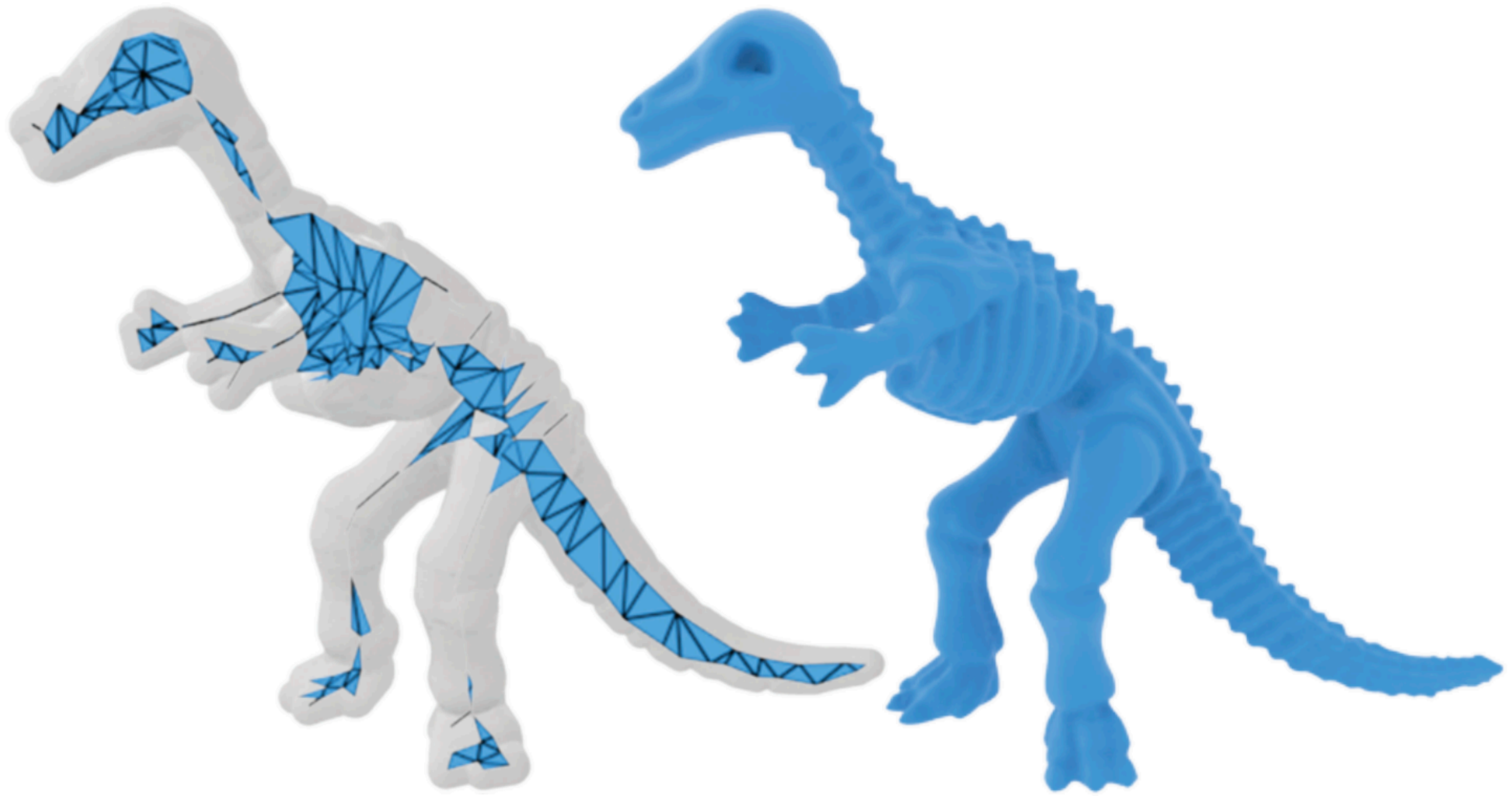
Instructor: Minchen Li



Lec 12: Reduced Order Models

15-763: Physics-Based Animation of Solids and Fluids (S25)

Recap: Frictional Self-Contact

Idea: Approximating Contact Forces as Conservative Forces

$$
\begin{aligned}
&\int_{\partial\Omega^0} Q_i(\mathbf{X},t)T_i(\mathbf{X},t)ds(\mathbf{X}) \\
&= \int_{\Gamma_D} Q_i(\mathbf{X},t)T_{D|i}(\mathbf{X},t)ds(\mathbf{X}) + \int_{\Gamma_N} Q_i(\mathbf{X},t)T_{N|i}(\mathbf{X},t)ds(\mathbf{X}) \\
&+ \int_{\Gamma_C} Q_i(\mathbf{X},t)T_{C|i}(\mathbf{X},t)ds(\mathbf{X}) + \int_{\Gamma_C} Q_i(\mathbf{X},t)T_{F|i}(\mathbf{X},t)ds(\mathbf{X}).
\end{aligned}
$$

(Here Γ_C can overlap with Γ_D or Γ_N)

Recap: Normal Self-Contact

Barrier Potential

$$\mathbf{T}_C(\mathbf{X}, t) = -\frac{\partial b(\min_{\mathbf{X}_2 \in \Gamma_C - \mathcal{N}(\mathbf{X})} \|\mathbf{x}(\mathbf{X}, t) - \mathbf{x}(\mathbf{X}_2, t)\|, \hat{d})}{\partial \mathbf{x}(\mathbf{X}, t)}$$

where $\mathcal{N}(\mathbf{X}) = \{\mathbf{X}_N \in \mathbb{R}^d \mid \|\mathbf{X}_N - \mathbf{X}\| < r\}$ is an infinitesimal circle around $\mathbf{X}$ with the radius r sufficiently small to avoid unnecessary contact forces between a point and its geodesic neighbors.

Need $\hat{d} \to 0$, $r \to 0$, and $\hat{d}/r \to 0$.

Barrier Potential:

$$\int_{\Gamma_C} \frac{1}{2} b(\min_{\mathbf{X}_2 \in \Gamma_C - \mathcal{N}(\mathbf{X})} \|\mathbf{x}(\mathbf{X}, t) - \mathbf{x}(\mathbf{X}_2, t)\|, \hat{d}) ds(\mathbf{X}) \implies$$

But $\min()$ is non-smooth!

$b()$ is monotonically decreasing,

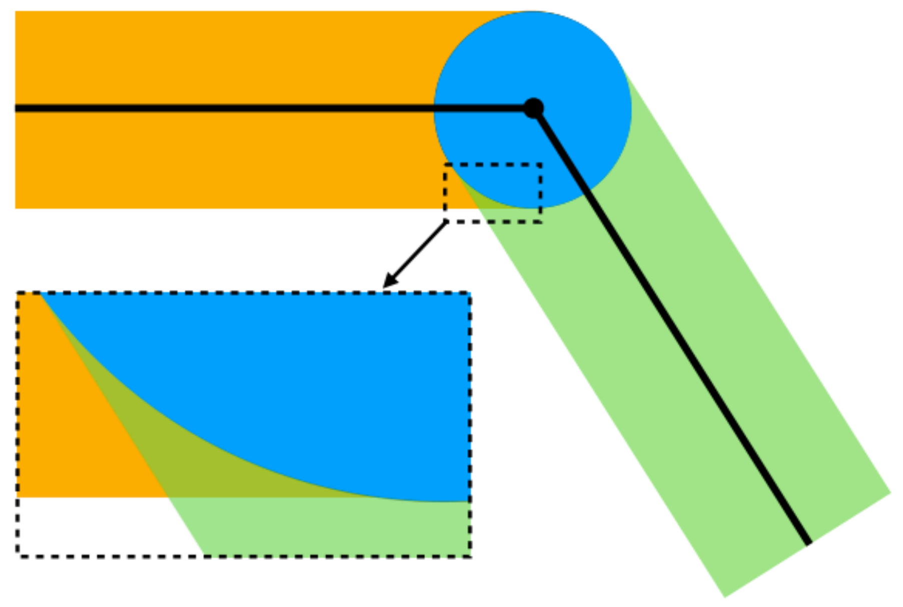
$$\int_{\Gamma_C} \frac{1}{2} \max_{e \in \mathcal{E} - I(\mathbf{X})} b(d^{\mathrm{PE}}(\mathbf{x}(\mathbf{X}, t), e), \hat{d}) ds(\mathbf{X})$$

$$\max(a_1, a_2, ..., a_n) \approx (a_1^p + a_2^p + ... + a_n^p)^{\frac{1}{p}}$$

Accurate when $p \to \infty$: Expensive!

