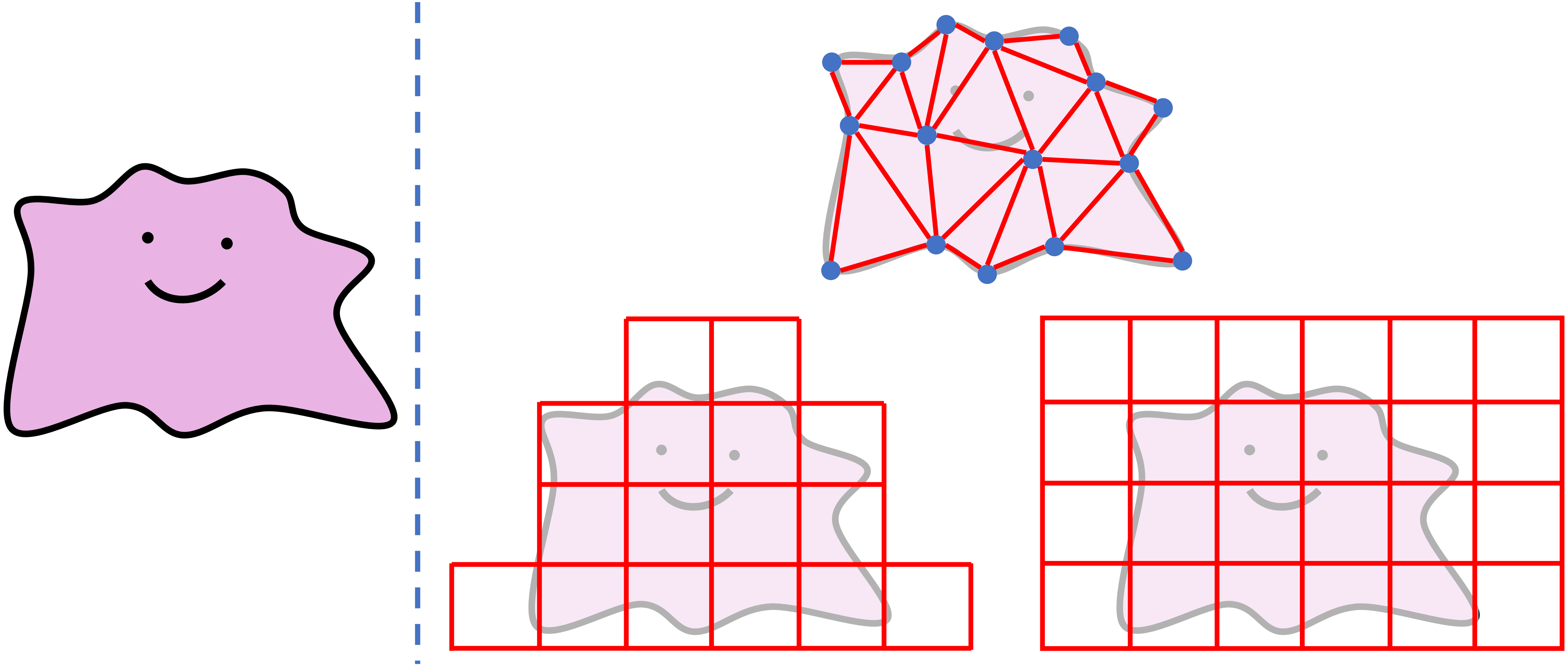


Deformable Body Simulation

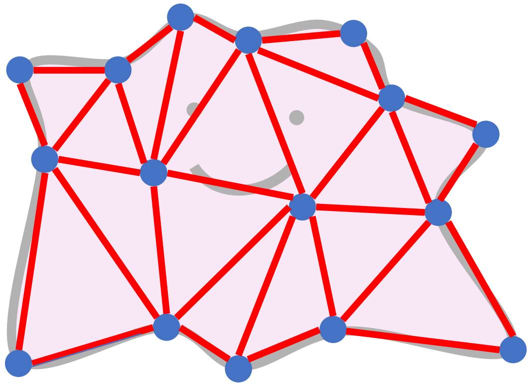


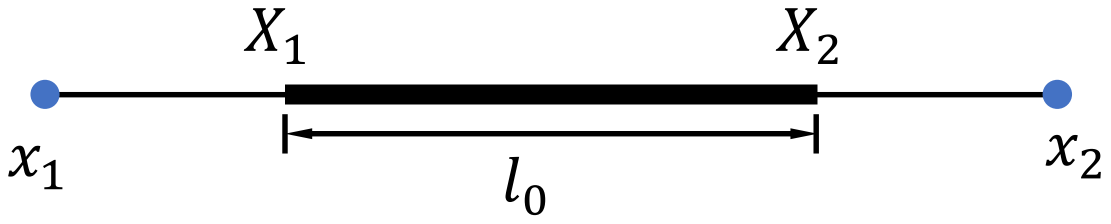
[Liu et al. 2017]

Spatial Discretization



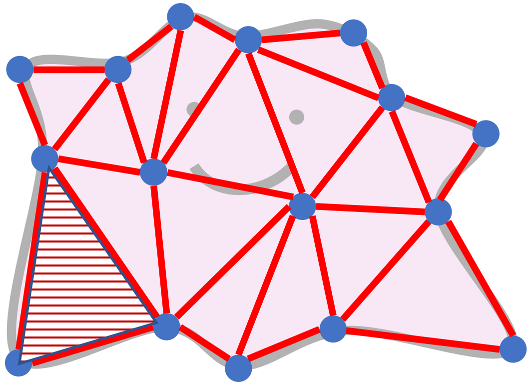
Representation of a Deformable Body

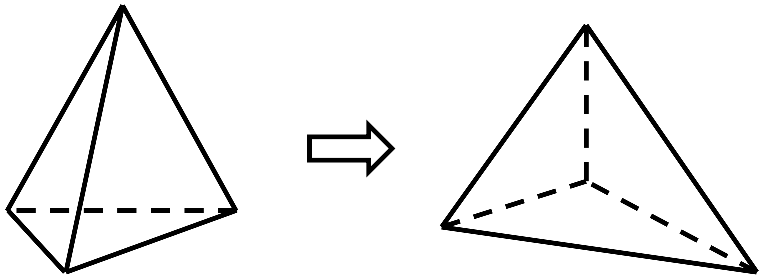




$$E(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1}{2} k (\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| - l_0)^2$$

Representation of a Deformable Body





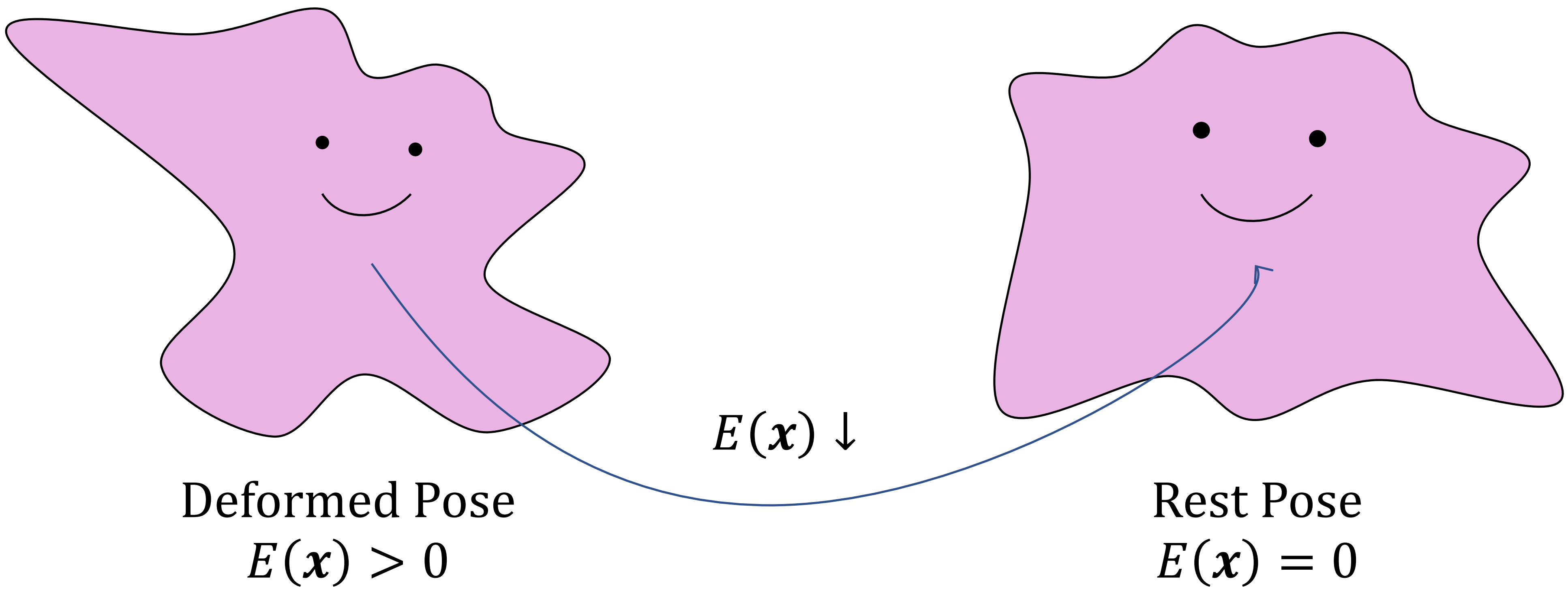
$$\Psi(\mathbf{F}(\boldsymbol{x})) = \mu\|\boldsymbol{\epsilon}(\mathbf{F})\|_F^2 + \frac{1}{2}\lambda tr^2(\boldsymbol{\epsilon}(\mathbf{F}))$$

Energy Density

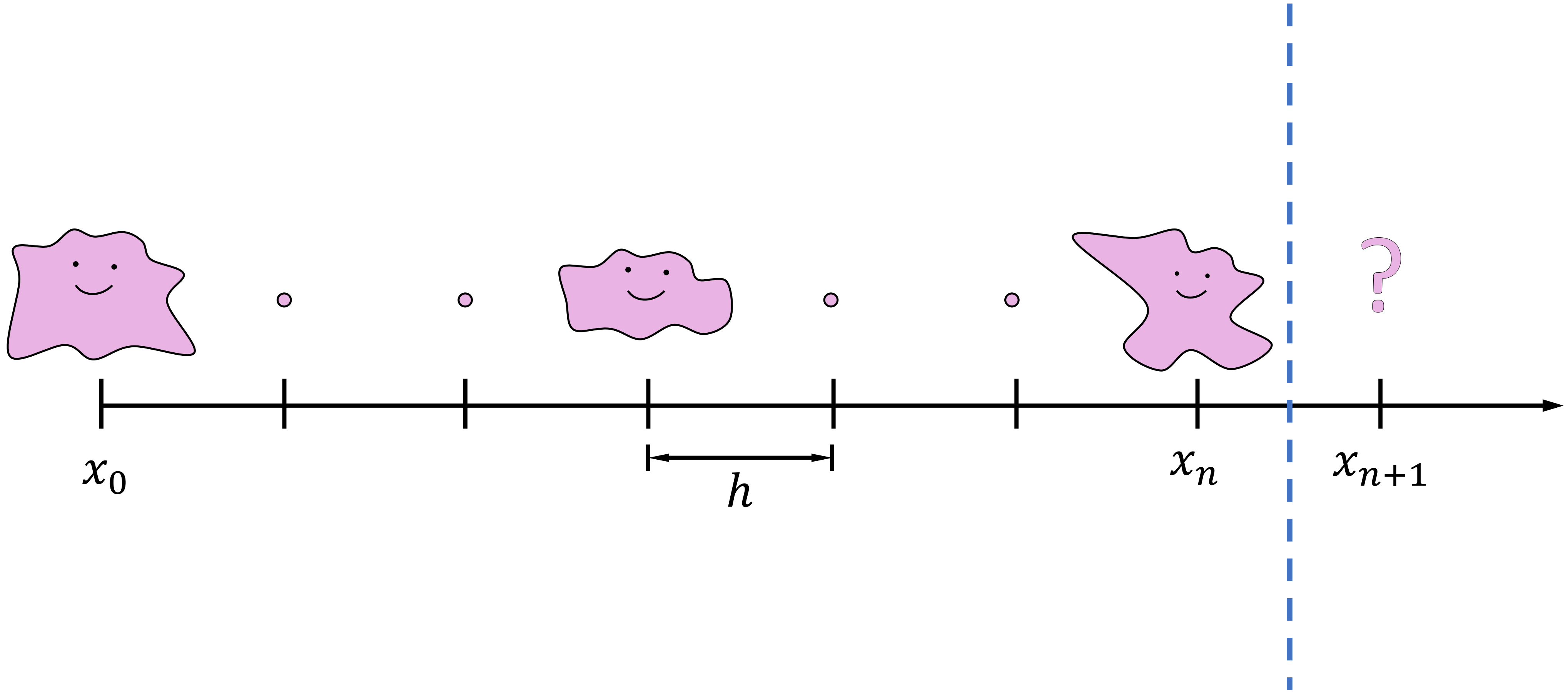
Deformation Gradient

Strain Tensor

Elastic Energy



Temporal Discretization



Newton's 2nd Law of Motion

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

Time integration: Implicit Euler

- $x_{n+1} = x_n + h v_{n+1}$
- $v_{n+1} = v_n + h M^{-1}(f_{int}(x_{n+1}) + f_{ext})$

