DiffPD: Differentiable Projective Dynamics

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, Wojciech Matusik

Presenter: Deepanshu
2019CS50427

ACM Reference format:
Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2021. DiffPD: Differentiable Projective Dynamics. ACM Trans. Graph. 41, 2, Article 13 (November 2021), 21 pages.

Differentiable simulators and soft body dynamics

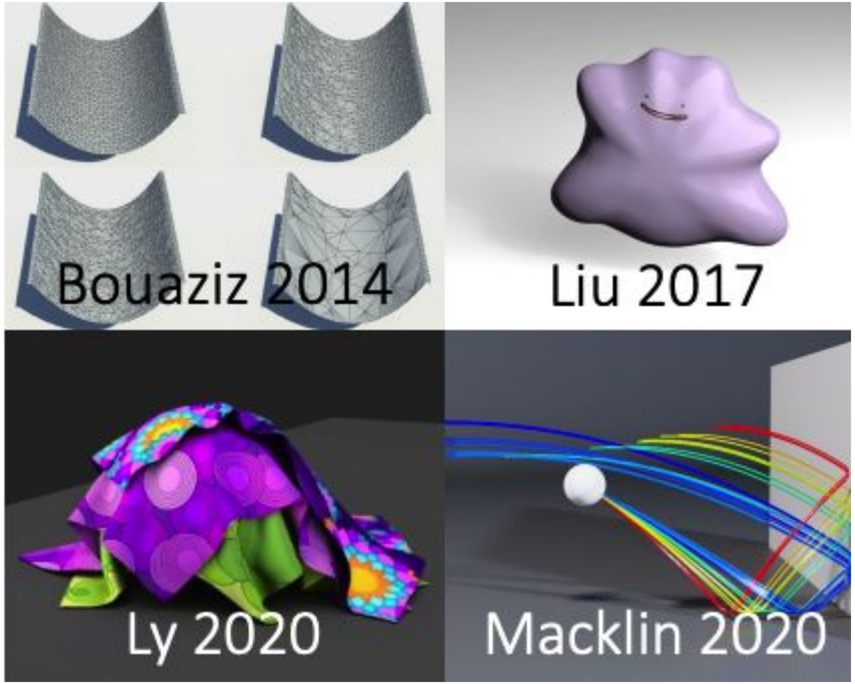
- Gradient knowledge helps in physics systems