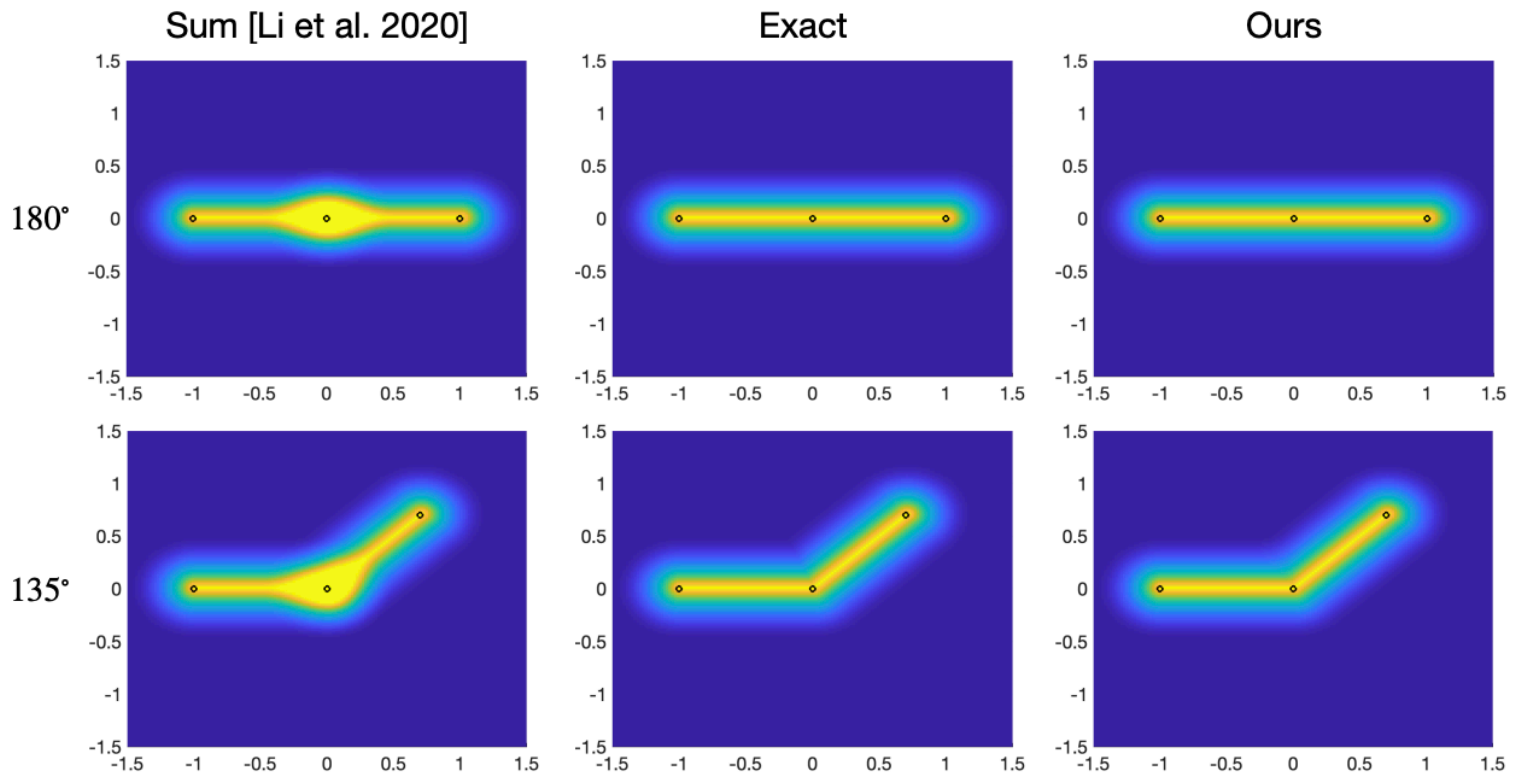
Recap: Normal Self-Contact

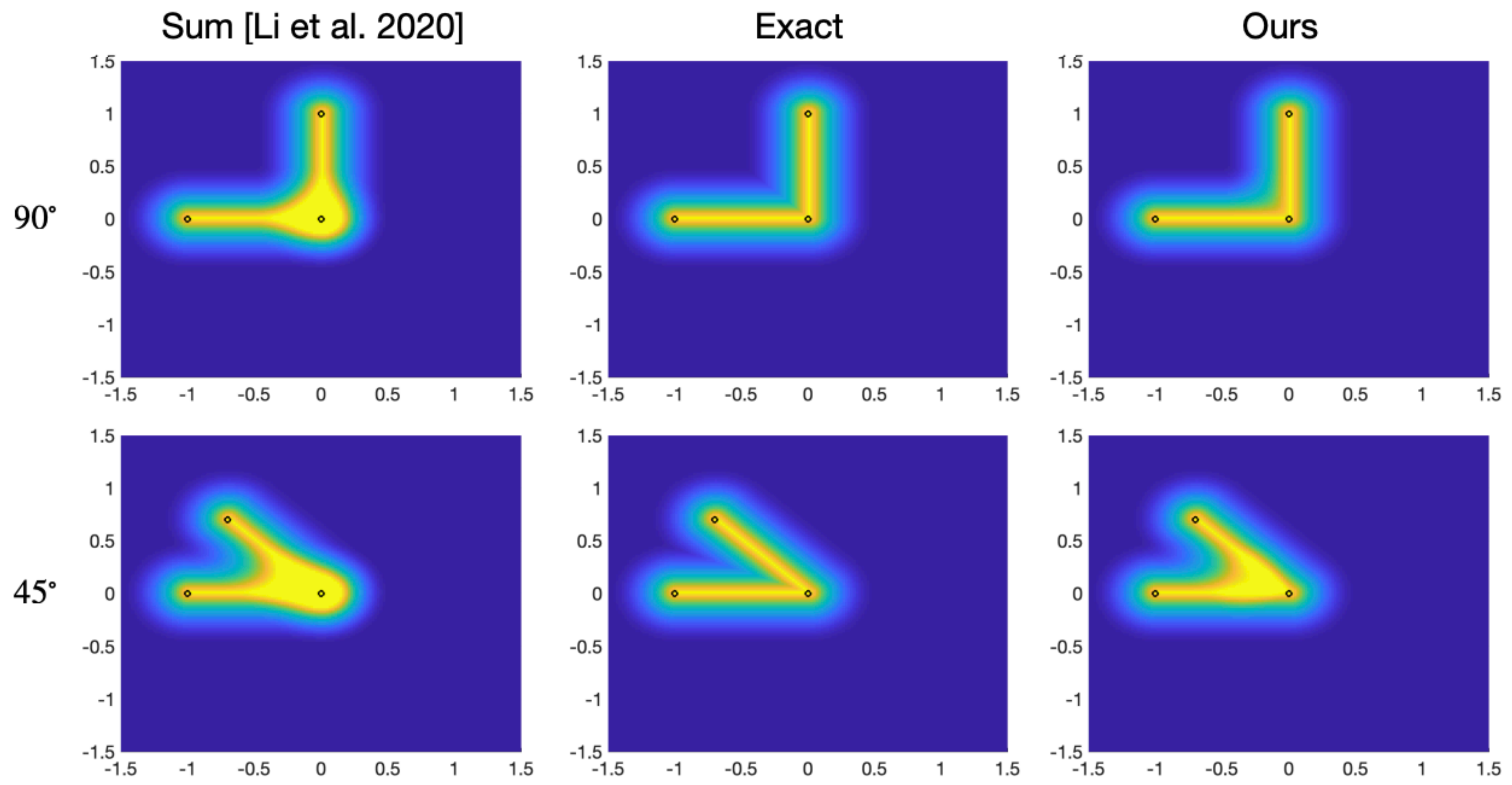
Smoothly Approximating the Barrier Potential



Can subtract the duplicate point-point barrier [Li et al. 2023]:

$$\Psi_c(x) = \sum_{e \in E \setminus x} b(d(x,e),\hat{d}) \quad - \sum_{x_2 \in V_{int} \setminus x} b(d(x,x_2),\hat{d}) \approx \max_{e \in E \setminus x} b(d(x,e),\hat{d})$$

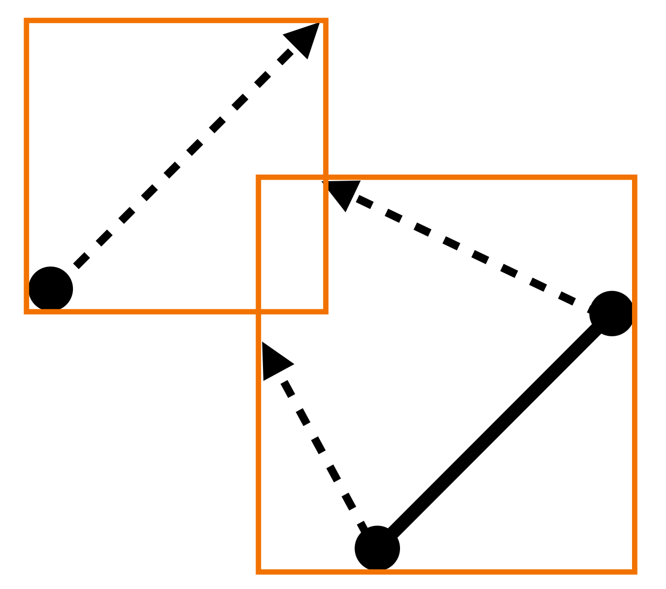


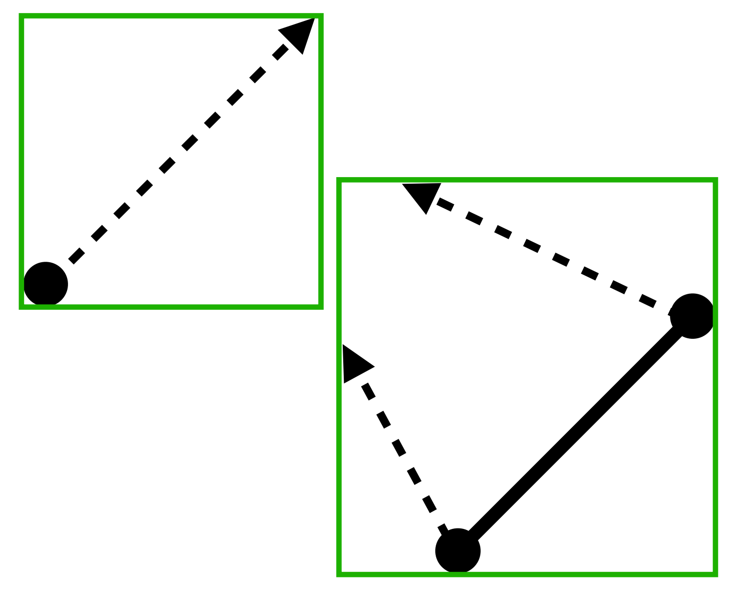


Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, Danny M. Kaufman. Convergent Incremental Potential Contact. Arxiv 2307.15908.

Recap: Broad Phase CCD

- Step 1: query proximal primitive pairs using spatial data structures:
  - Spatial Hash
  - Bounding Box Hierarchy (BVH)
  - …
- Step 2: Check bounding box overlap:


Case 1: needs narrow phase


Case 2: can skip

Recap: Narrow Phase CCD

Additive CCD [Li et al. 2021]

Taking a point-edge pair as an example, the key insight of ACCD is that, given the current positions $\mathbf{p}$, $\mathbf{e}_0$, $\mathbf{e}_1$ and search directions $\mathbf{d}_p$, $\mathbf{d}_{e0}$, $\mathbf{d}_{e1}$, its TOI can be calculated as

$$\alpha_{\mathrm{TOI}} = \frac{\|\mathbf{p} - ((1-\lambda)\mathbf{e}_0 + \lambda\mathbf{e}_1)\|}{\|\mathbf{d}_p - ((1-\lambda)\mathbf{d}_{e0} + \lambda\mathbf{d}_{e1})\|},$$

assuming $(1-\lambda)\mathbf{e}_0 + \lambda\mathbf{e}_1$ is the point on the edge that $\mathbf{p}$ will first collide with. The issue is that we do not a priori know λ. But we can derive a lower bound of α_{TOI} as

$$\begin{aligned}\alpha_{\mathrm{TOI}} &\geq \frac{\min_{\lambda\in[0,1]} \|\mathbf{p} - ((1-\lambda)\mathbf{e}_0 + \lambda\mathbf{e}_1)\|}{\|\mathbf{d}_p\| + \|(1-\lambda)\mathbf{d}_{e0} + \lambda\mathbf{d}_{e1}\|} \\ &\geq \frac{d^{\mathrm{PE}}(\mathbf{p}, \mathbf{e}_0, \mathbf{e}_1)}{\|\mathbf{d}_p\| + \max(\|\mathbf{d}_{e0}\|, \|\mathbf{d}_{e1}\|)} = \alpha_l\end{aligned}$$

```
$\bar{p} \leftarrow \sum_i p_i/4$
for $i$ in $\{0, 1, 2, 3\}$ do
    $p_i \leftarrow p_i - \bar{p}$
```