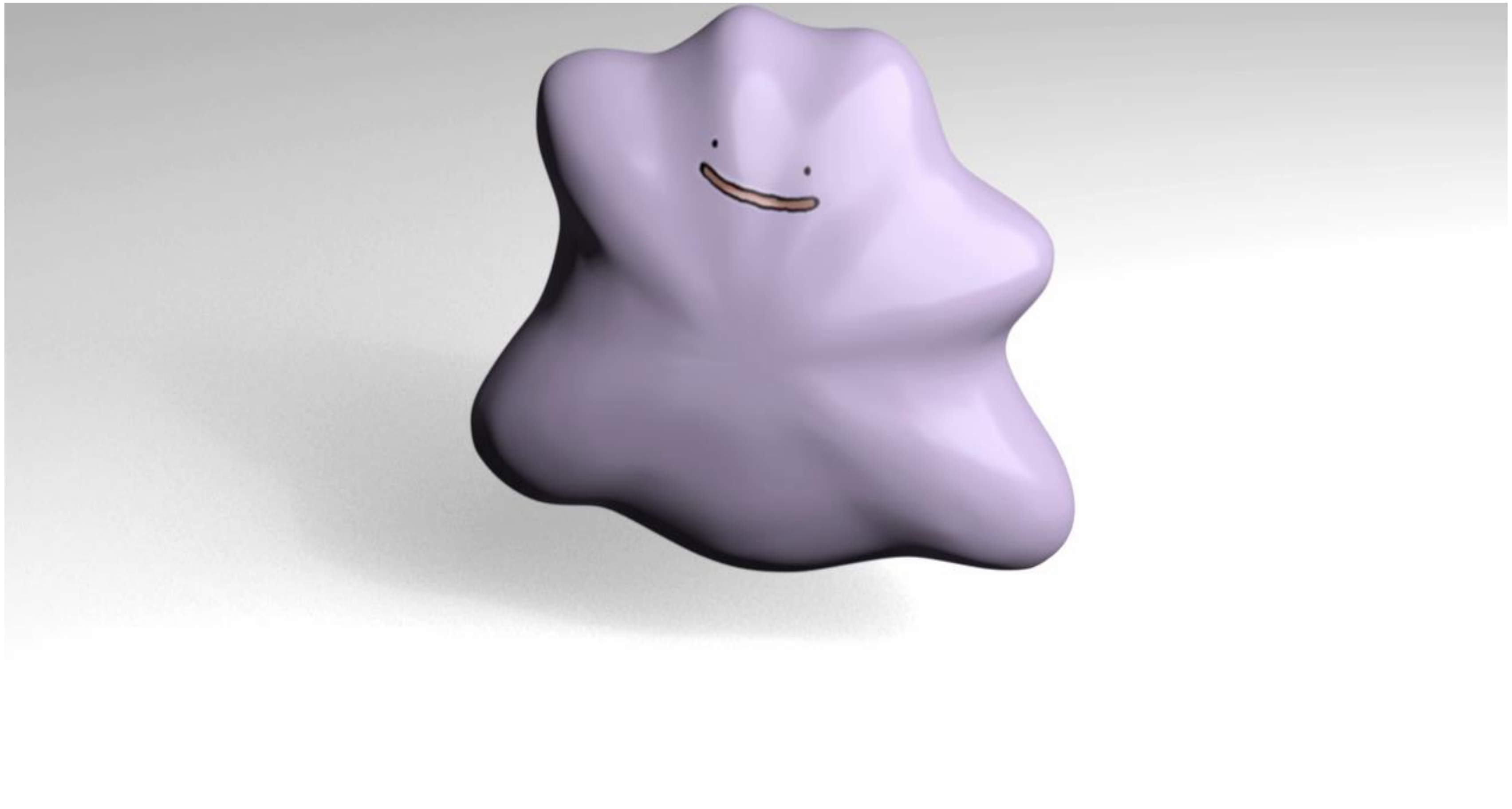
- $x_{n+1} = \underbrace{x_n + h v_n + h^2 M^{-1} f_{ext}}_{y} + h^2 M^{-1} f_{int}(x_{n+1})$

Variational Implicit Euler

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

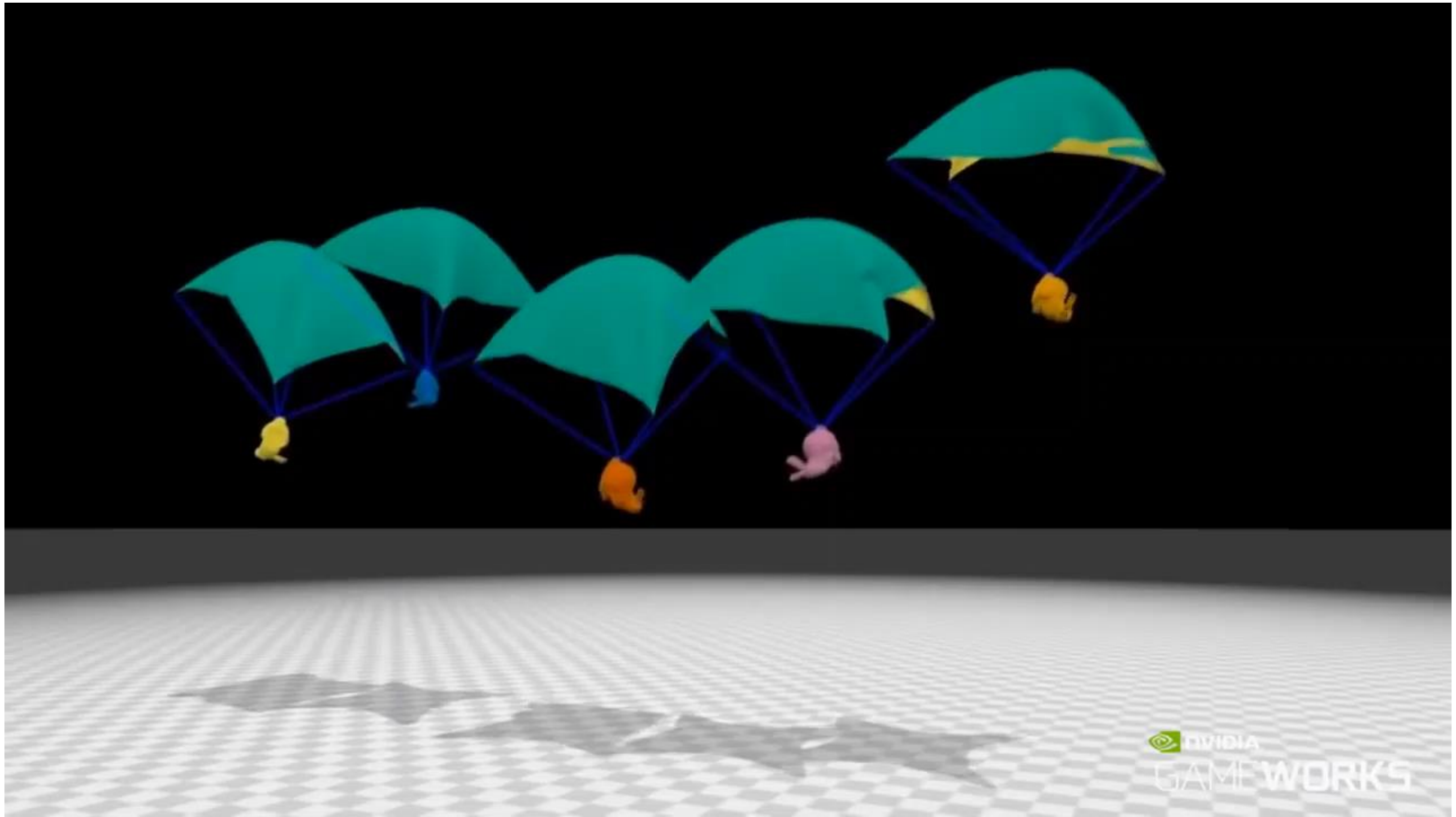
- Pick your favorite optimization tool to solve
  - Gradient Descent / Newton / Quasi-Newton etc…

Deformable Body Simulation



[Liu et al. 2017]