- Motion of soft bodies: Not differentiable

- DoFs, friction, mass distribution

Related Work

Soft body dynamics



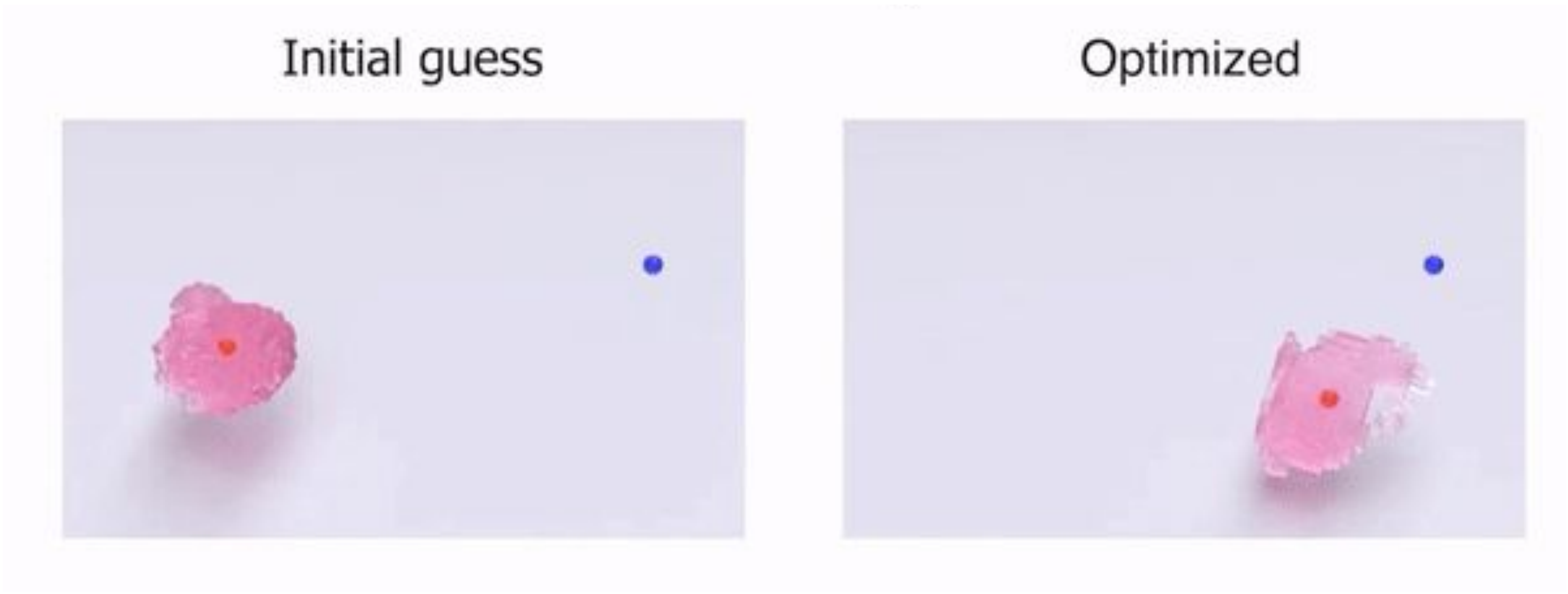
Differentiable physics

System Identification

Goal: estimate the material parameters from the motion

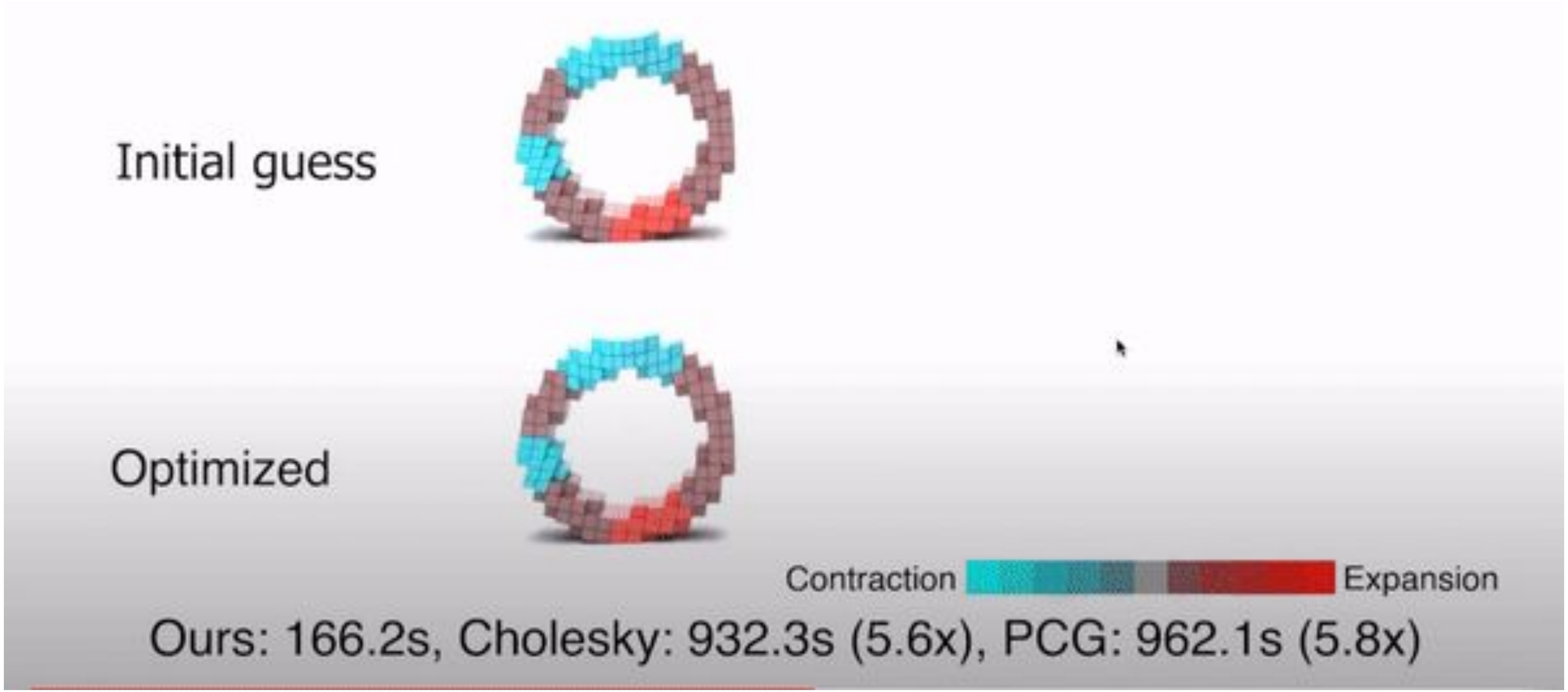
Initial State Optimisation

Goal: optimise the initial state and velocity to reach a final position



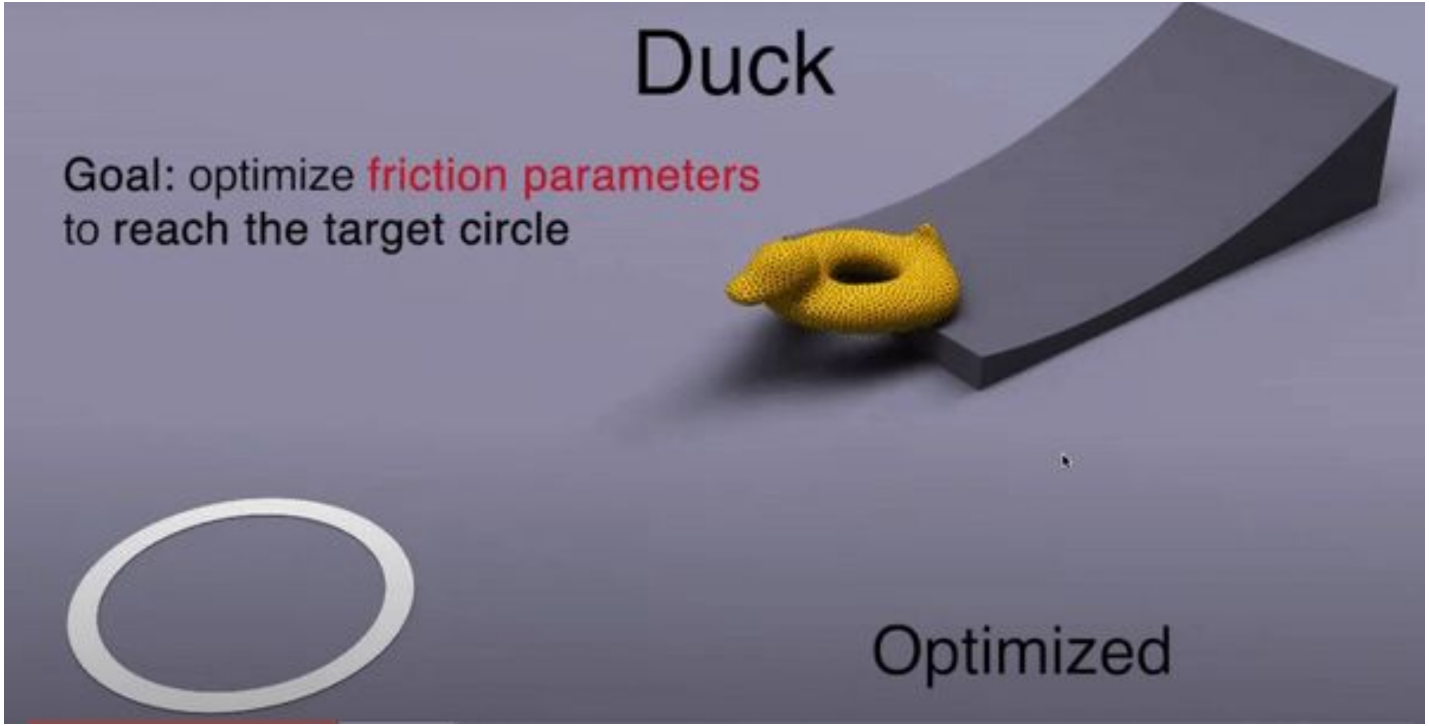
Trajectory optimisation