Algorithm:

```
Make a local copy of $x$
$\alpha \leftarrow 0$
While distance not close enough
    Calculate lower bound $\alpha_l$
    $x \leftarrow x + \alpha_l p$
    $\alpha \leftarrow \alpha + \alpha_l$
Return $\alpha$
```

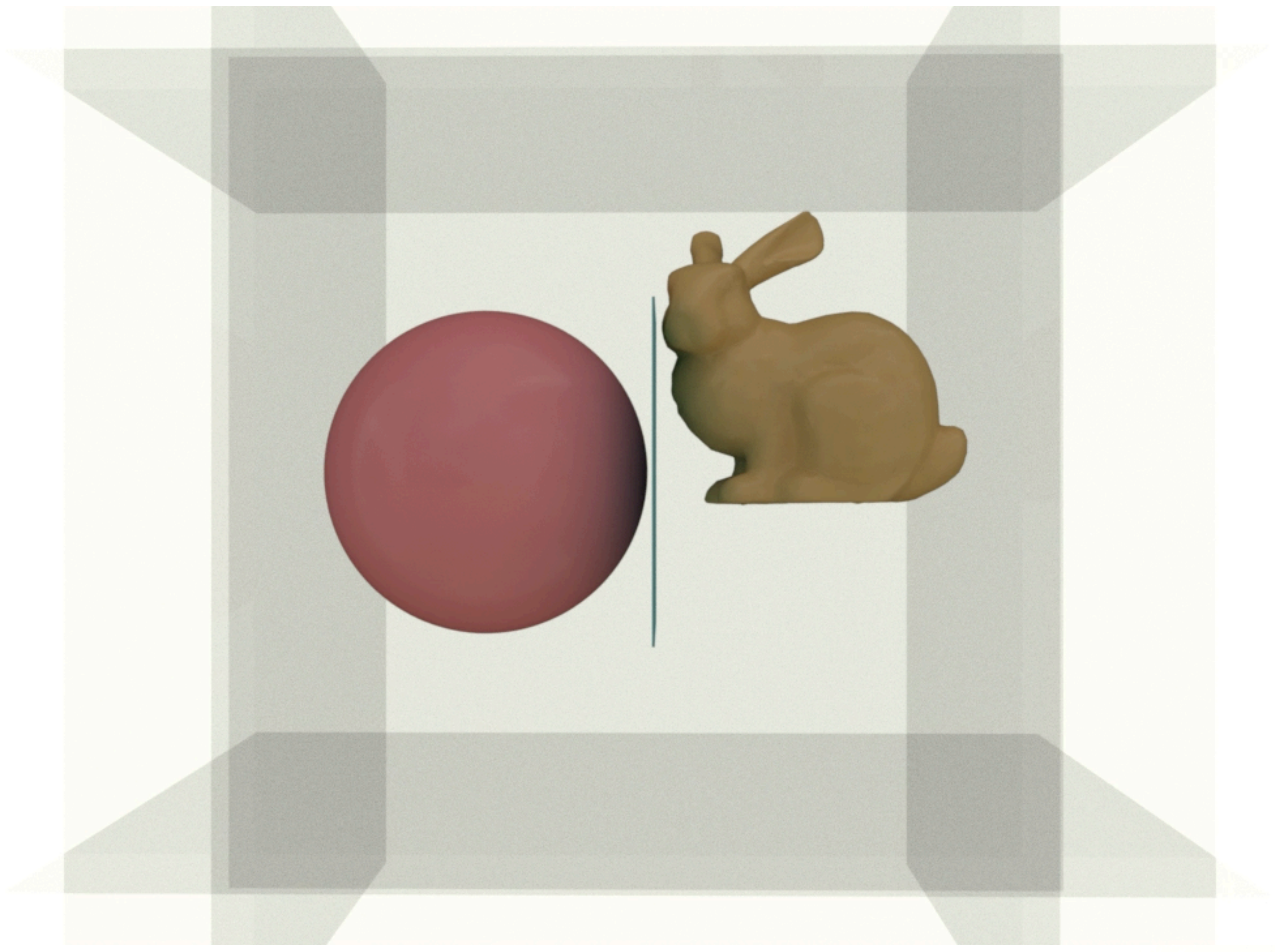
Only need to evaluate distances;

More robust than root-finding;

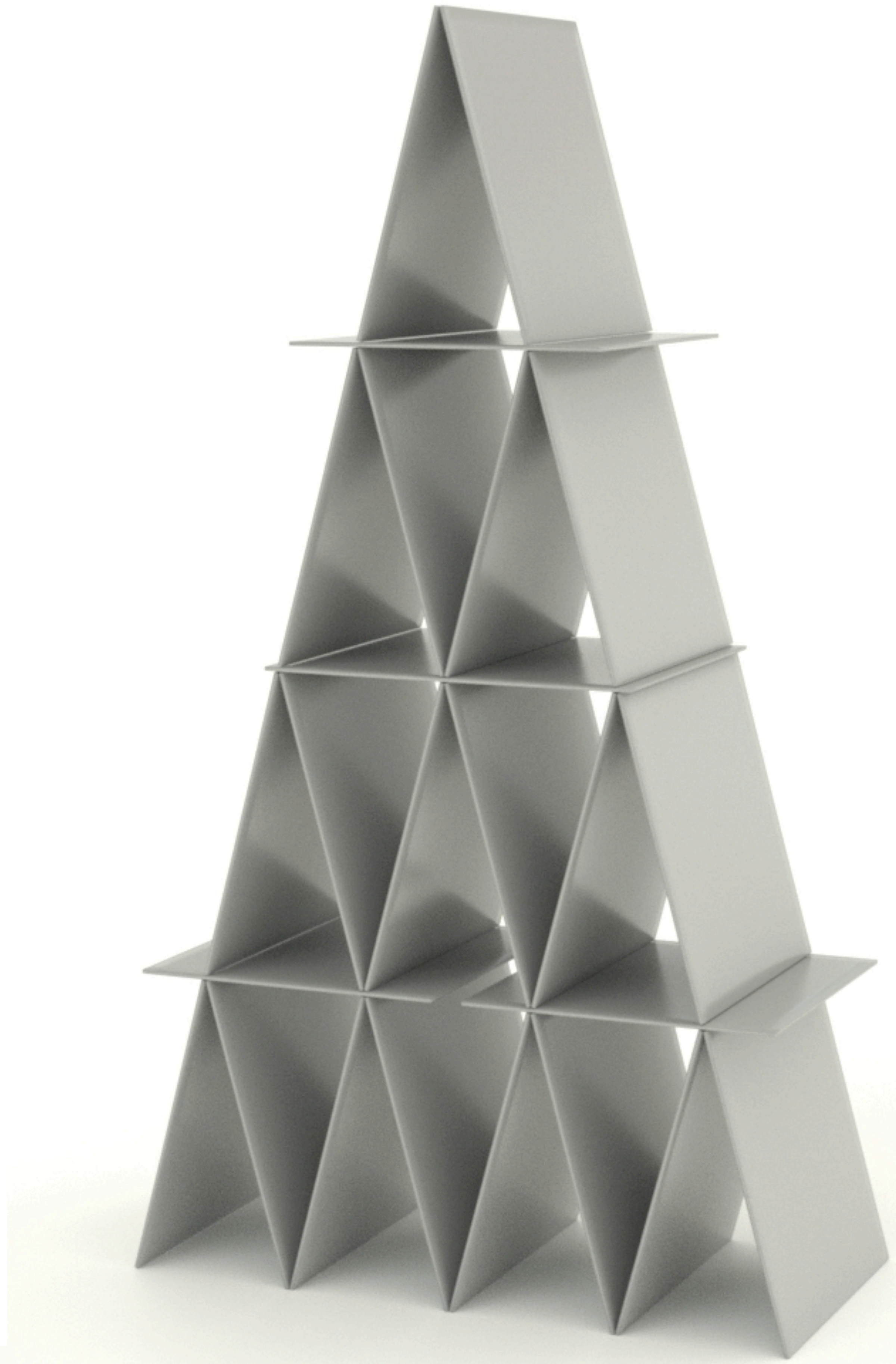
Generalize to higher-order primitives.

Results: Elastic Body Simulation

With Guarantees of Nonpenetration, Non-inversion, and Convergence



E = ~10^5 Pa



E = ~10^9 Pa

Today:

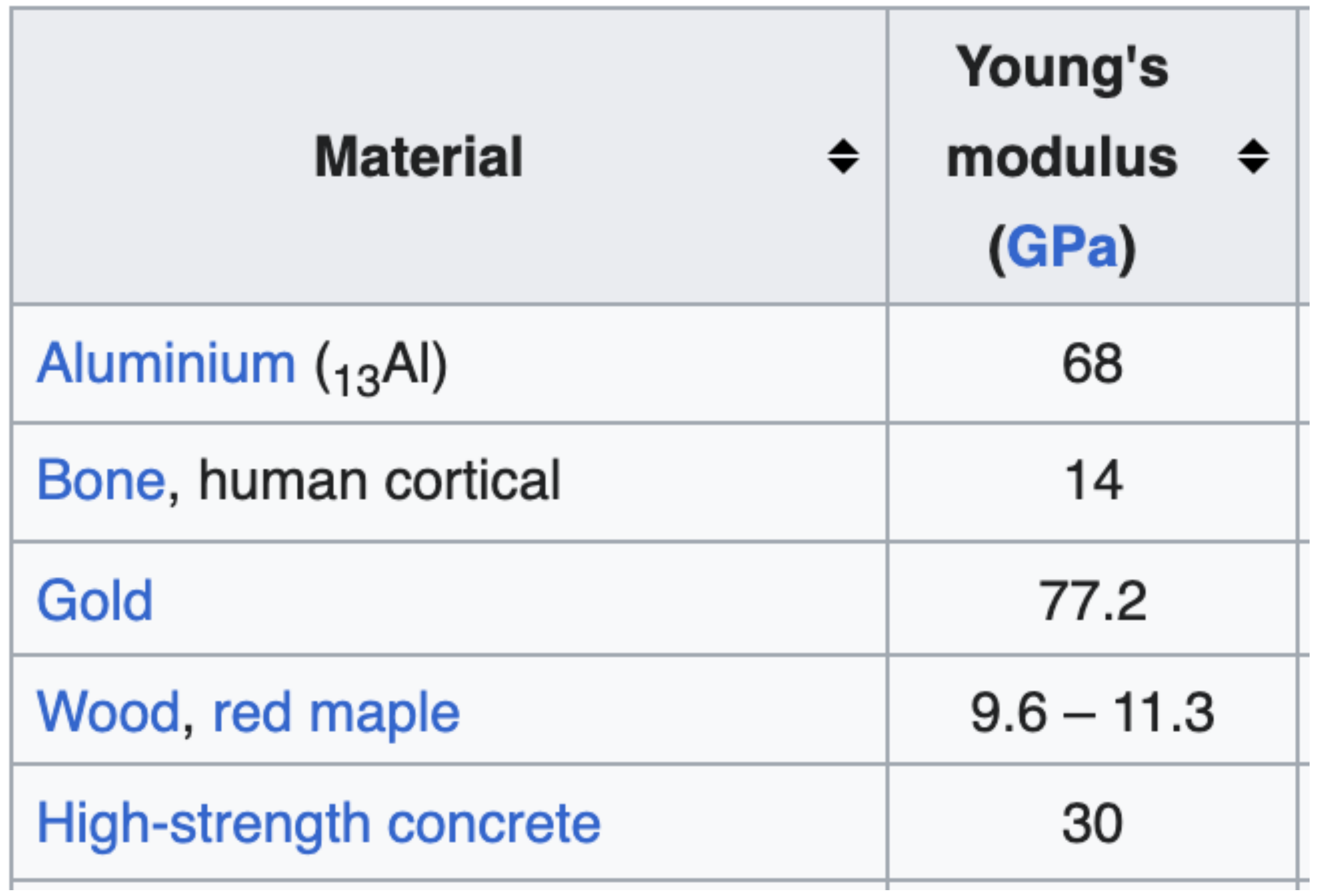
- Simulating Stiff Elastic Solids
- Modal-Order Reduction
- Implementation & Demo