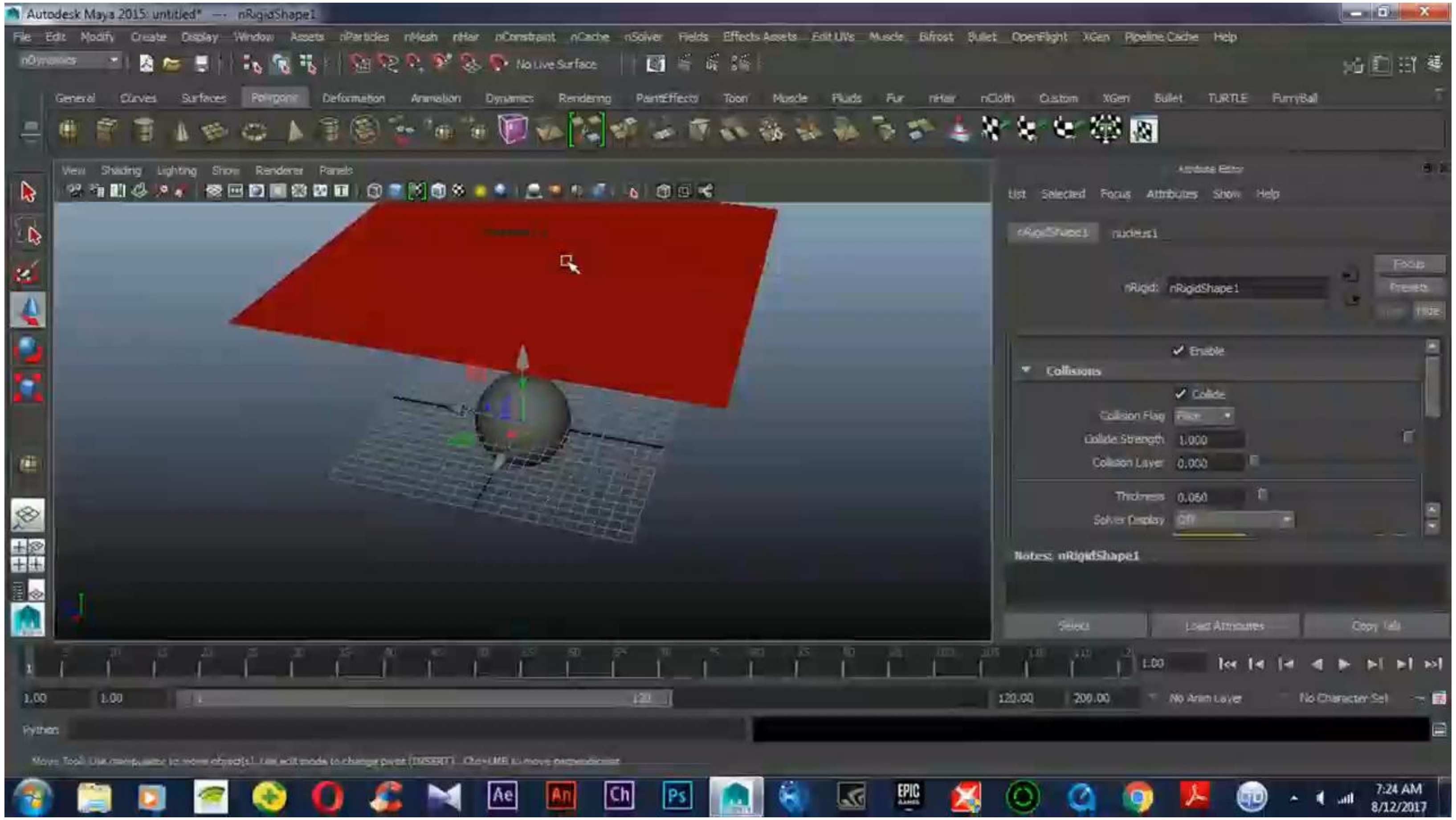
State-of-the-Art Real-time Simulators



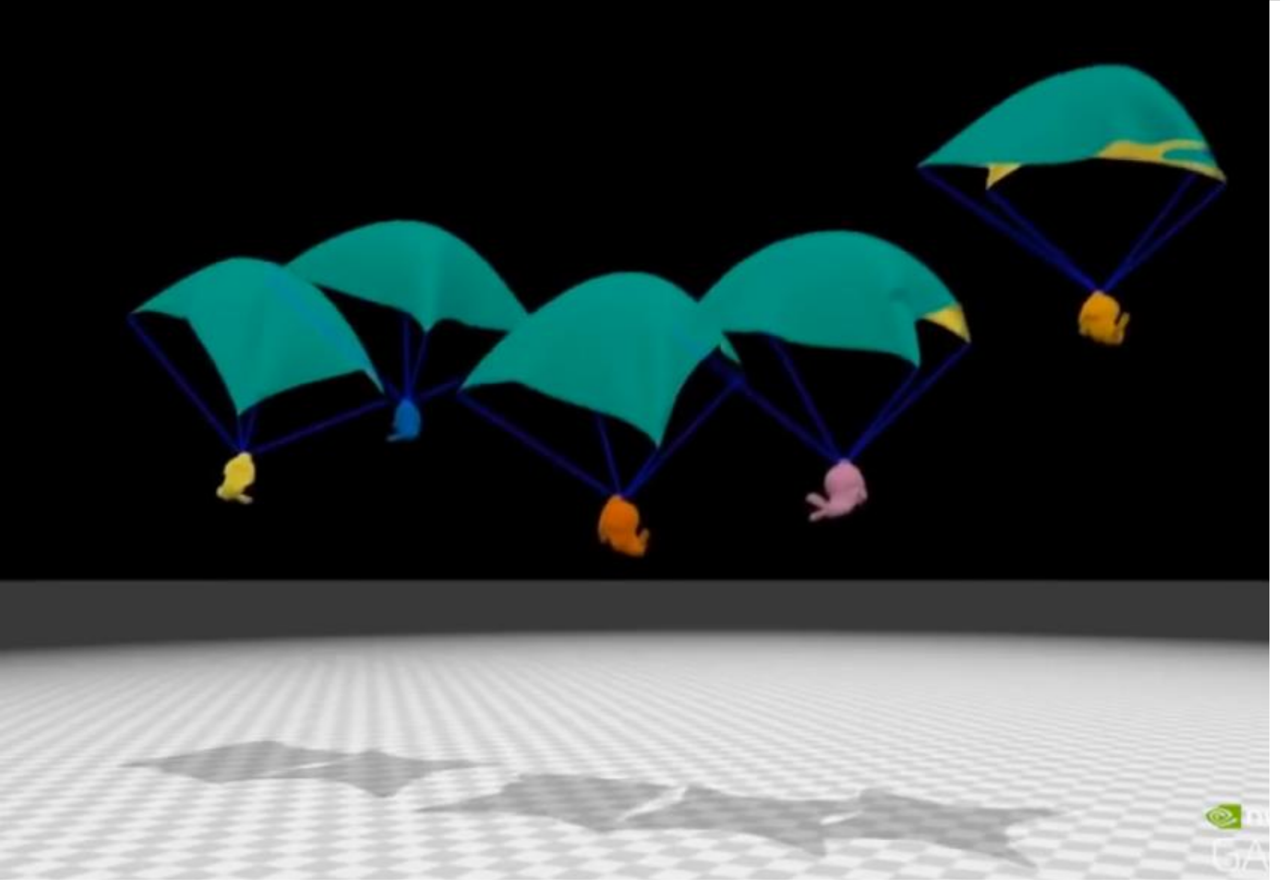
NVIDIA FleX

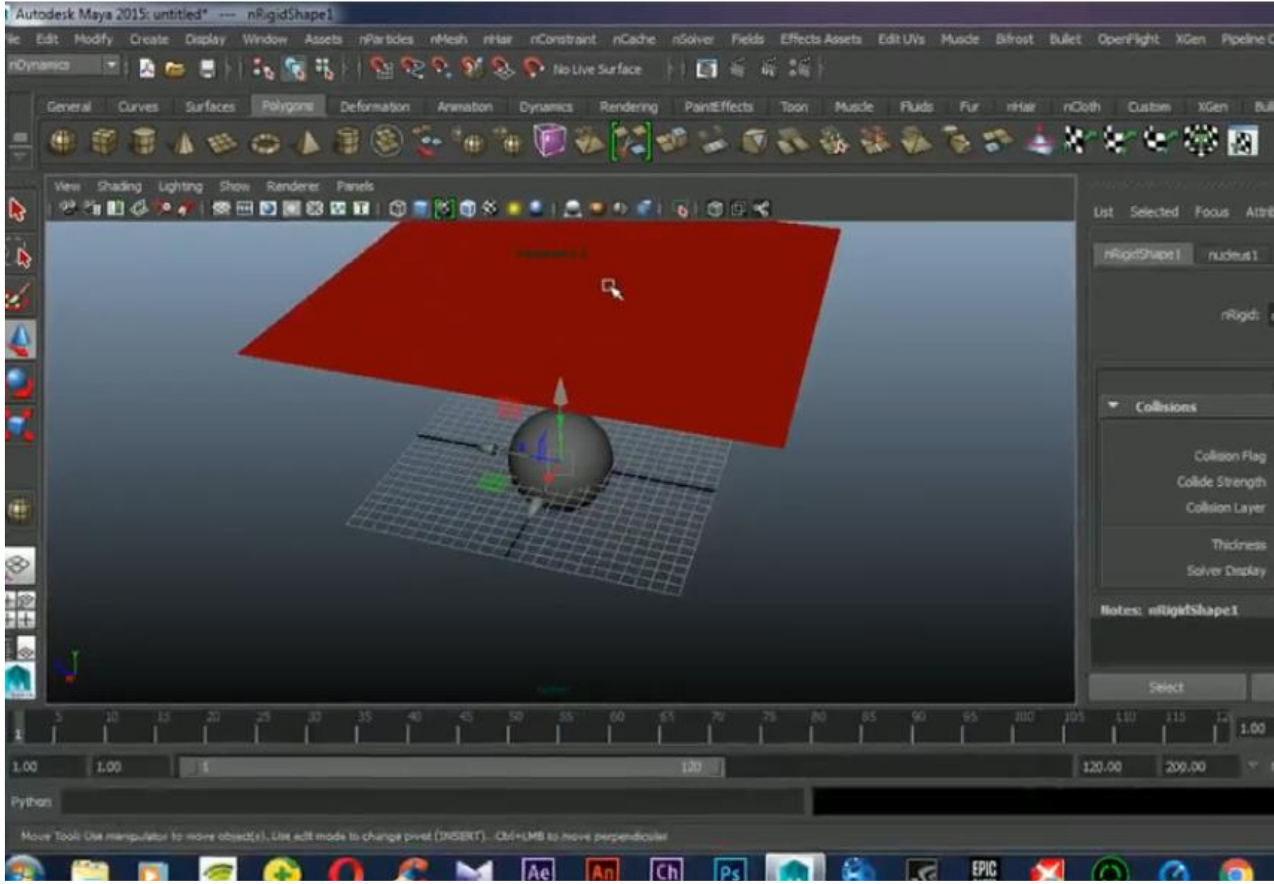
[Video courtesy of NVIDIA]

State-of-the-Art Real-time Simulators



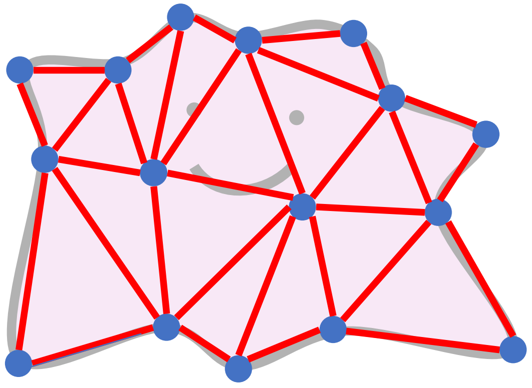
Maya nDynamics [Video courtesy of Pixclue Studios]

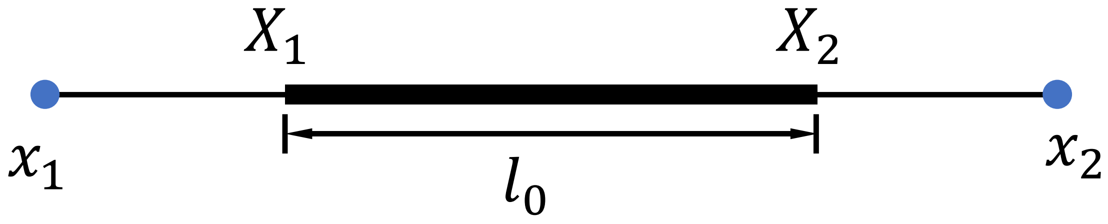




Position Based Dynamics

Position Based Dynamics





$$c(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\|x_1 - x_2\|}{l_0} - 1$$

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

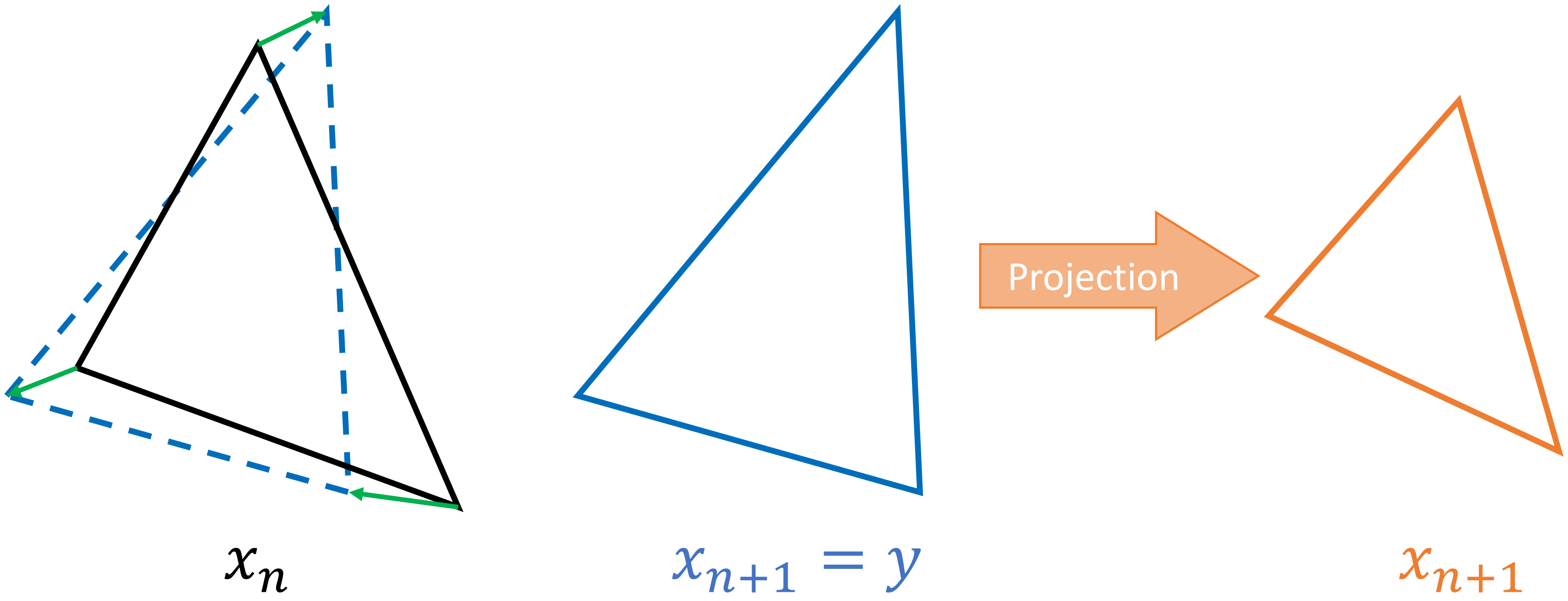
PBD: Pipeline

forall vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
dampVelocities($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
forall vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
forall vertices i **do** generateCollisionConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
loop solverIterations **times**
    projectConstraints($C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
endloop
forall vertices i
    $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
    $\mathbf{x}_i \leftarrow \mathbf{p}_i$
endfor
velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)