Goal: optimizing time-invariant actuation to get the desired trajectory



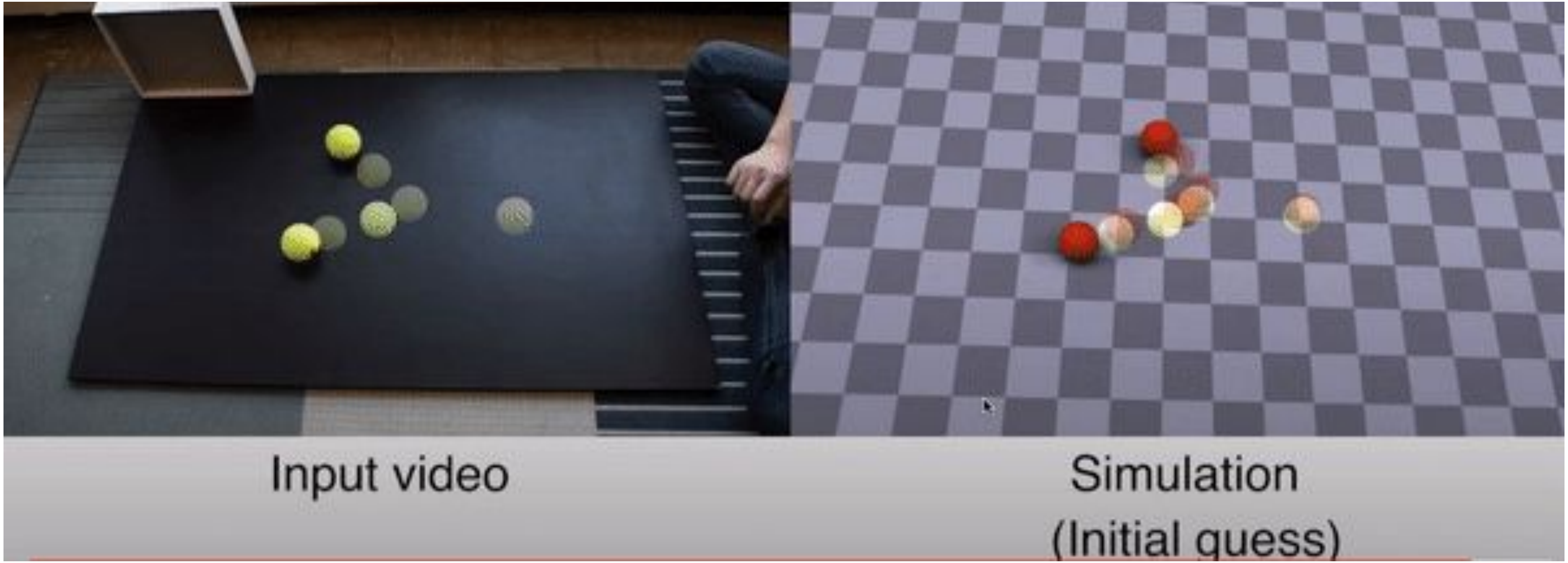
Motion planning

Goal: estimate the motion parameters to reach a given destination



Real-to-Sim

Goal: duplicate an actual scene in a simulator



Paper Contributions

- A fast PD-based differentiable soft-body simulator
- a differentiable collision handling algorithm
- demonstrations of the efficacy of our method on a wide range of applications
- 8x - 10x times faster than the benchmarks

Background: Implicit Time Integration

$$x_{i+1} = x_i + hv_{i+1}$$

$$v_{i+1} = v_i + hM^{-1}[-\nabla E(x_{i+1}) + f_{ext}]$$

Recast it as a saddle-point problem: find $\nabla g(\mathbf{x}_{i+1}) = \mathbf{0}$ where

$$g(\mathbf{x}) := \frac{1}{2h^2}(\mathbf{x} - \mathbf{y})^\top \mathbf{M}(\mathbf{x} - \mathbf{y}) + E(\mathbf{x})$$

$\mathbf{y} := \mathbf{x}_i + h\mathbf{v}_i + h^2\mathbf{M}^{-1}\mathbf{f}_{ext}$ is independent of $\mathbf{x}$.