Today:

- Simulating Stiff Elastic Solids
- Modal-Order Reduction
- Implementation & Demo

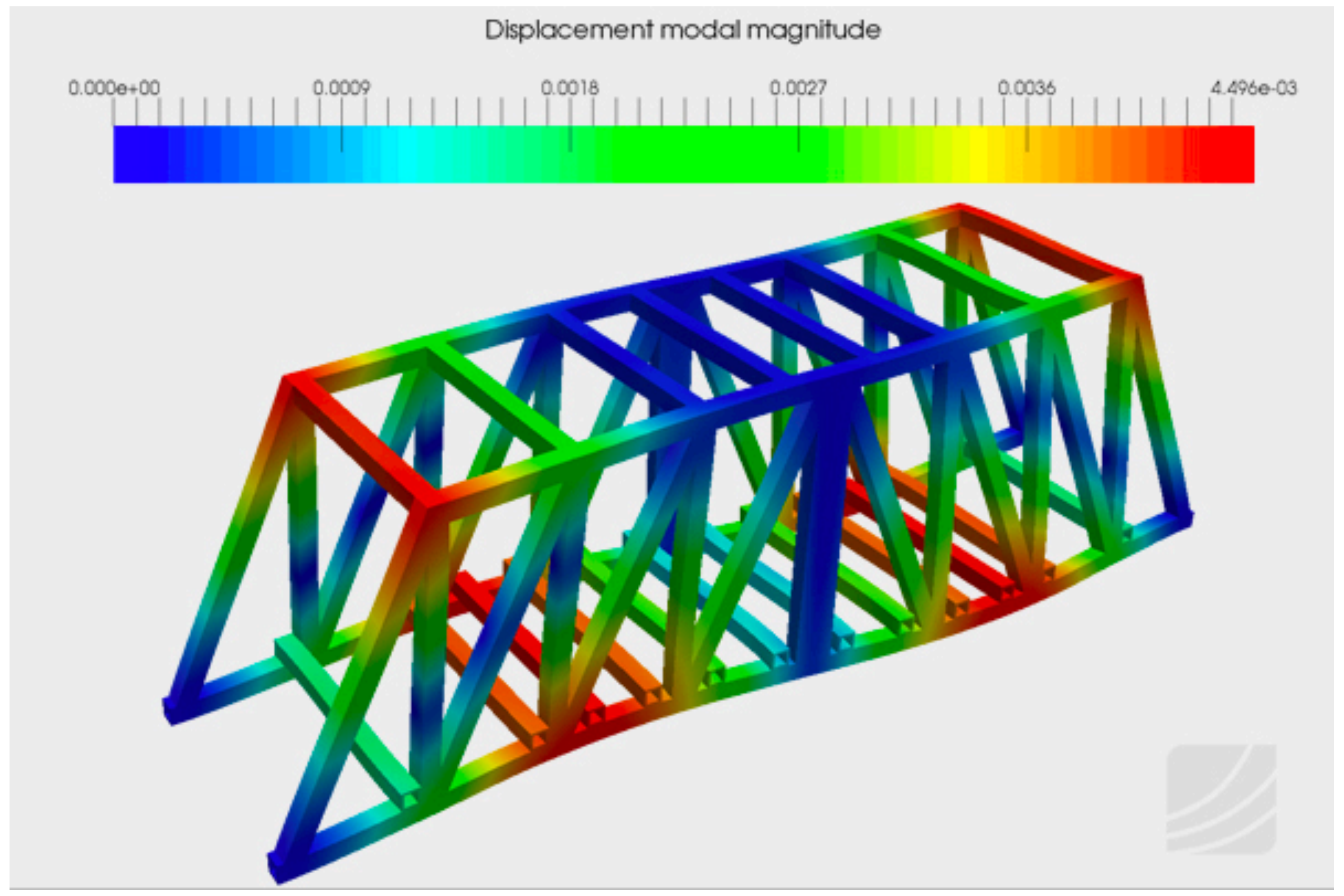
Simulating Super Stiff Materials

Finite Element Method (FEM)



Material	Young's modulus (GPa)
Aluminium ($_{13}$Al)	68
Bone, human cortical	14
Gold	77.2
Wood, red maple	9.6 – 11.3
High-strength concrete	30

- Applications:



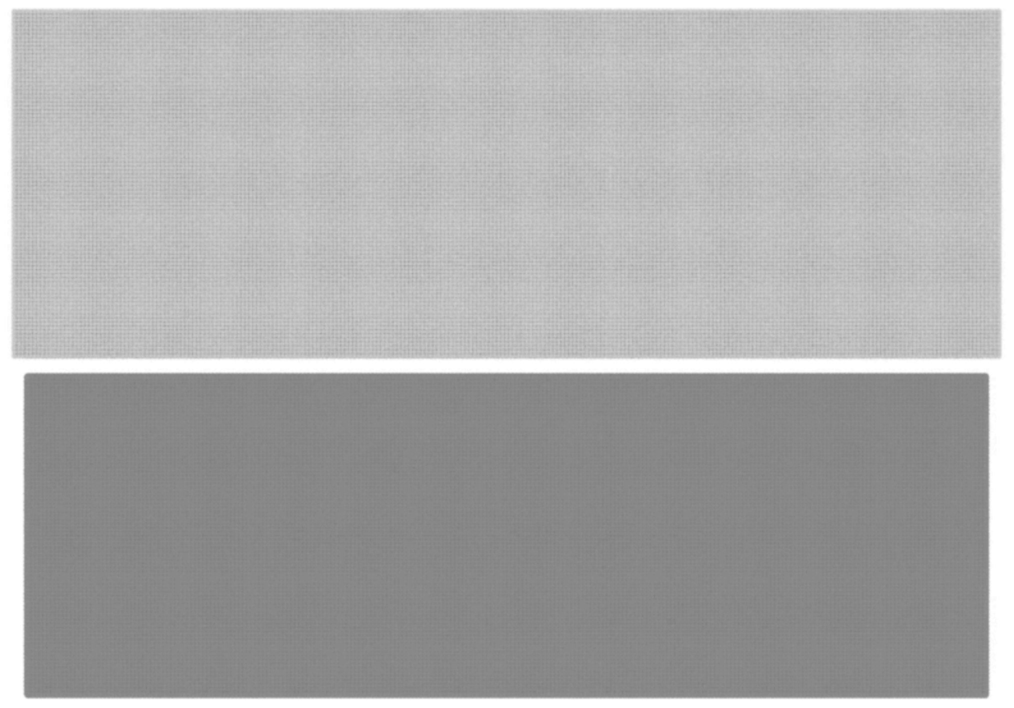
Structural Analysis

- Usually no visible deformation before fracture
- Can compute stress distribution using FEM