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

Project:
Move x_{n+1} to satisfy each constraint

PBD: Pipeline (Illustrated version)



PBD: Projection

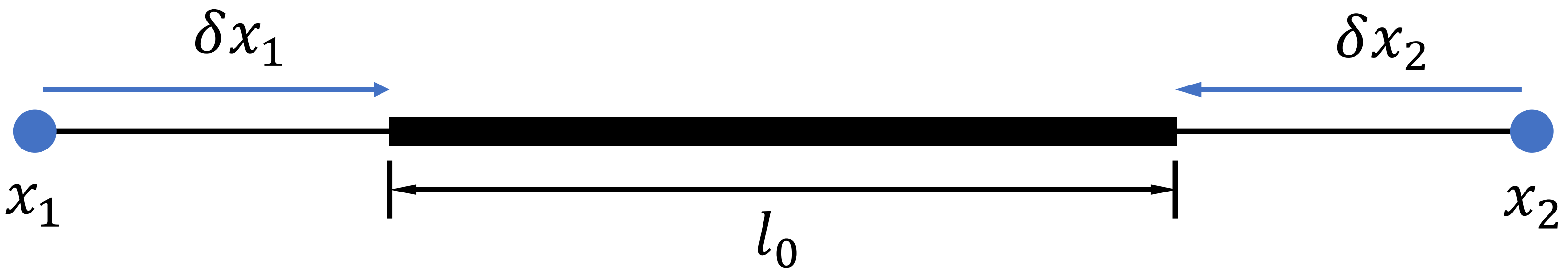
- Find a projection direction δx to:
  - Satisfy $c(x+\delta x) \approx c(x) + \nabla c(x)\delta x = 0$
  - Conserve linear momentum: $\Sigma m_i \delta x_i = 0$
  - Conserve angular momentum: $\Sigma m_i x_i \times \delta x_i = 0$

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 $c_j(x) + \nabla c_j \delta x_j = 0$
  - $\delta \lambda_j = \frac{c_j}{\nabla c_j M_j^{-1} \nabla c_j^T}$

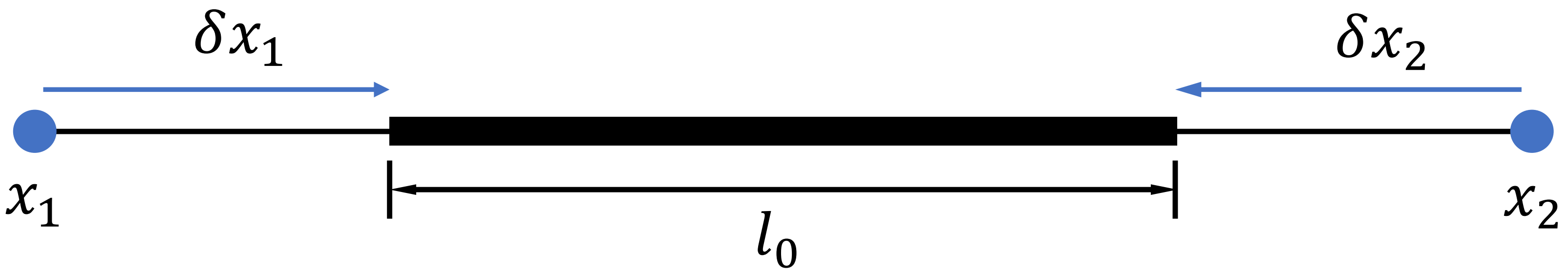
PBD Projection Example



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

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

PBD Projection Example

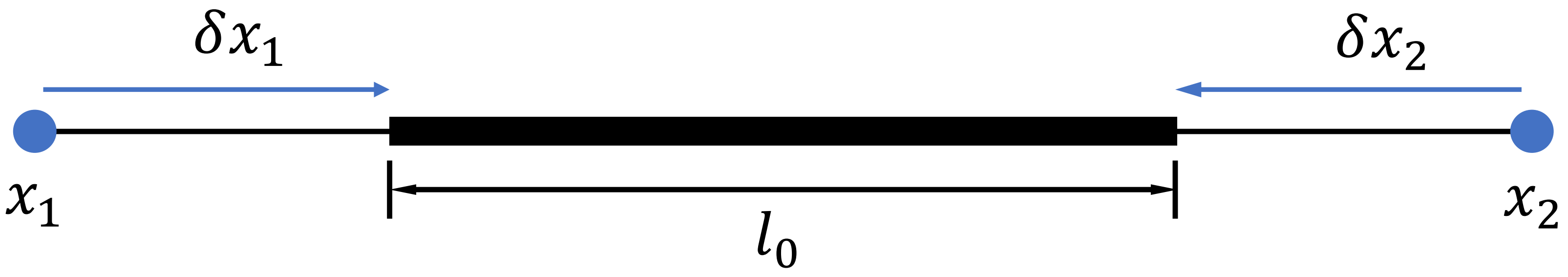


$$\delta x = -\mathbf{M}^{-1}\nabla c^T \delta\lambda$$

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

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

PBD Projection Example



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

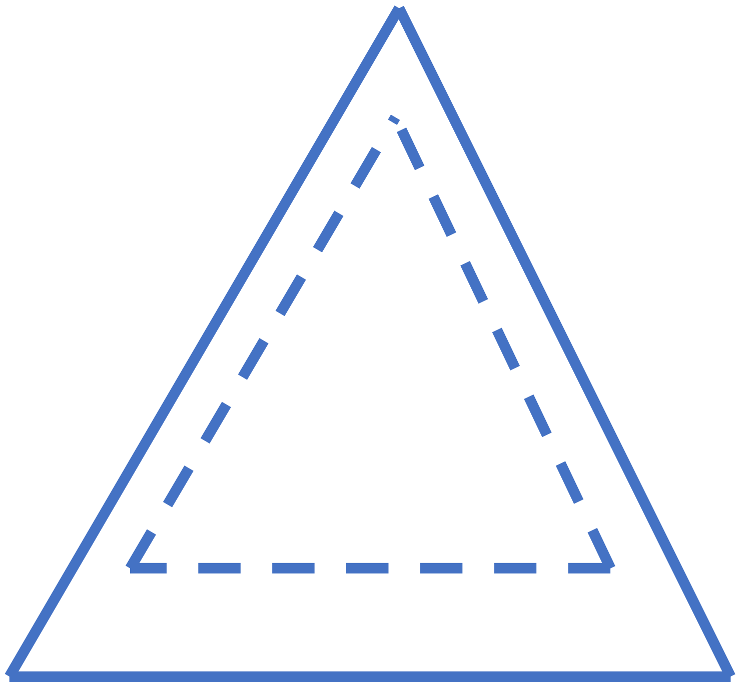
$$\delta x_2 = +\frac{m_2^{-1}}{m_1^{-1} + m_2^{-1}}(\|x_1 - x_2\| - l_0)\frac{x_1 - x_2}{\|x_1 - x_2\|}$$

PBD: Pipeline

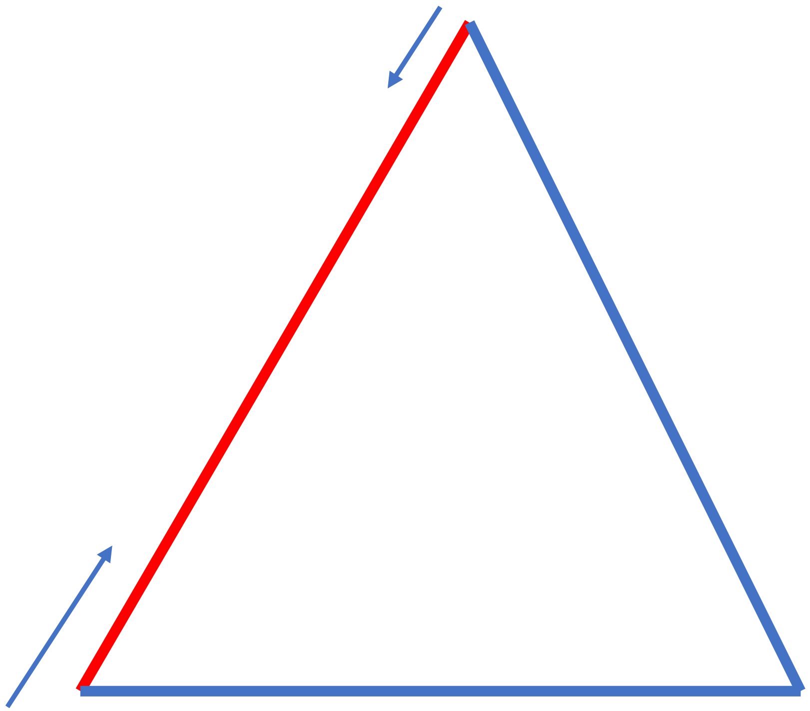
forall vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$
dampVelocities($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
forall vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
forall vertices i **do** generateCollisionConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
loop solverIterations **times**
    projectConstraints($C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
endloop
forall vertices i
    $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
    $\mathbf{x}_i \leftarrow \mathbf{p}_i$
endfor
velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)

Project:
Move x_{n+1} to satisfy each constraint

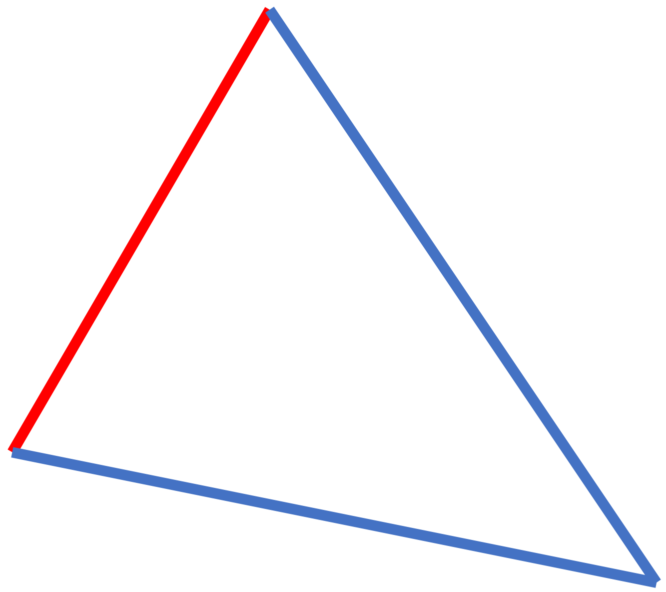
Iteration Strategy: Gauss-Seidel v.s. Jacobi