Source: Paper slides

Stuart and Humphries [1996] and Martin et al. [2011]

Background: Implicit Time Integration

Newton's method: $\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k$ where

$$\nabla^2 g(\mathbf{x}^k)\Delta\mathbf{x}^k = \nabla g(\mathbf{x}^k)$$

Bottleneck: solving the matrix $\nabla^2 g(\mathbf{x}^k)$:

$$\nabla^2 g(\mathbf{x}^k) = \frac{1}{h^2}\mathbf{M} + \nabla^2 E(\mathbf{x}^k)$$

requires recomputation whenever $\mathbf{x}^k$ changes!

Source: Paper slides

Time Integration

Forward simulation: $\nabla^2 g(\mathbf{x}^k)\Delta\mathbf{x}^k = \nabla g(\mathbf{x}^k).$

Backpropagation: $\nabla^2 g(\mathbf{x}_{i+1})\mathbf{z} = \left(\frac{\partial L}{\partial \mathbf{x}_{i+1}}\right)^{\top}.$

Source: Paper slides

Numerical techniques in forward simulation and backpropagation are two sides of the same coin.

Efficient forward simulation solvers can be transferred to efficient backpropagation solvers!

Proposed Approach

- Consider a special case when Energy E consists of quadratic terms and is dependent on local features eg: Deformation gradient

$$g(\mathbf{x}) := \frac{1}{2h^2}(\mathbf{x}-\mathbf{y})^\top \mathbf{M}(\mathbf{x}-\mathbf{y}) + E(\mathbf{x})$$

The saddle-point problem $\nabla g = \mathbf{0}$ is now modified accordingly:

$$\min_{\mathbf{x},\{\mathbf{p}_c \in \mathcal{M}_c\}} \frac{1}{2h^2}(\mathbf{x}-\mathbf{y})^\top \mathbf{M}(\mathbf{x}-\mathbf{y}) + \sum_c ||\mathbf{G}_c \mathbf{x} - \mathbf{p}_c||_2^2$$

Background: Implicit Time Integration

Newton's method: $\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k$ where

$$\nabla^2 g(\mathbf{x}^k)\Delta\mathbf{x}^k = \nabla g(\mathbf{x}^k)$$

Bottleneck: solving the matrix $\nabla^2 g(\mathbf{x}^k)$:

$$\nabla^2 g(\mathbf{x}^k) = \frac{1}{h^2}\mathbf{M} + \nabla^2 E(\mathbf{x}^k)$$

requires recomputation whenever $\mathbf{x}^k$ changes!

Source: Paper slides

Proposed Approach

- Consider a special case when Energy E consists of quadratic terms and is dependent on local features eg: Deformation gradient

The saddle-point problem $\nabla g = \mathbf{0}$ is now modified accordingly:

$$\min_{\mathbf{x}, \{\mathbf{p}_c \in \mathcal{M}_c\}} \frac{1}{2h^2} (\mathbf{x} - \mathbf{y})^\top \mathbf{M} (\mathbf{x} - \mathbf{y}) + \sum_c ||\mathbf{G}_c \mathbf{x} - \mathbf{p}_c||_2^2$$

Proposed Approach

With PD, $\nabla^2 g$ becomes

$$\nabla^2 g(\mathbf{x}) = \frac{1}{h^2}\mathbf{M} + \sum_c \mathbf{G}_c^\top \mathbf{G}_c - \sum_c \mathbf{G}_c^\top \frac{\partial \mathbf{p}_c}{\partial \mathbf{x}}$$