$$\min_{\textbf{structure}} \Psi$$

$$\textbf{s.t.} -\nabla_x \Psi + f^{\textbf{ext}} = 0$$

$$\textbf{volume} < \textbf{target}$$



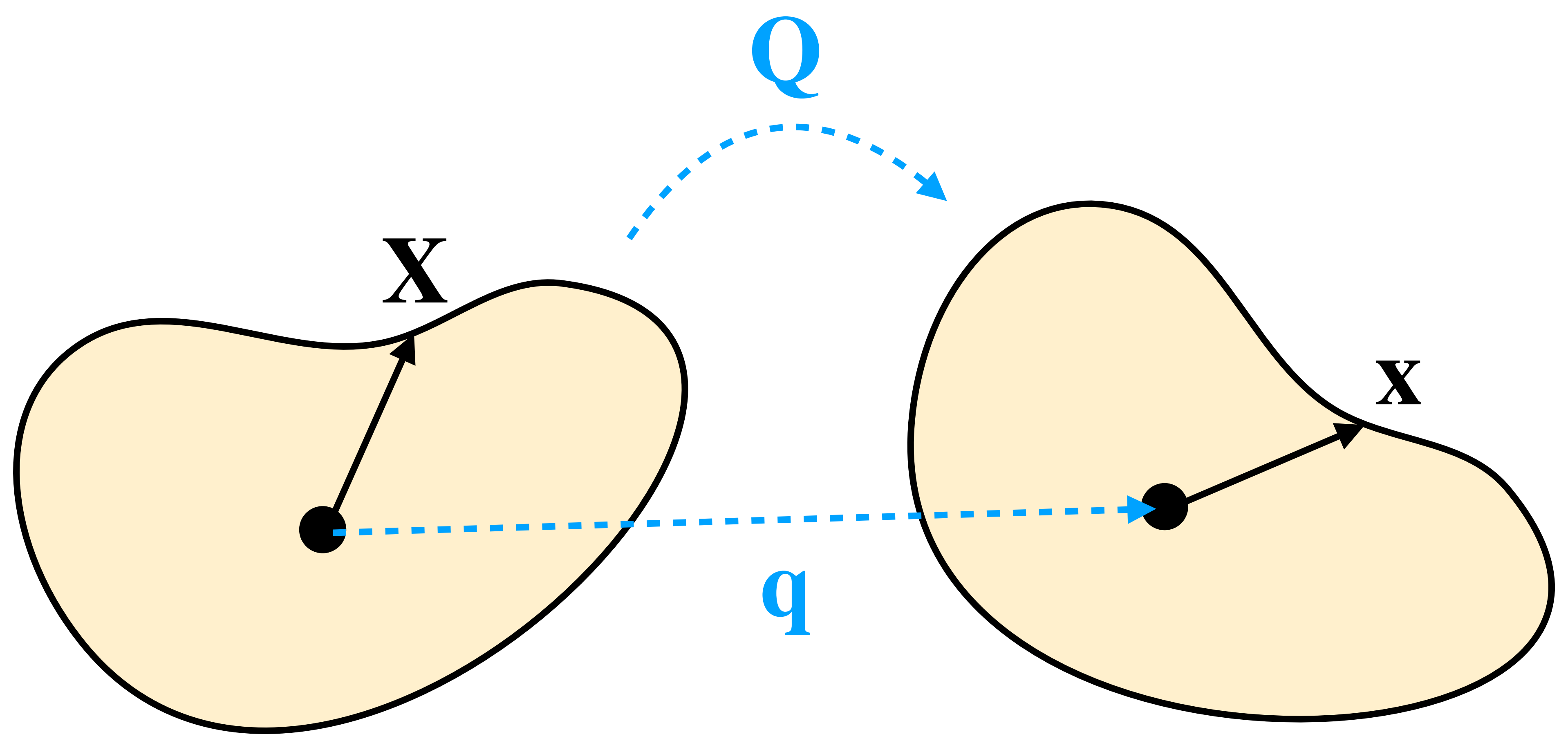
Topology Optimization

Simulating Super Stiff Materials

Rigid Body Representation

If only care about the motions,

Can simply track rotation $\mathbf{Q}$ and translation $\mathbf{q}$ per body!



$$\mathbf{x} = \mathbf{Q}\mathbf{X} + \mathbf{q}$$

Constant deformation gradient per body,

No volumetric discretization needed!

Simulating Super Stiff Materials

Rigid Body Dynamics: Derivation

Full order dynamics:

$$\min_x \frac{1}{2}\|x - \tilde{x}^n\|_M^2 + h^2 \sum P(x)$$

Reduced order DOF:

$$\mathbf{x} = \mathbf{Q}\mathbf{X} + \mathbf{q} \in \mathbb{R}^3 \iff \begin{array}{l} x = \bar{X}Q + \bar{S}q \in \mathbb{R}^{3l} \\ Q \in \mathbb{R}^{9m},\ \ \bar{X} \in \mathbb{R}^{3l\times 9m} \\ q \in \mathbb{R}^{3m},\ \ \bar{S} \in \mathbb{R}^{3l\times 3m} \end{array}$$

l: number of nodes,
m: number of bodies

Reduced order dynamics (from subspace optimization):

$$\min_{Q,q} \frac{1}{2}\|\bar{X}Q + \bar{S}q - \tilde{x}^n\|_M^2 + h^2 \sum P(\bar{X}Q + \bar{S}q) \quad \textbf{s.t.} \quad \mathbf{Q}^T\mathbf{Q} = \mathbf{I} \ \ \forall \mathbf{Q} \ \ (\textbf{or } f(Q) = 0)$$

$$\Longrightarrow$$

$$\bar{X}^T M(\bar{X}Q + \bar{S}q - \tilde{x}^n) + h^2 \sum \bar{X}^T \nabla P(\bar{X}Q + \bar{S}q) + (\nabla f(Q))^T \lambda = 0$$

$$\bar{S}^T M(\bar{X}Q + \bar{S}q - \tilde{x}^n) + h^2 \sum \bar{S}^T \nabla P(\bar{X}Q + \bar{S}q) = 0$$

$$f(Q) = 0$$

Alternative derivations:

- **Lagrangian Mechanics;**
- **Linear and Angular Momentum Conservations;**
- **…**

Simulating Super Stiff Materials

Rigid Body Dynamics: Mass Matrix and Inertia Tensor

Reduced order dynamics (from subspace optimization):

$$\bar{X}^T M(\bar{X}Q + \bar{S}q - \tilde{x}^n) + h^2 \sum \bar{X}^T \nabla P(\bar{X}Q + \bar{S}q) + (\nabla f(Q))^T \lambda = 0$$

$$\bar{S}^T M(\bar{X}Q + \bar{S}q - \tilde{x}^n) + h^2 \sum \bar{S}^T \nabla P(\bar{X}Q + \bar{S}q) = 0$$

$$f(Q) = 0$$

- $\bar{X}^T M \bar{X}$ **is the mass matrix of** Q **related to inertia tensor**

Calculating $\bar{X}^T M \bar{X}$ **without volumetric discretization:**

1. **Convert to continuous form** $\int_{\Omega^0} \rho \mathbf{X}\mathbf{X}^T d\mathbf{X}$
2. **Transform to surface integral using Divergence Theorem**
3. **Discretize the surface integral**

Simulating Super Stiff Materials

Rigid Body Dynamics: Change of Variable

Reduced order dynamics (from subspace optimization):

$$\min_{Q,q} \frac{1}{2}\|\bar{X}Q + \bar{S}q - \tilde{x}^n\|_M^2 + h^2 \sum P(\bar{X}Q + \bar{S}q) \quad \textbf{s.t.} \quad \mathbf{Q}^T\mathbf{Q} = \mathbf{I} \ \ \forall \mathbf{Q} \ \ (\text{or } f(Q) = 0)$$

Use rotation vector θ:

$$\min_{\theta,q} \frac{1}{2}\|\bar{X}R(\theta) + \bar{S}q - \tilde{x}^n\|_M^2 + h^2 \sum P(\bar{X}R(\theta) + \bar{S}q)$$

Unconstrained!

6 DOF per body!

Rodrigues' Rotation Formula:

$$\mathcal{R}(\boldsymbol{\theta}) = \mathrm{Id} + \sin(\|\boldsymbol{\theta}\|)\left[\frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}\right] + (1 - \cos(\|\boldsymbol{\theta}\|))\left[\frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}\right]^2$$

Highly nonlinear!

Simulating Super Stiff Materials

Rigid Body Dynamics: Frictional Contact via IPC [Li et al. 2020]

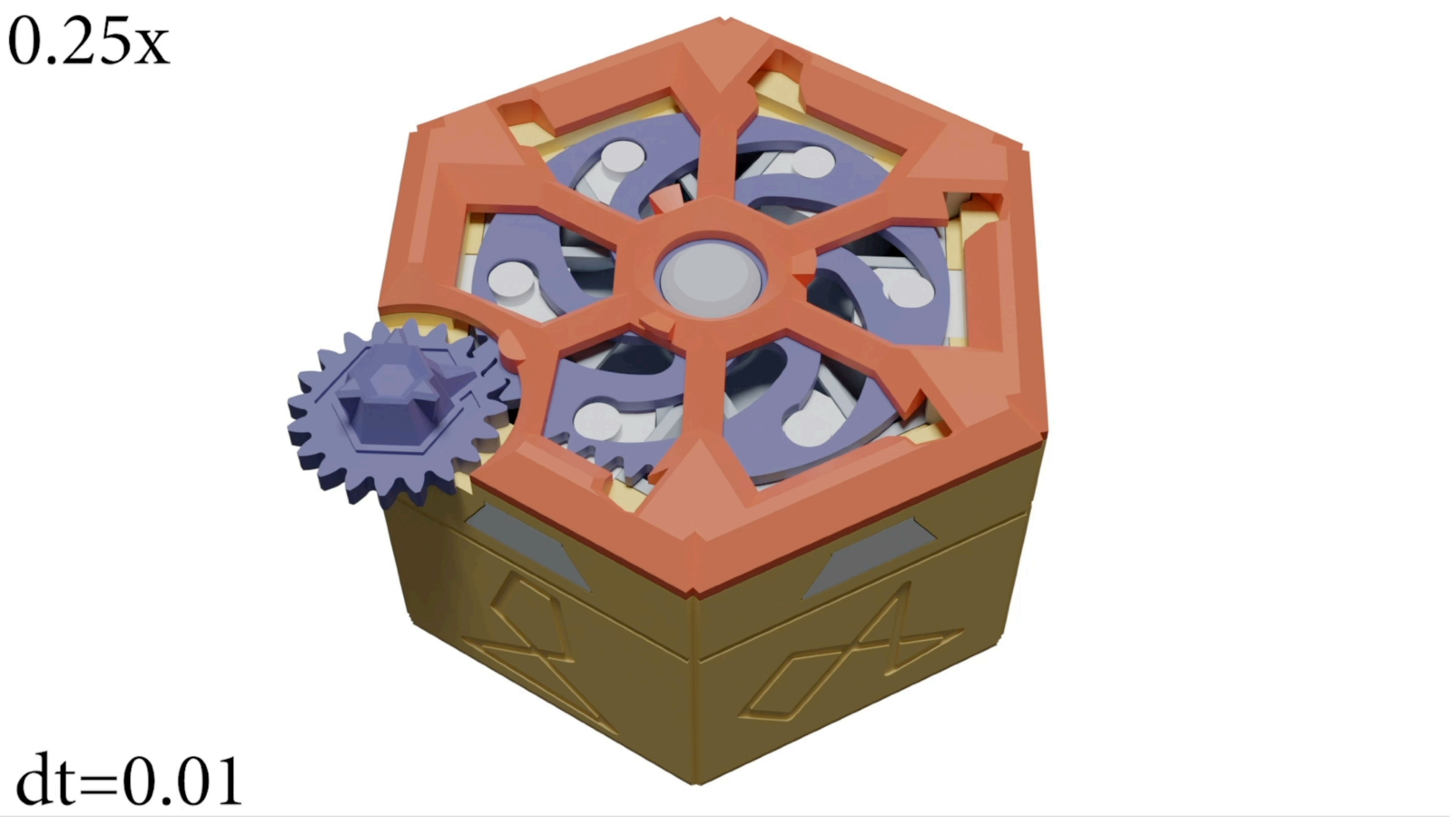
Reduced order dynamics (from subspace optimization):