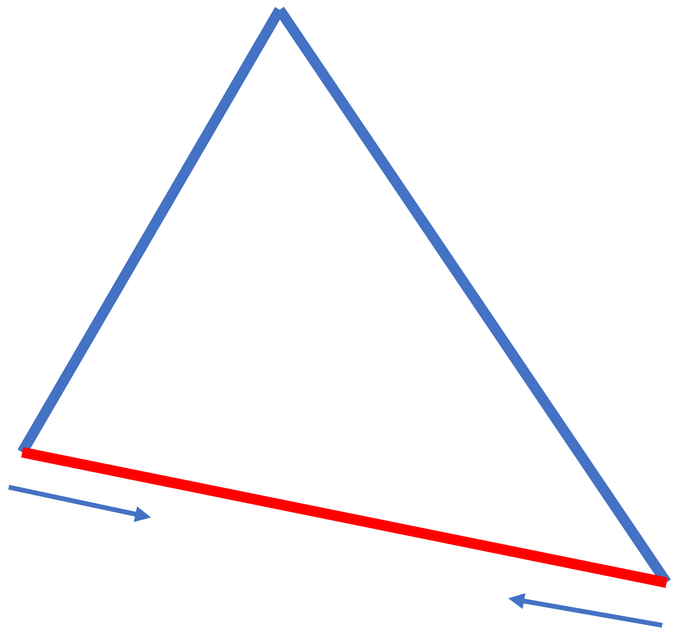
Gauss-Seidel



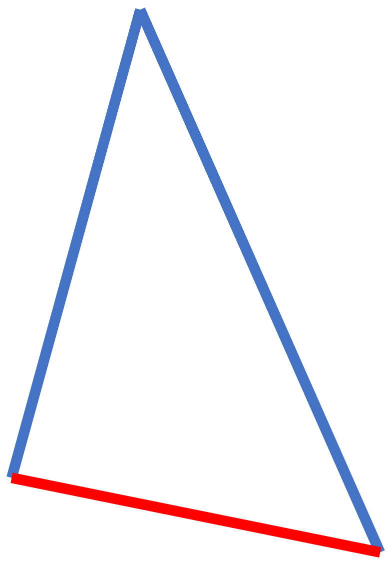
Gauss-Seidel



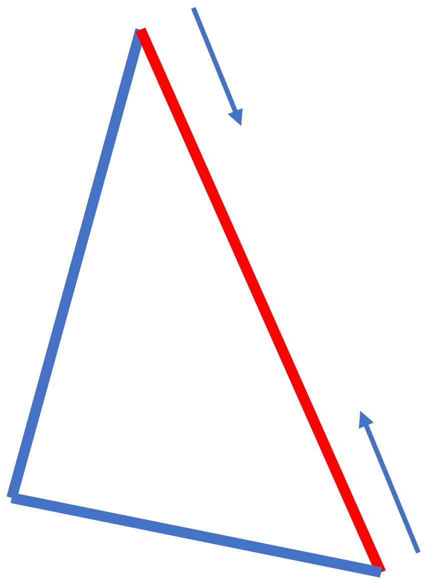
Gauss-Seidel



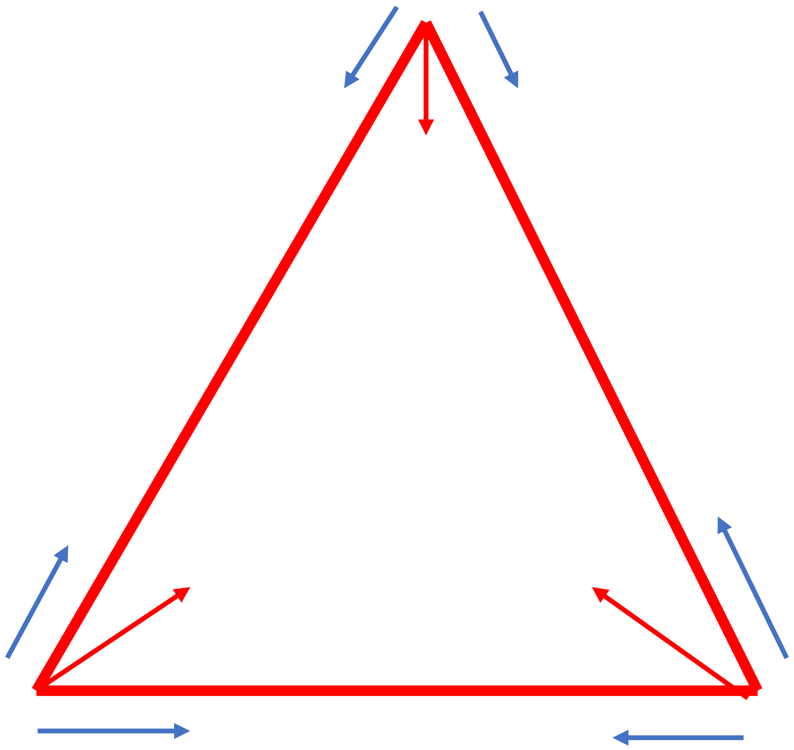
Gauss-Seidel



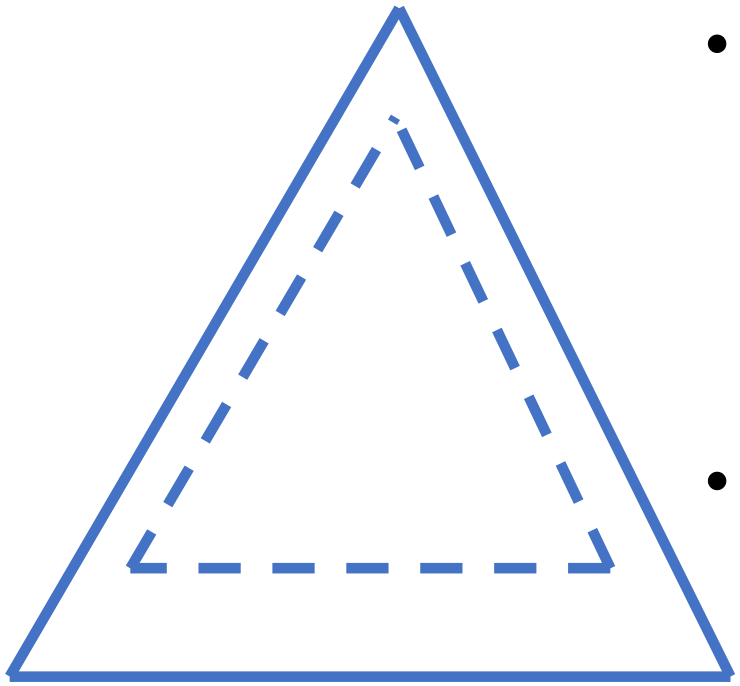
Gauss-Seidel



Jacobi

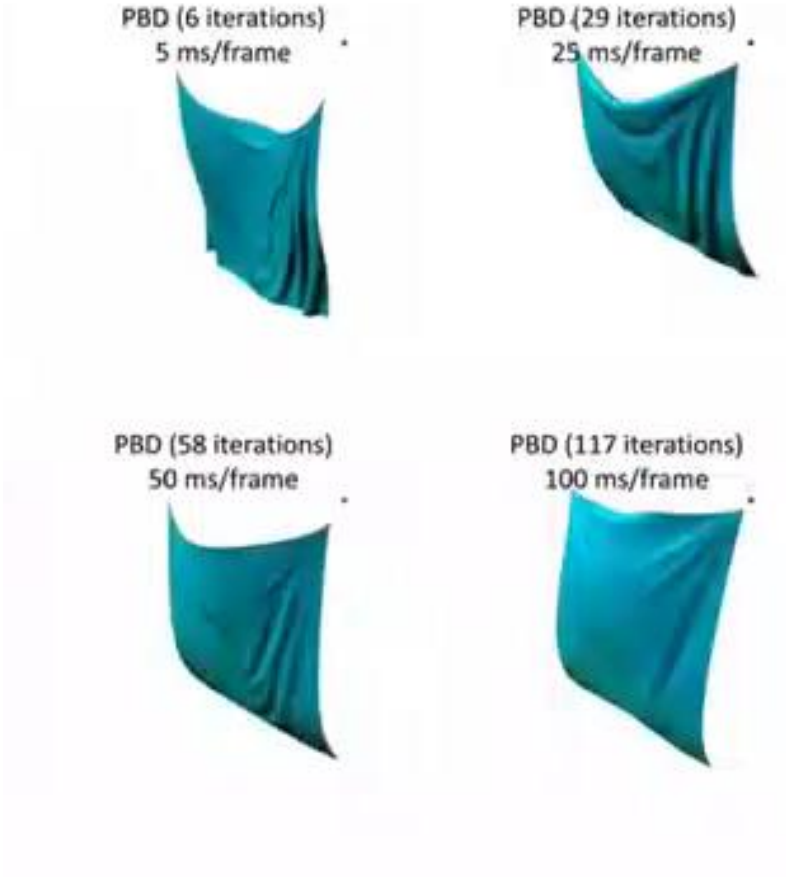


Gauss-Seidel v.s. Jacobi



- Gauss-Seidel
  - \+ Converges faster
  - \- Hard to parallelize <-- Colored GS [Fratarcangeli et al. 2016]
  - \- May break the symmetry <-- Symmetric GS
- Jacobi
  - \+ Easy to parallelize
  - \- Converges slower <-- Chebyshev Acceleration [Wang 2015]
  - \- Less stable <-- Under Relaxation

Problems



[Liu et al. 2013]

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

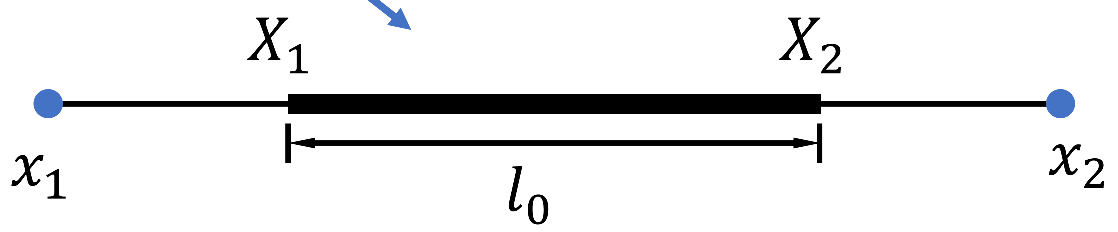
Variational Implicit Euler

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

- What if $E(x)$ is (almost) infinitely stiff?

Constraint-based Variational Implicit Euler

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



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

Constraint-based Variational Implicit Euler

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

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

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

Stiffness Matrix

Compliance Matrix

Variational Implicit Euler with compliant constraints

- $\min_x \frac{1}{2h^2}\|x-y\|_M^2 + \frac{1}{2} c(x)^T \boldsymbol{\alpha}^{-\mathbf{1}} c(x)$

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

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

- 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]