$$:= \mathbf{A} - \Delta\mathbf{A}$$

Source: Paper slides

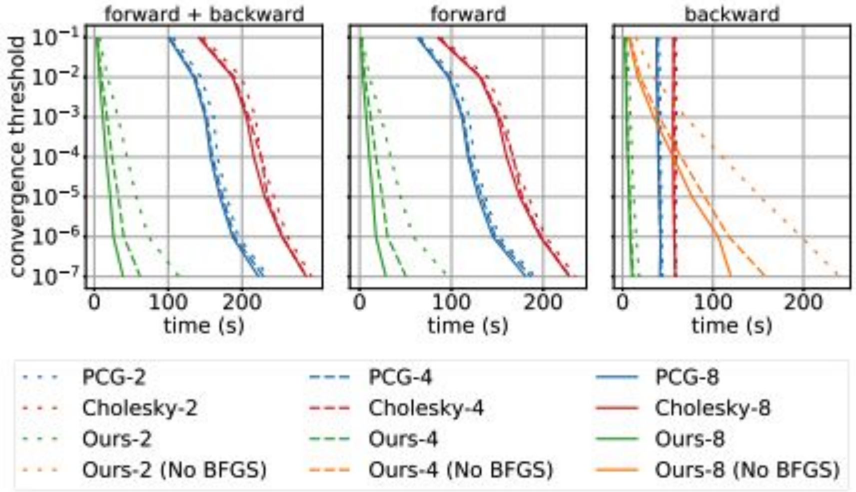
Summary

Efficient forward simulation: $\nabla^2 g(\mathbf{x}^k)\Delta\mathbf{x}^k = \nabla g(\mathbf{x}^k)$.

Efficient backpropagation: $\nabla^2 g(\mathbf{x}_{i+1})\mathbf{z} = \left(\frac{\partial L}{\partial \mathbf{x}_{i+1}}\right)^{\top}$.

Source: Paper slides

Performance



Sec.	Task name	Newton-PCG				Newton-Cholesky				DiffPD (Ours)				
		Fwd.	Back.	Eval.	Loss	Fwd.	Back.	Eval.	Loss	Fwd.	Back.	Eval.	Loss	Speedup
6.2	Cantilever	118.2	39.4	-	-	160.1	55.9	-	-	10.5	5.5	-	-	10×
	Rolling sphere	107.3	31.3	-	-	135.6	36.6	-	-	14.0	5.7	-	-	8×
7.1	Plant	1,089.5	530.5	10	1.9e-3	929.6	525.2	10	1.9e-3	71.6	94.7	28	**5.9e-7**	9×
	Bouncing ball	269.3	90.9	43	**7.9e-2**	262.6	102.5	22	8.4e-2	15.8	14.2	12	9.6e-2	12×
7.2	Bunny	277.7	88.0	21	7.0e-3	358.2	126.9	29	**5.1e-3**	24.0	17.3	11	2.3e-2	9×
	Routing tendon	108.2	56.7	36	6.0e-4	107.3	58.7	38	**4.9e-4**	8.3	9.9	30	9.6e-4	9×
7.3	Torus	751.9	210.3	47	−2.3e-3	719.9	212.4	43	**−2.4e-2**	84.3	81.9	27	0	6×
	Quadruped	289.2	51.5	69	**−1.8e0**	246.3	47.8	54	-1.1e0	50.2	15.8	30	0	4×
	Cow	771.7	141.7	14	9.7e-1	620.1	140.2	20	9.8e-1	105.3	43.7	31	**0**	5×
7.4	Starfish	217.7	105.1	100	4.8e-1	244.0	129.4	100	1.4e-1	5.7	10.8	100	**0**	19×
	Shark	260.7	159.3	100	9.8e-1	599.4	241.8	100	**−9.0e-3**	35.5	15.3	100	0	8×
7.5	Tennis balls	54.6	6.4	14	7.2e-2	26.8	5.8	12	7.2e-2	24.1	15.9	41	**6.9e-2**	0.8×

Penalty Based Contact

- Previous PD simulations use penalty-based soft contact models.
- Contact forces are represented with fictitious energy Ec and matrix Gc.
- Ec pushes nodes back upon contact surface penetration.
- Handling friction with penalty-based forces in PD.

Background: Implicit Time Integration

$$x_{i+1} = x_i + hv_{i+1}$$

$$v_{i+1} = v_i + hM^{-1}[-\nabla E(x_{i+1}) + f_{ext}]$$

Recast it as a saddle-point problem: find $\nabla g(\mathbf{x}_{i+1}) = \mathbf{0}$ where

$$g(\mathbf{x}) := \frac{1}{2h^2}(\mathbf{x} - \mathbf{y})^\top \mathbf{M}(\mathbf{x} - \mathbf{y}) + E(\mathbf{x})$$

$\mathbf{y} := \mathbf{x}_i + h\mathbf{v}_i + h^2\mathbf{M}^{-1}\mathbf{f}_{ext}$ is independent of $\mathbf{x}$.

Source: Paper slides

Limitations

- Energy model assumption restricts material diversity.
- Contact models prioritize differentiability over realism.
- Scalability limited to thousands of elements.
- Slower for locomotion tasks due to contact inclusion.
- Optimization methods may struggle with non-convex landscapes.

Thank you