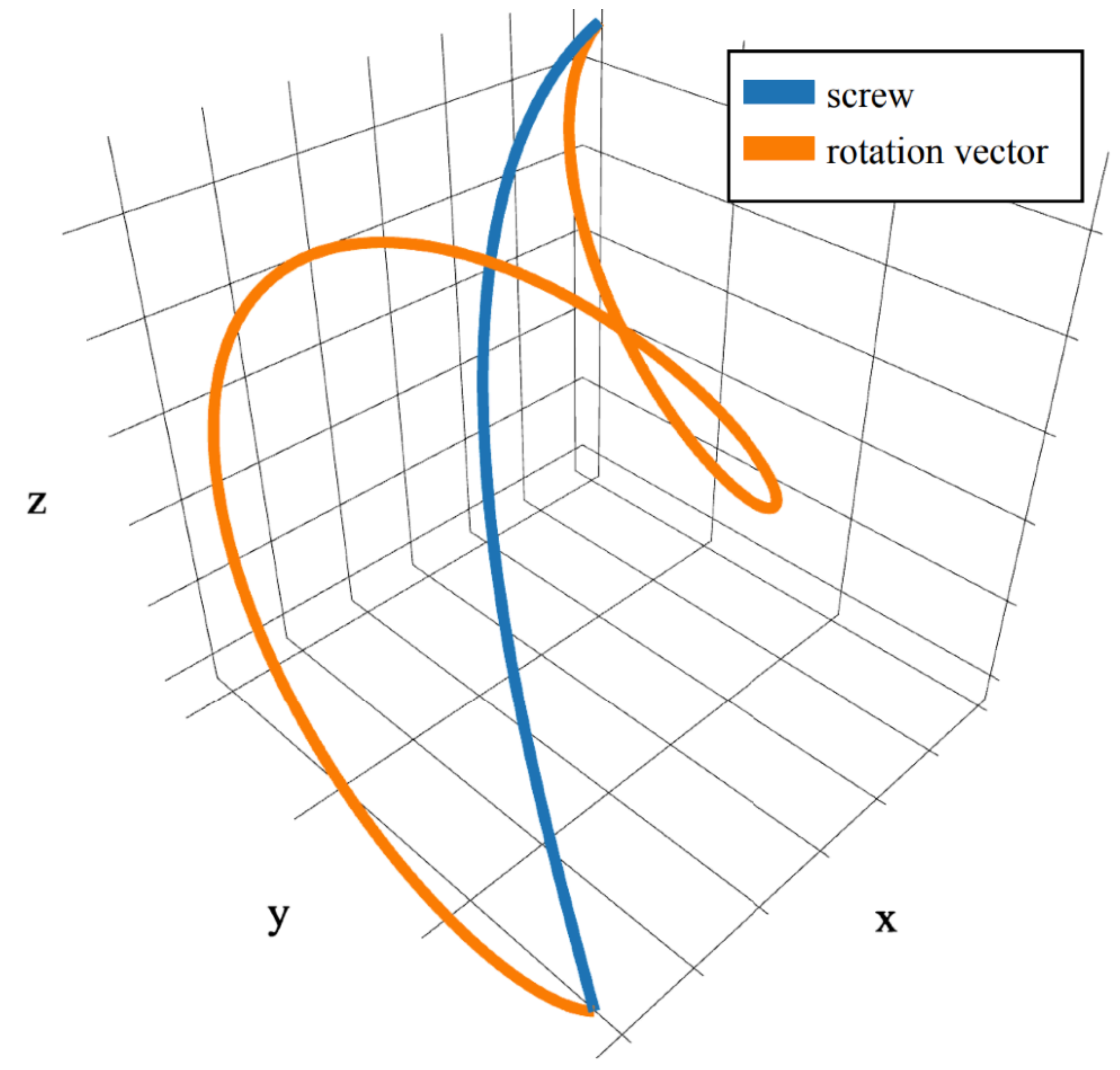
$$\min_{\theta,q} \frac{1}{2}\|\bar{X}R(\theta) + \bar{S}q - \tilde{x}^n\|_M^2 + h^2 \sum P(\bar{X}R(\theta) + \bar{S}q)$$

Just include IPC energies here

But line search is on θ, and $x(\theta)$ is nonlinear

So CCD is on nonlinear trajectories:



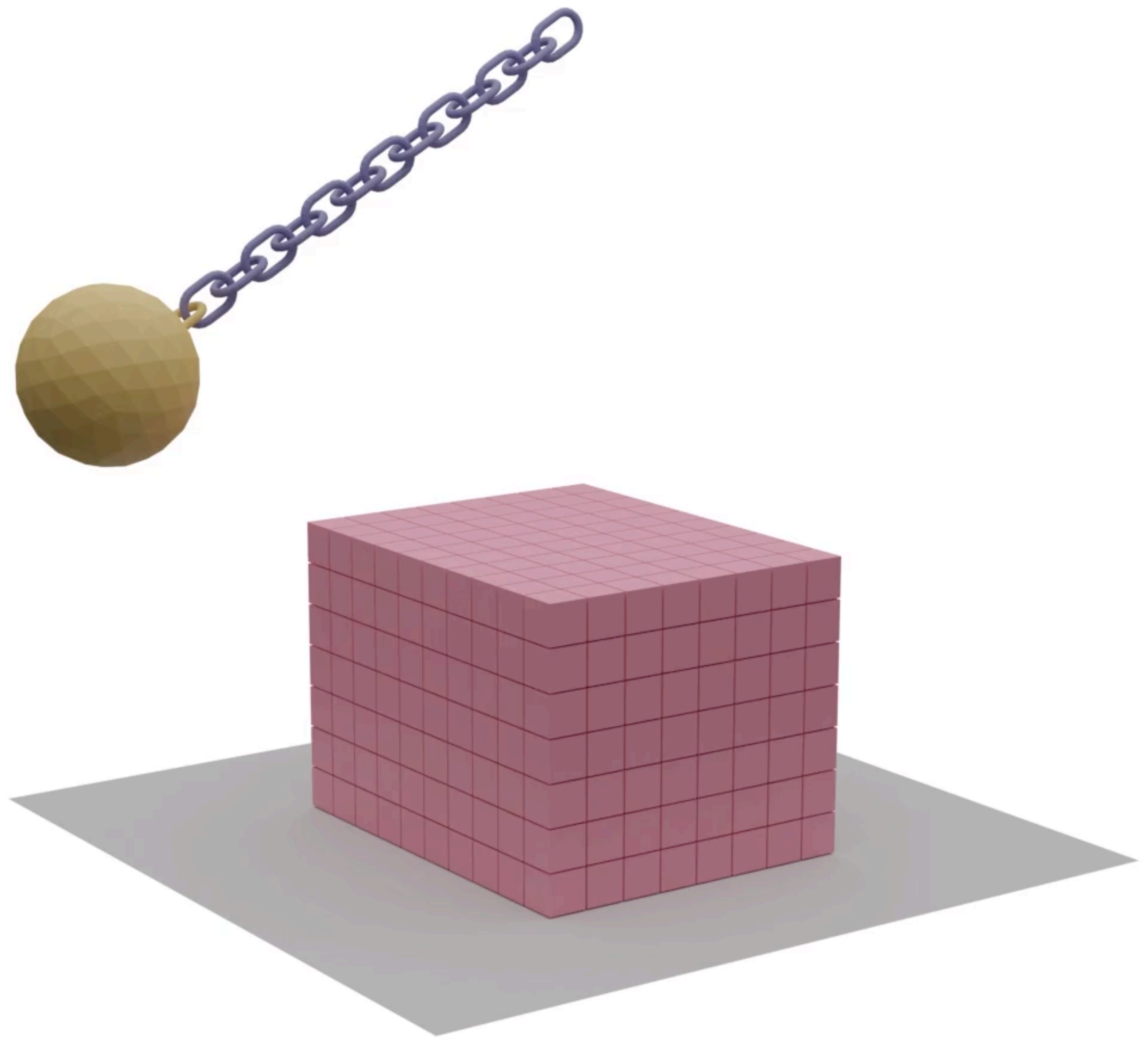


Very expensive!

Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois,Chenfanfu Jiang, Denis Zorin, Danny M. Kaufman, Daniele Panozzo. Intersection-free Rigid Body Dynamics. SIGGRAPH 2021.

Simulating Super Stiff Materials

Rigid-IPC [Ferguson et al. 2021] vs IPC [Li et al. 2020]



Example	runtime (s) (IPC)	runtime (s) (Rigid)	speed-up	iterations (IPC)	iterations (Rigid)
Pendulum	339.7	133.1	2.6x	10K	3K
Double pendulum	914.0	1559.9	0.6x	12K	4K
Arch (25 stones)	26.5	55.8	0.5x	2K	2K
Arch (101 stones)	238.3	487.8	0.5x	4K	5K
Wrecking ball	7179.8	5748.1	1.2x	9K	18K

Rigid-IPC performs well for complex geometries

Simulating Super Stiff Materials

Enforcing Rigidity via Penalty Method

Reduced order dynamics (from subspace optimization):

$$\min_{Q,q} \frac{1}{2} \|\bar{X}Q + \bar{S}q - \tilde{x}^n\|_M^2 + h^2 \boxed{\sum P(\bar{X}Q + \bar{S}q)} \quad \text{s.t.} \quad \mathbf{Q}^T\mathbf{Q} = \mathbf{I}\ \forall \mathbf{Q}\ (\text{or } f(Q) = 0)$$

Don't need elasticity

Reduced order dynamics with penalty method:

$$\min_{Q,q} \frac{1}{2} \|\bar{X}Q + \bar{S}q - \tilde{x}^n\|_M^2 + h^2 \boxed{\sum P(\bar{X}Q + \bar{S}q)}$$