Newton-Raphson method

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

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

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

Hard Constraints

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

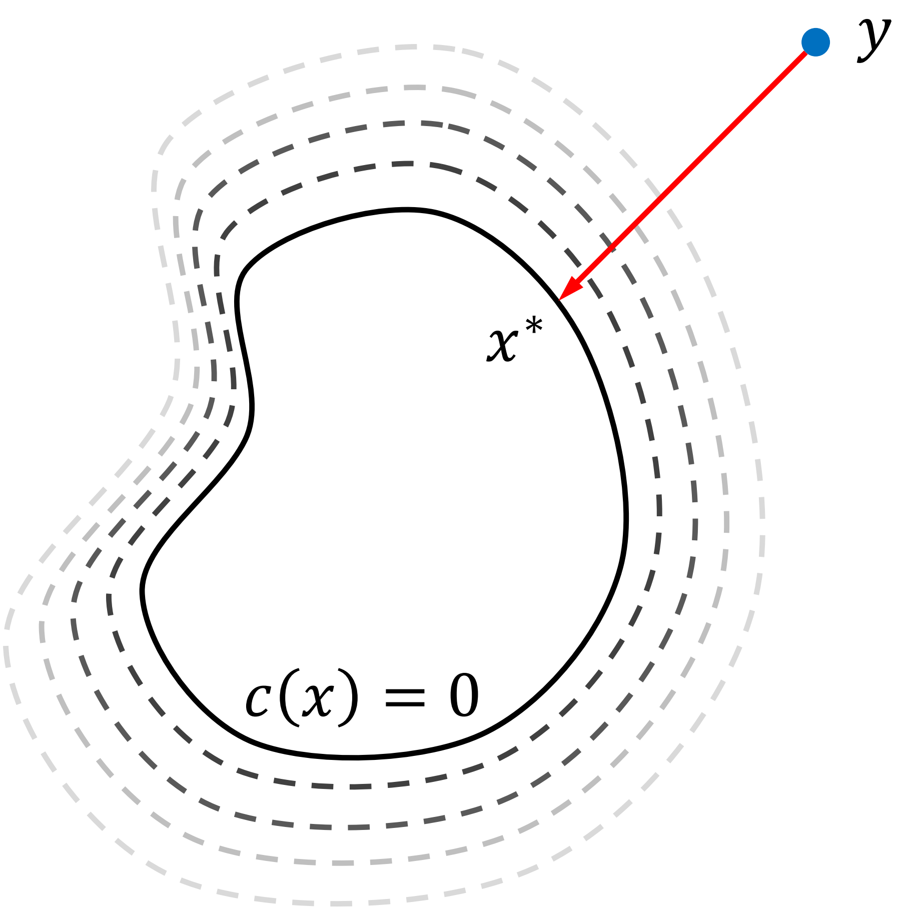
- $\boldsymbol{\alpha = 0}$

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

$\Longleftrightarrow$

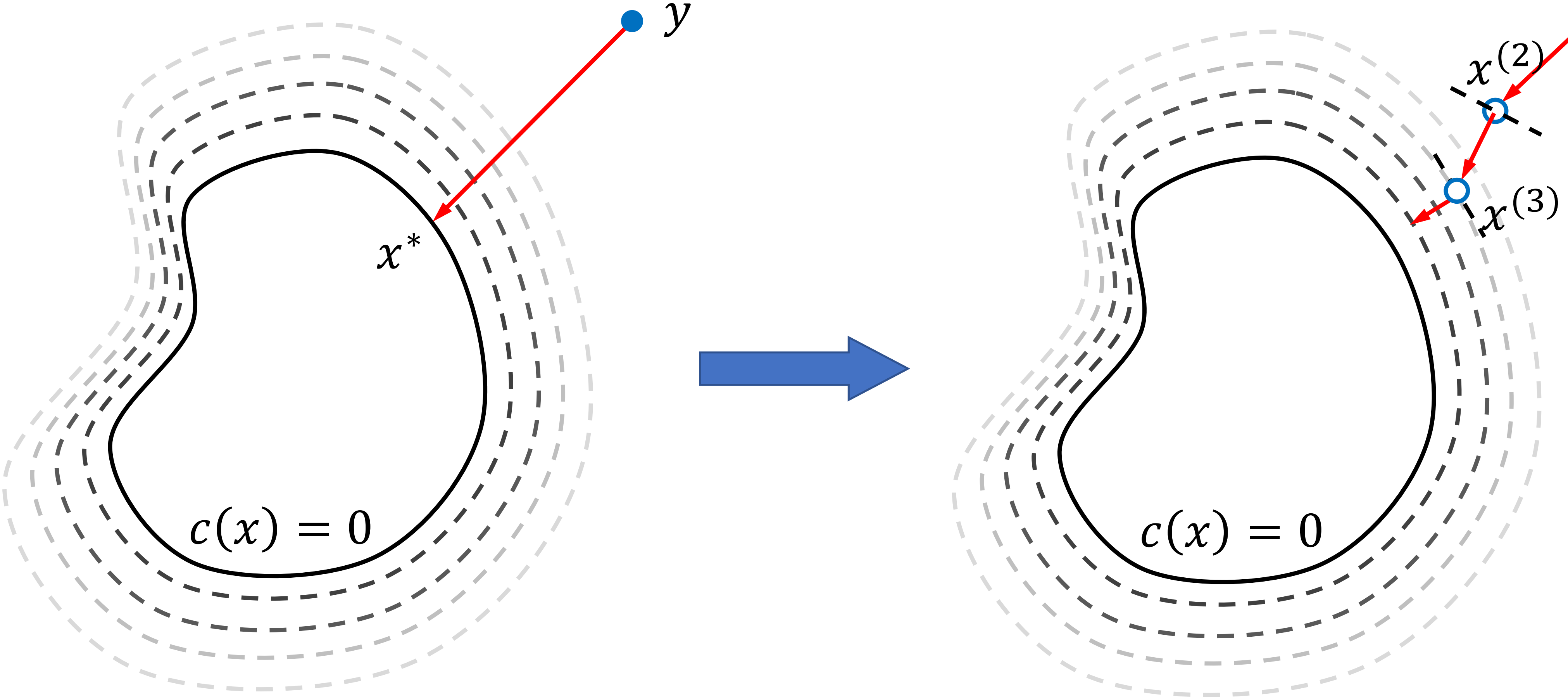
$$\min_x \frac{1}{2h^2}\|x-y\|_M^2$$
$$s.t.\, c(x) = 0$$

Hard Constraints: Geometric Interpretation

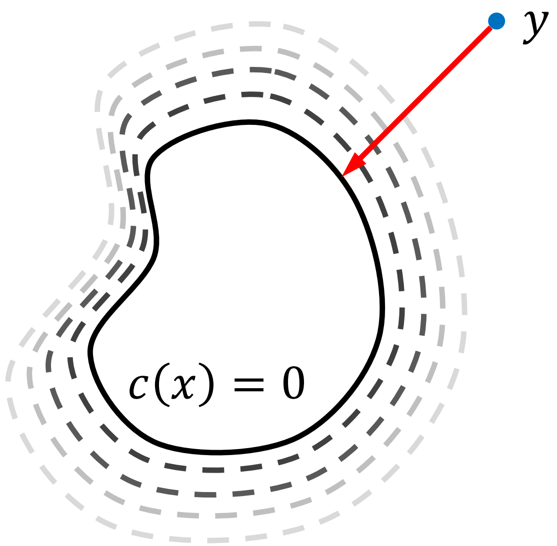


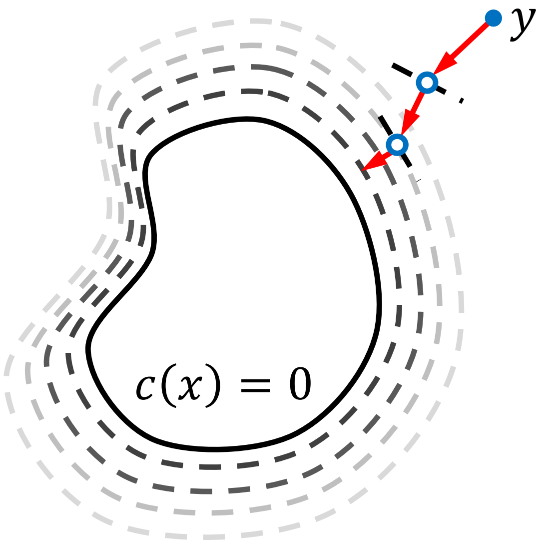
$$\min_x \frac{1}{2h^2} \|x - y\|_M^2$$
$$s.t.\, c(x) = 0$$

Step and Project (SAP) [Goldenthal et al. 2007]



Step and Project (SAP) [Goldenthal et al. 2007]





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

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

SAP: Update

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

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

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

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

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

SAP: Update

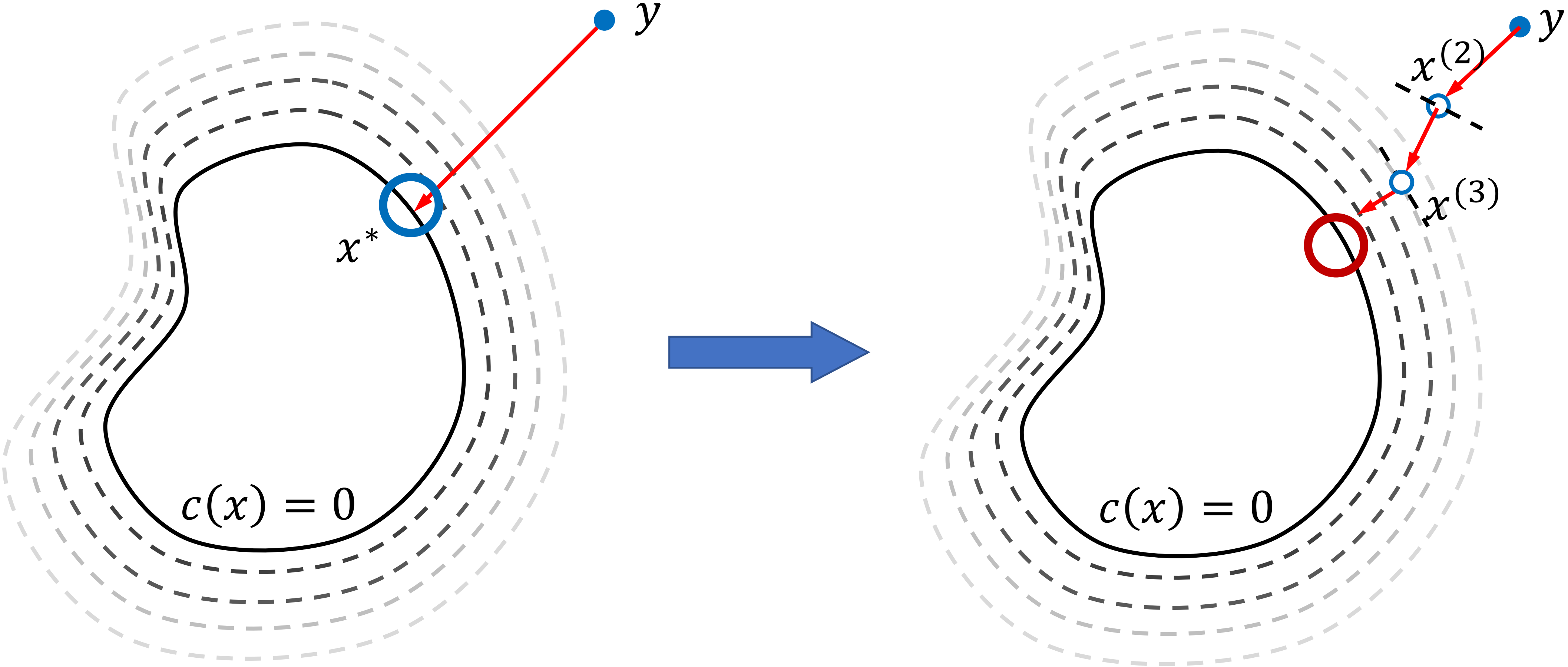
$$\frac{M}{h^2}\left(x - x^{(k)}\right) + \nabla_x c\left(x^{(k)}\right)^T \lambda = 0$$
$$c\left(x^{(k)}\right) + \nabla_x c(x^{(k)})(x - x^{(k)}) = 0$$

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

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

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

SAP: Solution Drifting



SAP vs. the Exact Solution

SAP

Exact Solution

PBD vs. SAP

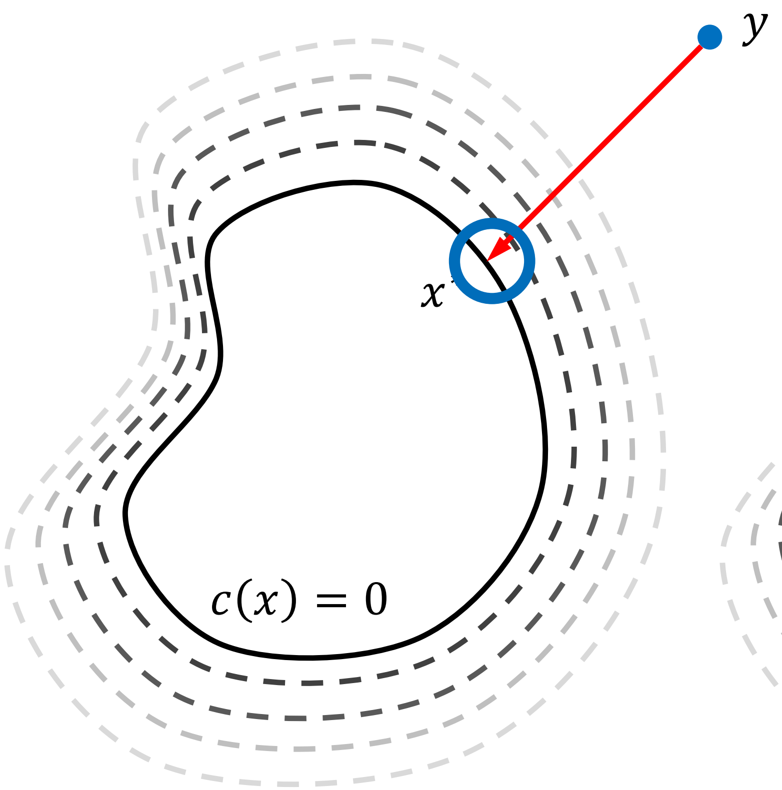
- For each constraint j:
  - Compute $\delta\lambda_j$:$\delta\lambda_j = \frac{c_j}{\nabla c_j\, M_j^{-1} \nabla c_j^T}$
  - Compute δx_j: $\delta x_j = -M_j^{-1} \nabla c_j^T \delta\lambda_j$
  - Commit: $x = x + \delta x_j$

PBD (G-S)