Use elasticity with large Young's modulus

— the strain energy Ψ is effectively a penalty function for

12 DOF per body, still significantly reduced

$x = \bar{X}Q + \bar{S}q$ is linear w.r.t. both Q and q -> linear CCD

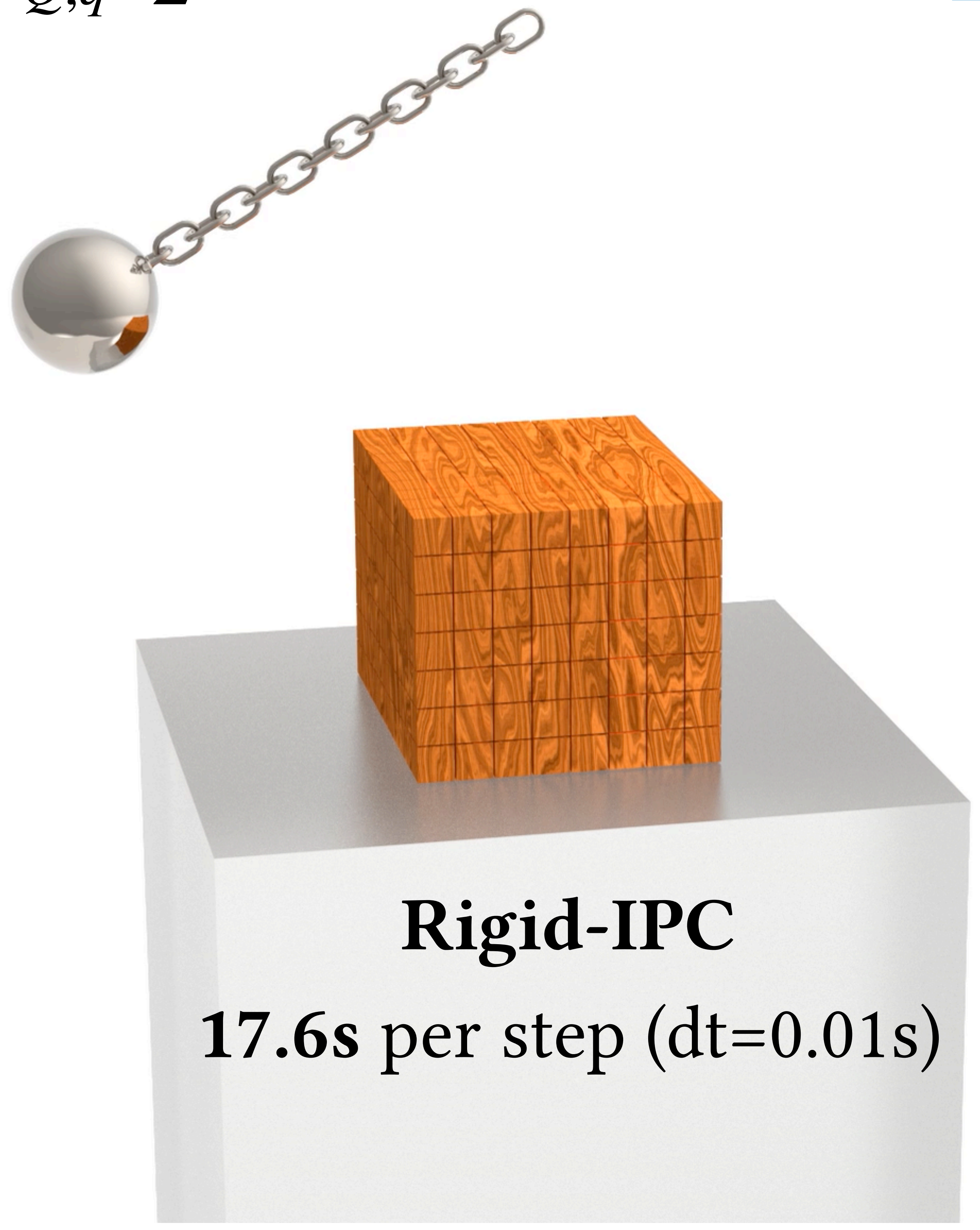
A stiff Ψ won't make the problem harder with stiff IPC energies

Simulating Super Stiff Materials

Affine Body Dynamics (ABD)

$$\min_{Q,q} \frac{1}{2}\|\bar{X}Q + \bar{S}q - \tilde{x}^n\|_M^2 + h^2 \boxed{\sum P(\bar{X}Q + \bar{S}q)}$$

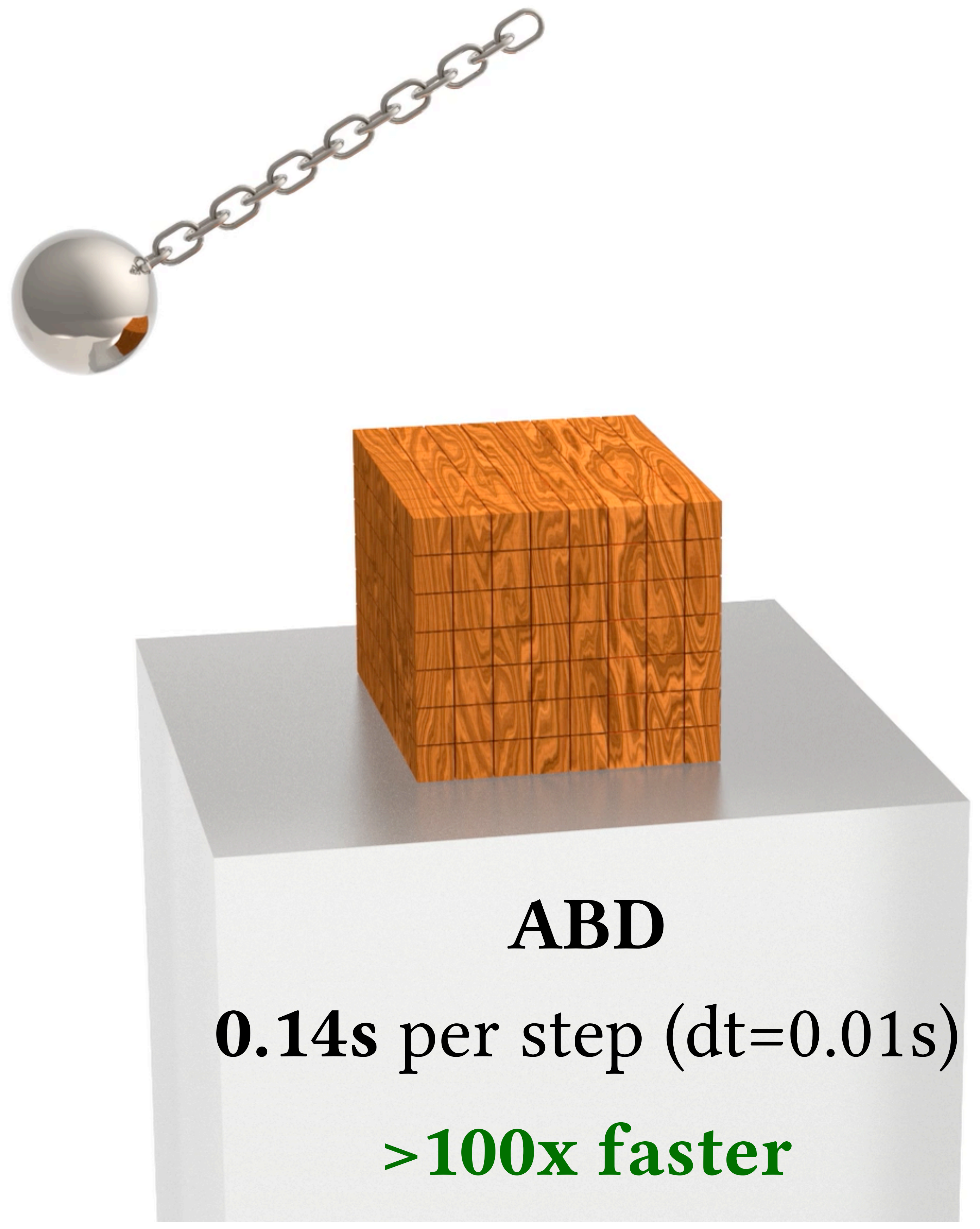
Use elasticity with large Young's modulus



Rigid-IPC

17.6s per step (dt=0.01s)

14K triangles
575 bodies



ABD

0.14s per step (dt=0.01s)

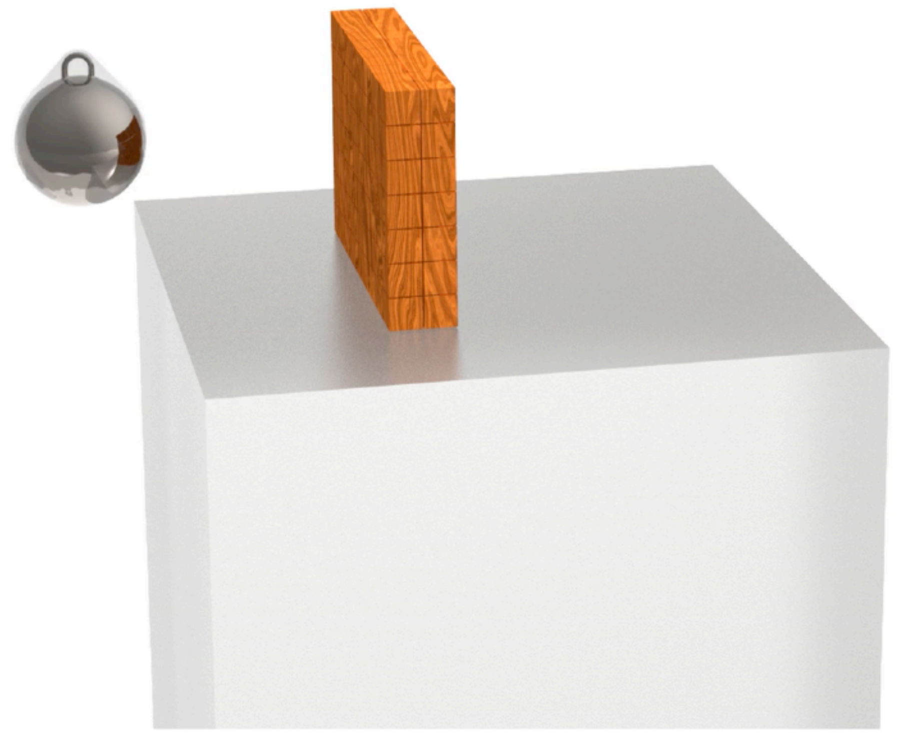
>100x faster

Lei Lan, Danny M. Kaufman, Minchen Li, Chenfanfu Jiang, Yin Yang. Affine Body Dynamics: Fast, Stable & Intersection-free Simulation of Stiff Materials. SIGGRAPH 2022.