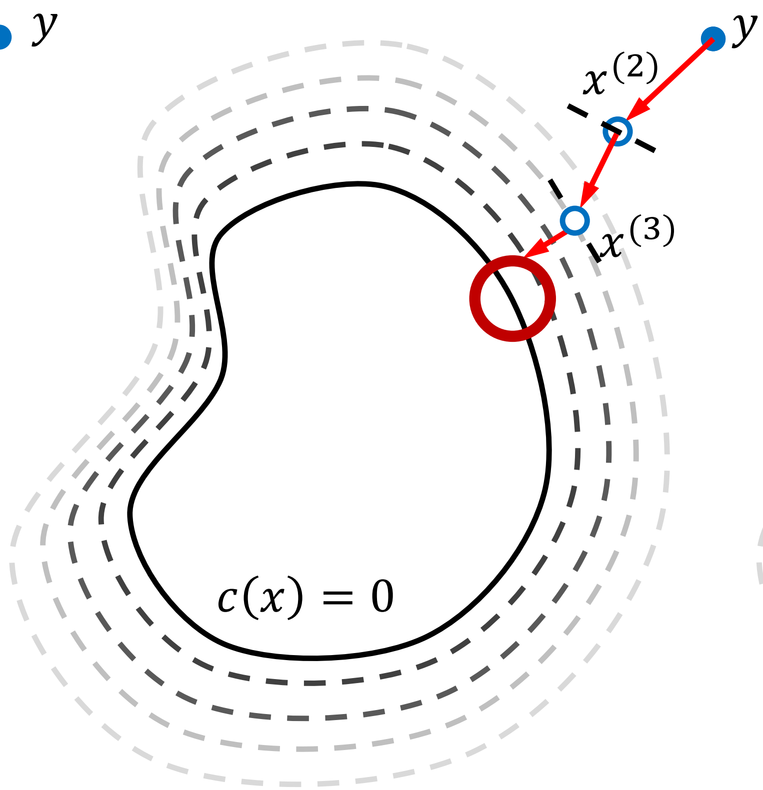
- $\delta\lambda = \frac{1}{h^2}\left(\nabla c\, M^{-1} \nabla c^T\right)^{-1} c$
- $\delta x = -h^2 M^{-1} \nabla c^T \delta\lambda$
- $x = x + \delta x$

SAP

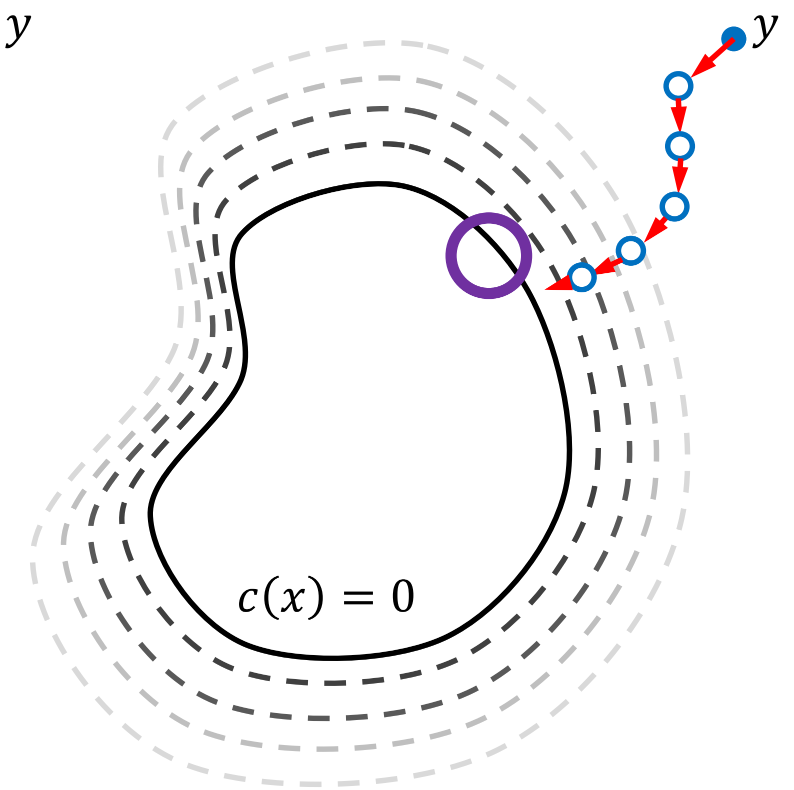
PBD v.s. SAP: Geometric Interpretation



Ground Truth



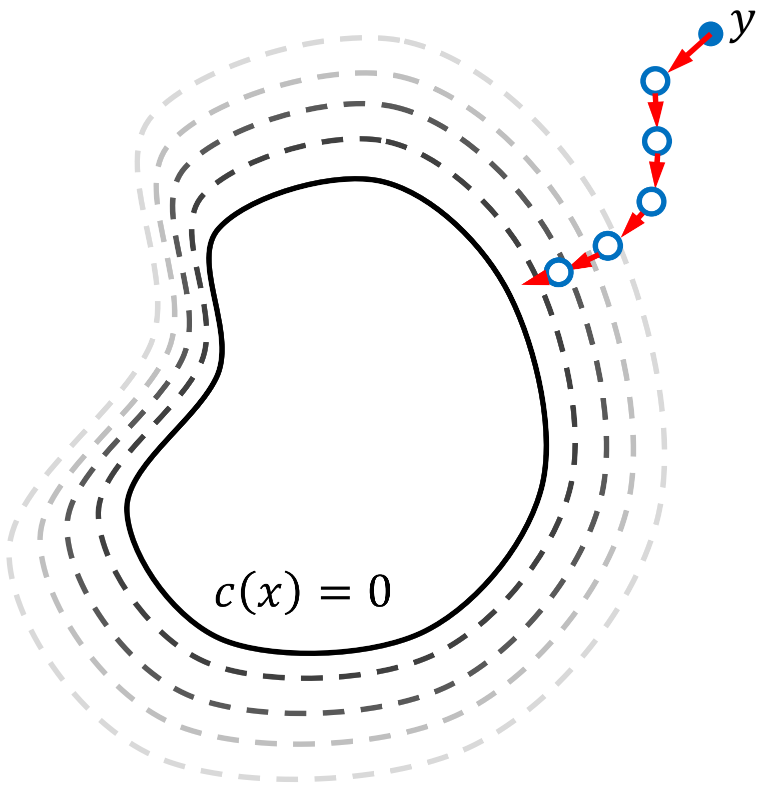
SAP



PBD

PBD Problems Visualized

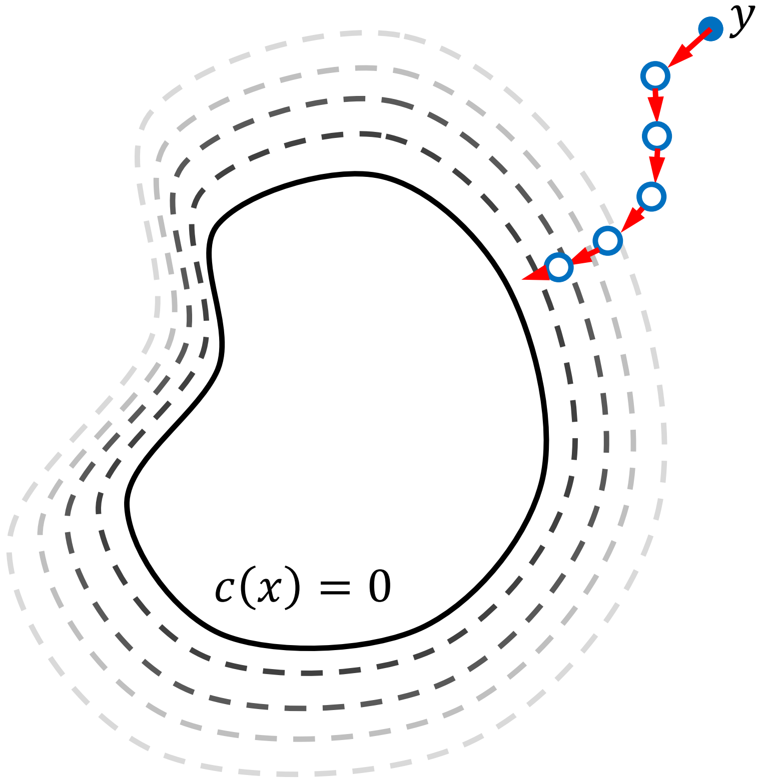
- Solution Drifts
- Non-physically-based
- Hard to Control



PBD

PBD Problems Visualized

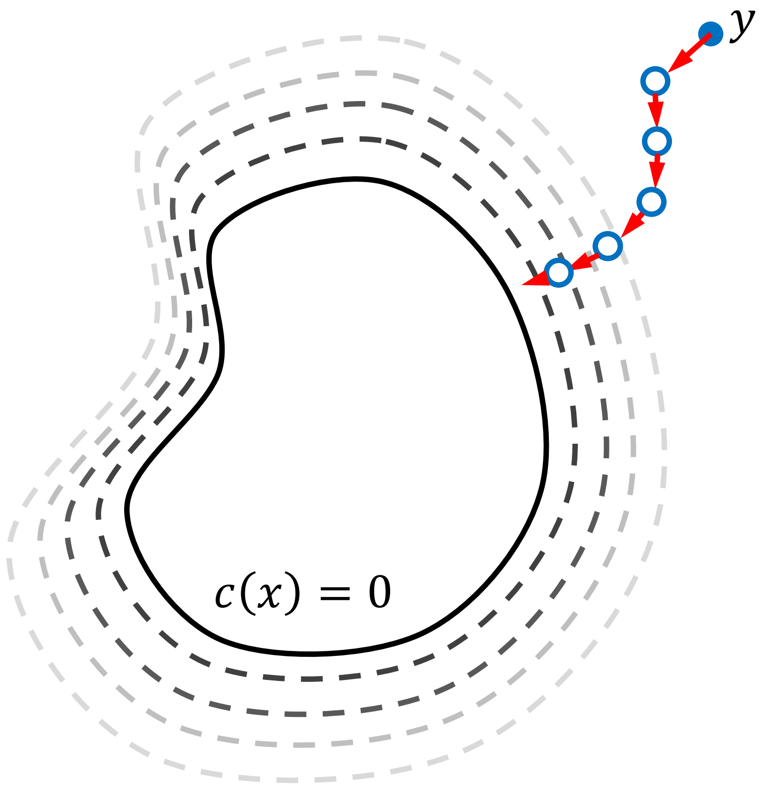
- Solution Drifts
- Non-physically-based
  - Stiffness depends on iteration count
  - $y = x_n + hv_n + h^2 M^{-1} f_{ext}$
- Hard to Control



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

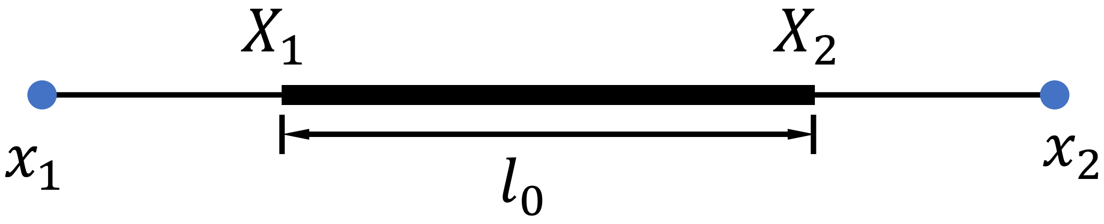
$$\frac{M}{h^2}(x-y)+\nabla_x c(x)^T \lambda = 0$$
$$c(x) - \boldsymbol{\alpha}\lambda = 0$$

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

Compliant Constraints

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

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



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

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

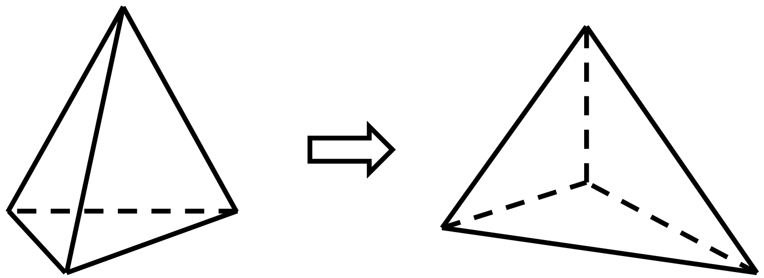
$$k = \frac{2\mu A}{l_0}$$

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

Compliant Constraints

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

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



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

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

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

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

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

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

Solve Compliant Constraints with Newton

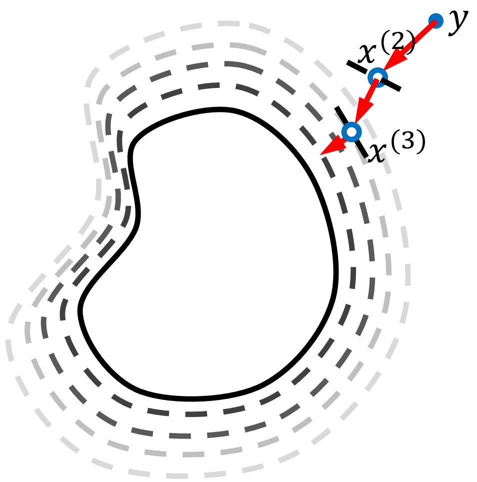
$$\begin{bmatrix} \dfrac{M}{h^2} + \sum_{j=1}^{m} \lambda_j^{(k)} \nabla^2 c_j & \nabla c^T \\ \nabla c & -\boldsymbol{\alpha} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \dfrac{M}{h^2}\left(x^{(k)} - y\right) + \nabla c^T \lambda^{(k)} \\ c\left(x^{(k)}\right)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

SAP Assumption:

$$y \to x^{(k)}$$
$$\nabla^2 c = 0$$

SAP Initialization:

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



Solve Compliant Constraints with SAP

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

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

Solve Compliant Constraints with XPBD [Macklin et al. 2016]

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

$For\ each\ constraint$: j

Compute: $\delta \lambda_j = \frac{1}{h^2}\left(\nabla c_j\, M_j^{-1} \nabla c_j^T + h^2 \boldsymbol{\alpha}_j\right)^{-1} \left(c_j\left(x^{(k)}\right) - \boldsymbol{\alpha}_j \lambda_j^{(k)}\right)$

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

PBD v.s. XPBD

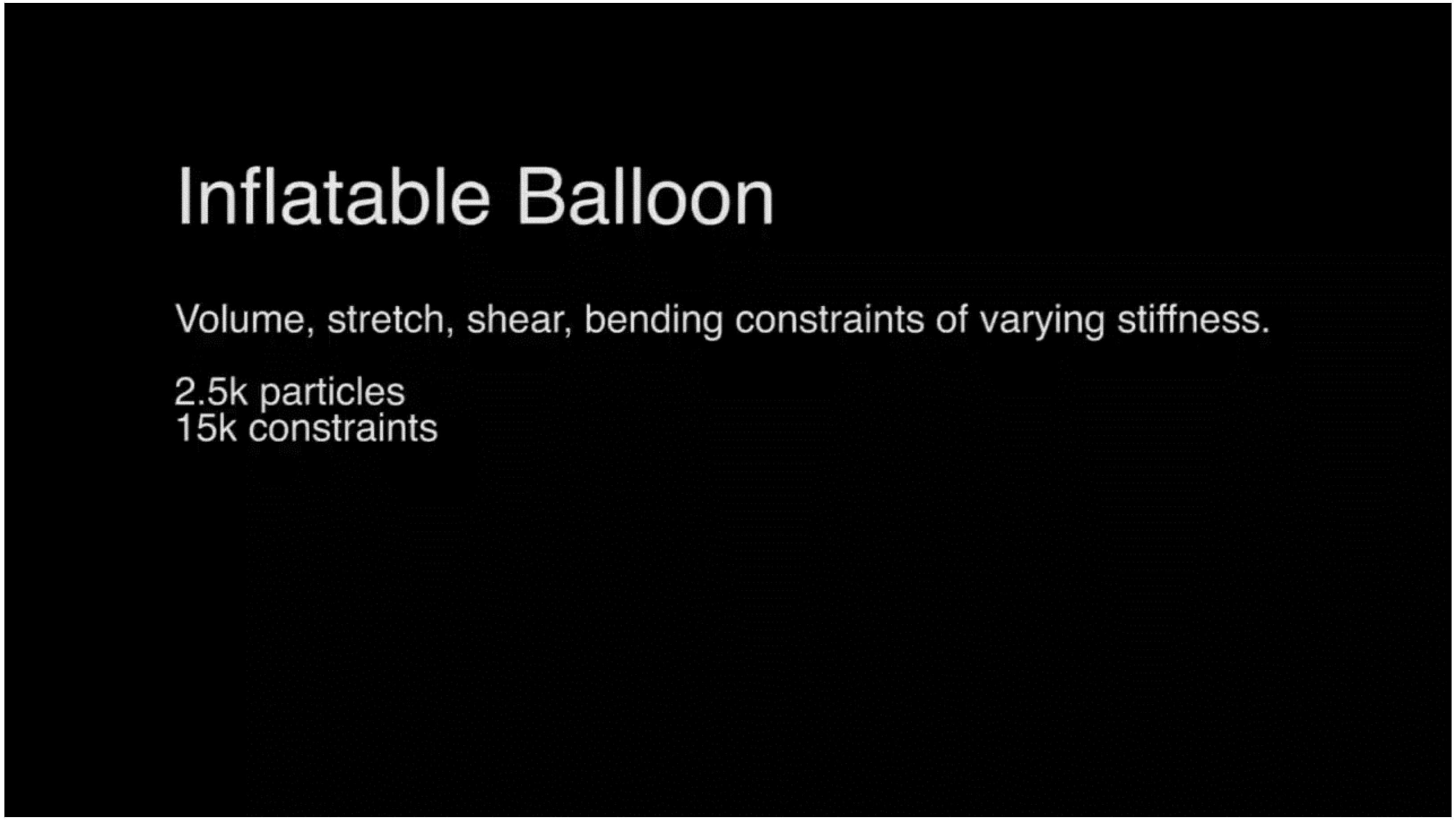
- For each constraint j:
  - Compute $\delta\lambda_j$:$\delta\lambda_j = \frac{1}{h^2}\frac{c_j}{\nabla c_j\, M_j^{-1}\nabla c_j^T}$
  - Commit: $\lambda_j = \delta\lambda_j$
  - Compute δx_j: $\delta x_j = -h^2 M_j^{-1}\nabla c_j^T\lambda_j$
  - Commit: $x = x + \delta x_j$

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



[Macklin et al. 2016]

What is left out?

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

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

Schur Complement of the Upper-left Block

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

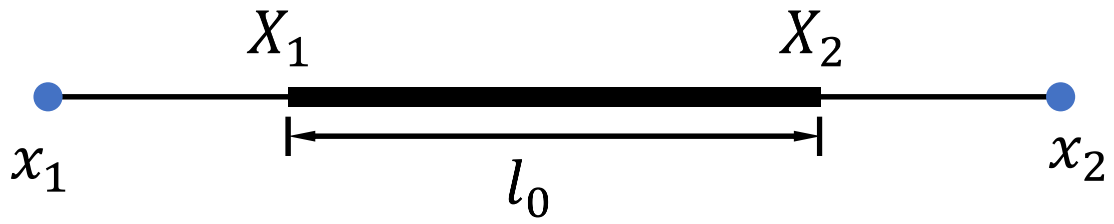
$$\frac{M}{h^2} + \nabla c^T \boldsymbol{\alpha}^{-1} \nabla c + \sum_{j=1}^{m} \lambda_j^{(k)} \nabla^2 c_j$$

Geometric Stiffness

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

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

Geometric Stiffness: Example



$$c = \frac{\|x_1 - x_2\|}{l_0} - 1 \qquad E = \frac{1}{2}\alpha^{-1}c^2$$

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

Geometric
Stiffness

Stable Constrained Dynamics [Tournier et al. 2015]

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

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

Conclusion

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

	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

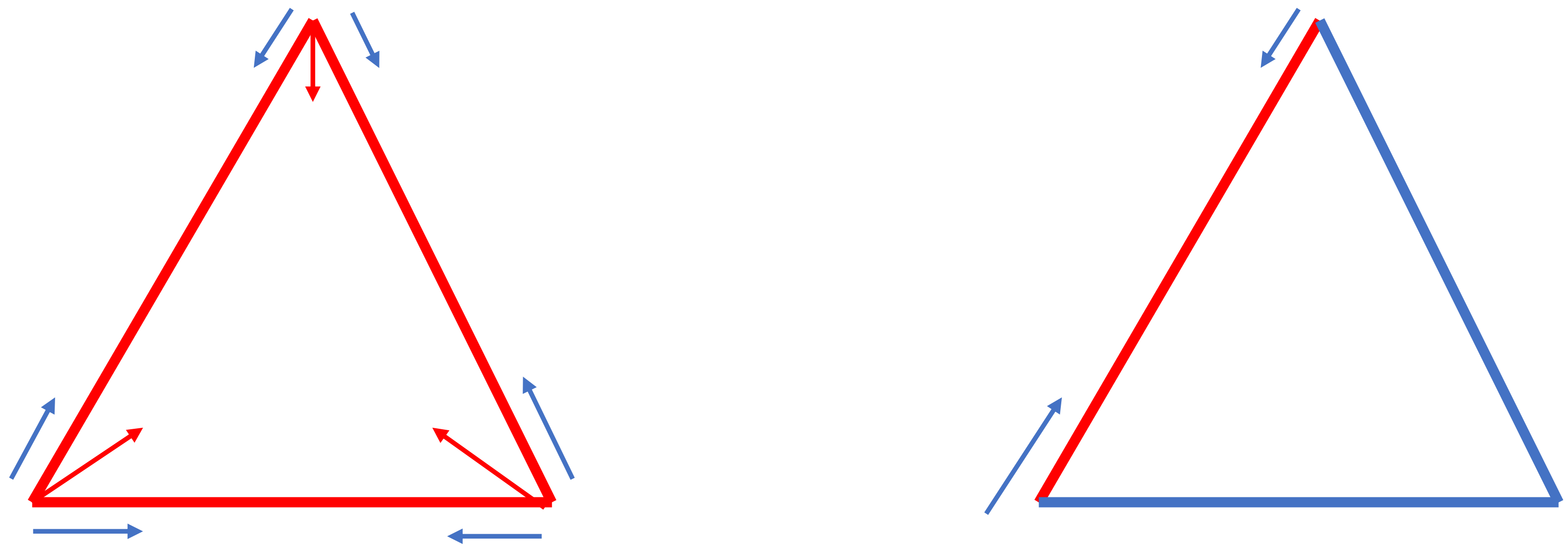
Conclusion: PBD

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

	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

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

	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

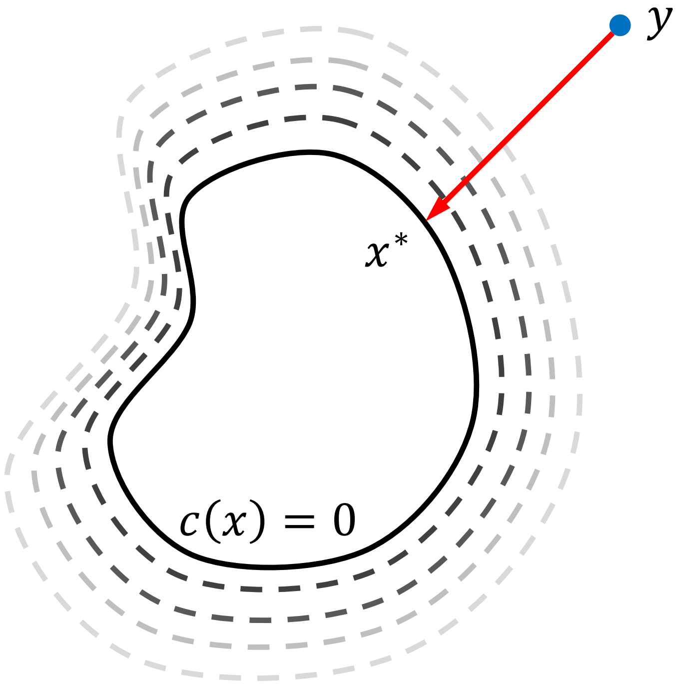
Conclusion: Geometric Stiffness

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

	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

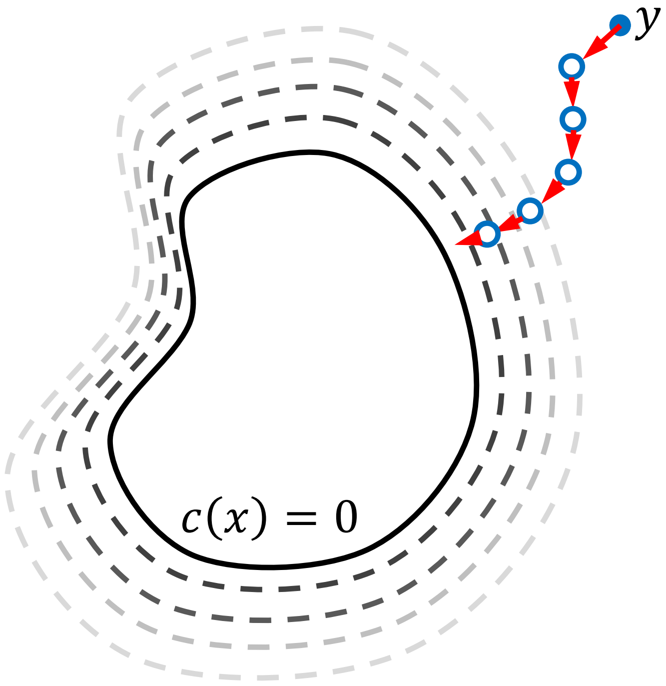
Key Takeaway from this Talk

- (X)PBD is approximately converging to…



Ground Truth

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



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