Simulating Super Stiff Materials

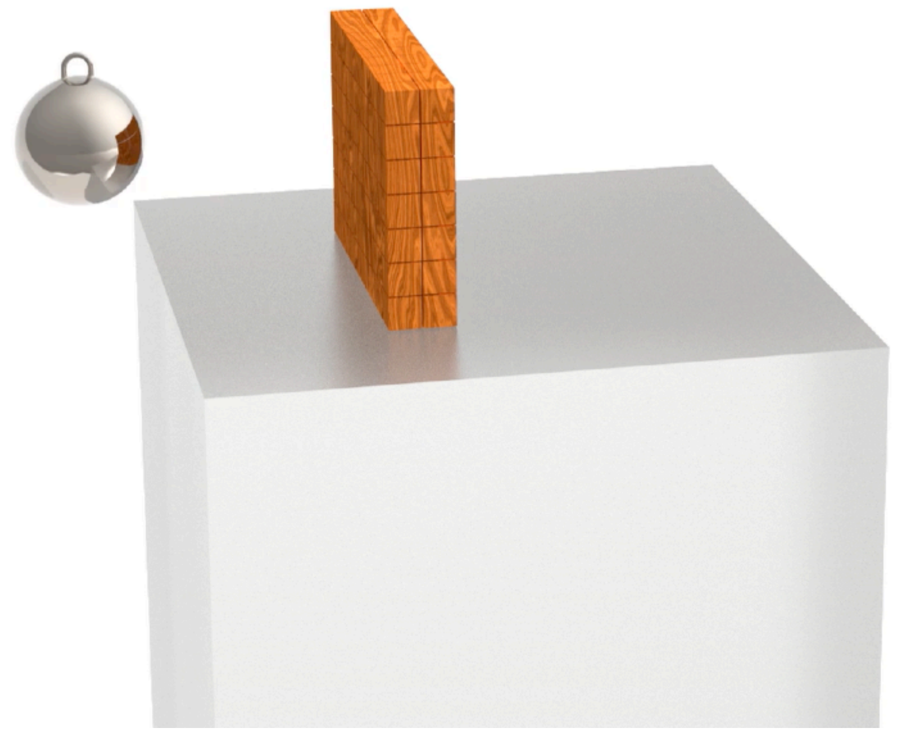
Bullet v.s. ABD

3.5K triangles
142 bodies



Bullet

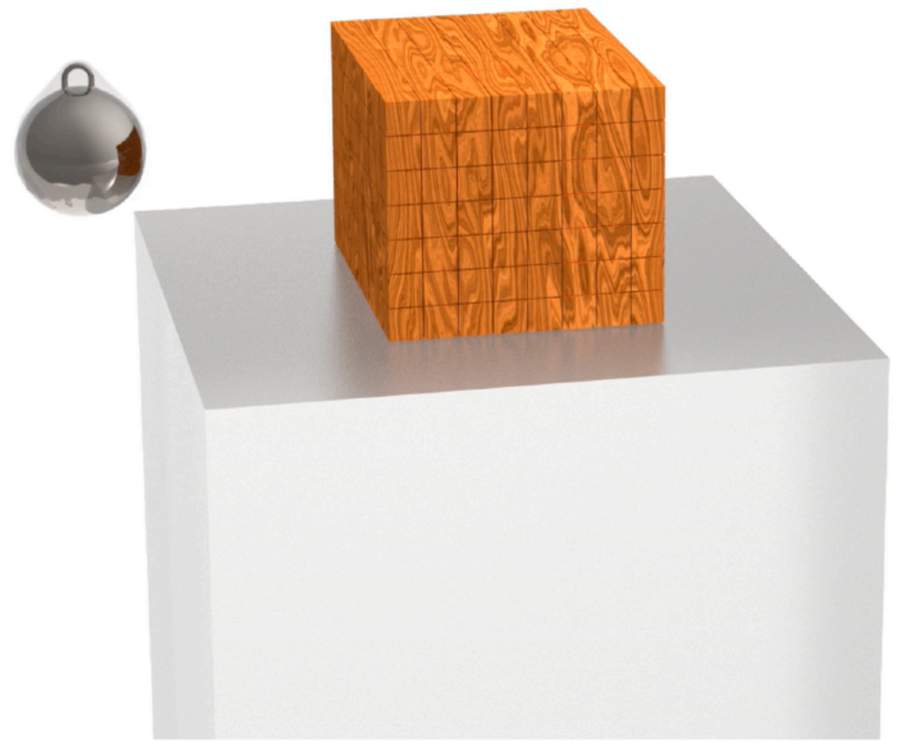
58ms per 1/240s step
82ms per 1ms step



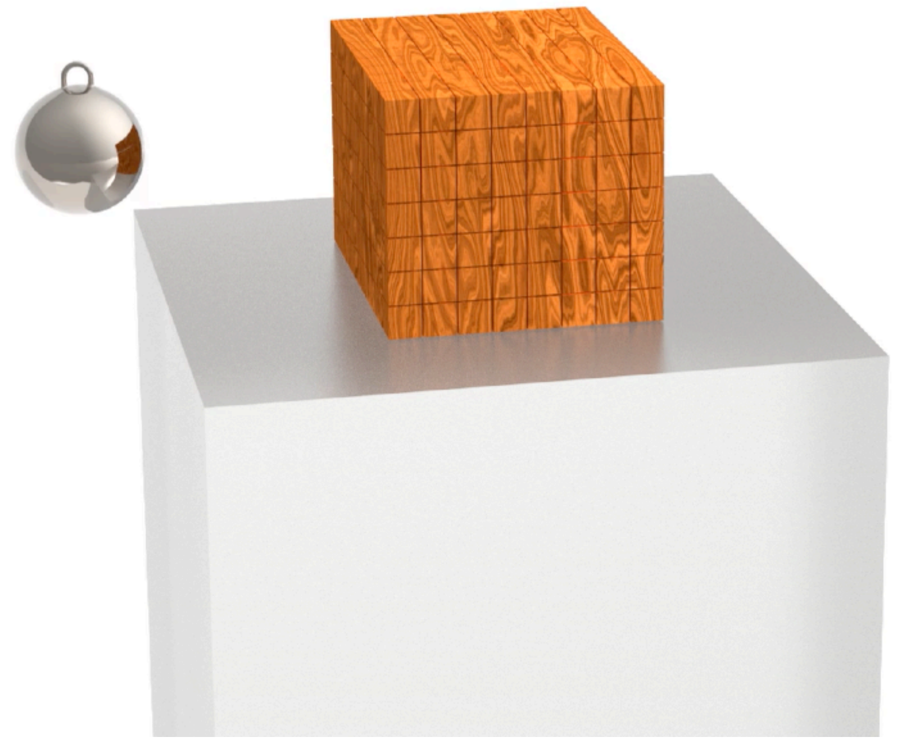
ABD

41ms per 1/240s step
19ms per 1ms step
>4x faster

11K triangles
562 bodies



809ms per 1/240s step
804ms per 1ms step



328ms per 1/240s step
102ms per 1ms step
>8x faster

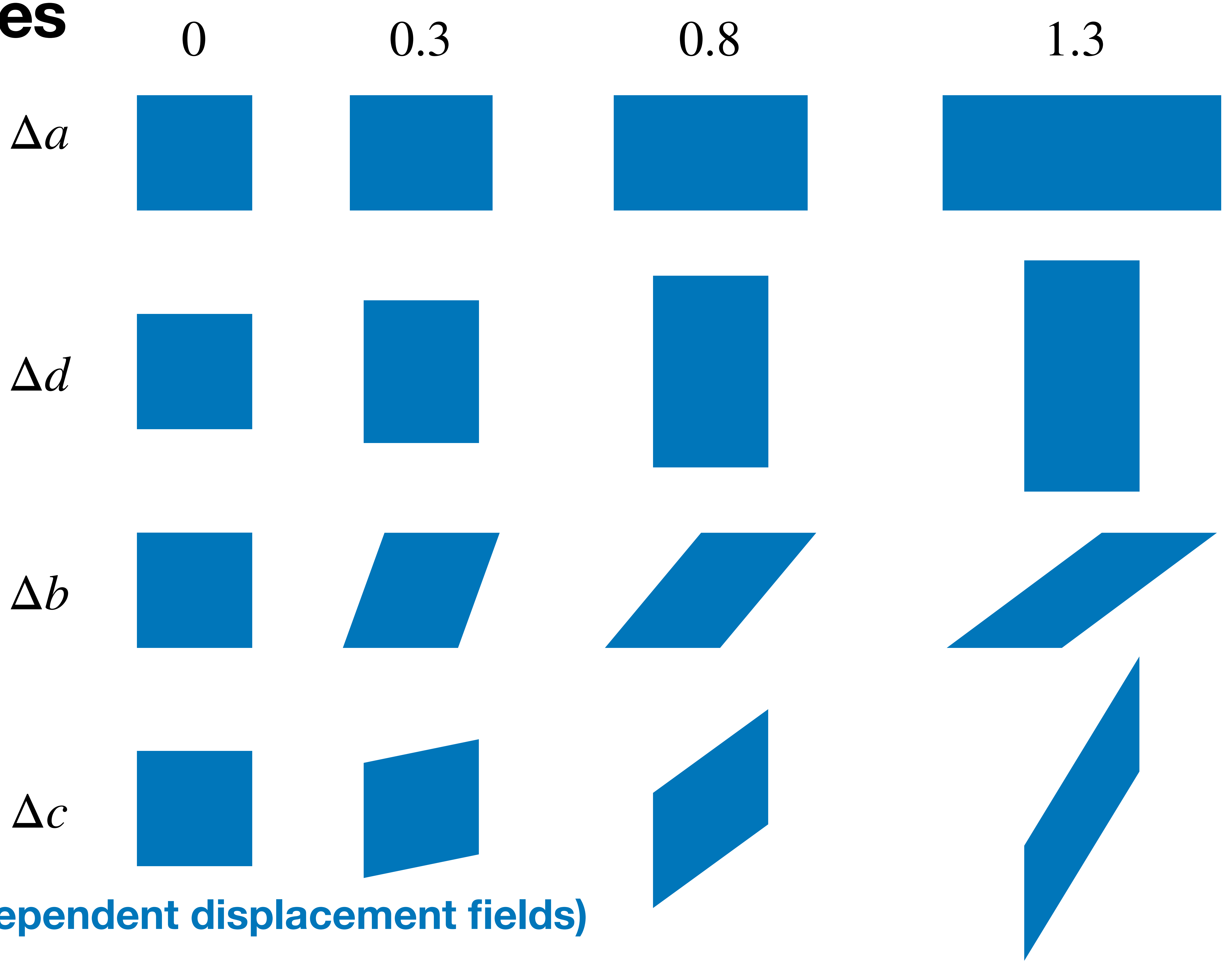
ABD in Another Perspective

Affine Deformation Modes

$$\mathbf{x} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \mathbf{X} + \begin{bmatrix} e \\ f \end{bmatrix}$$

DOF: a, b, c, d, e, f

X:

x: 

$$\mathbf{x} = A \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = aA_1 + bA_2 + \ldots$$

Deformation modes (linearly independent displacement fields)

Today:

- Simulating Stiff Elastic Solids
  SE(3), Affine Body, Deformation Modes
- Modal-Order Reduction
- Implementation & Demo

Reduced Simulation of Deformable Solids

Linear Modal Analysis

Deformation modes (linearly independent displacement fields)

$$\mathbf{x} = A\begin{bmatrix}a\\b\\c\\d\\e\\f\end{bmatrix} = a\overset{\downarrow}{A_1} + b\overset{\downarrow}{A_2} + \overset{\searrow}{\dots}$$

How do we generate more meaningful deformation modes?

Assume linear elasticity problem: $M\ddot{u} + Ku = f$ **s.t.** $Sx = 0$ **(Dirichlet BC)**

Intuition: Meaningful deformation modes are those don't generate large forces

Can solve the generalized Eigenvalue problem to find them: $\bar{K}y = \lambda \bar{M}y$

(where $\bar{K}$ and $\bar{M}$ do not account for BC nodes)

(Take the Eigenvectors with smallest Eigenvalues as modes.)

Reduced Simulation of Deformable Solids

Linear Modal Analysis: Time Integration

$$\mathbf{x} = A \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = a\underset{\uparrow}{A_1} + b\underset{\uparrow}{A_2} + \underset{\uparrow}{\dots}$$

Deformation modes (linearly independent displacement fields)

Can solve $\bar{K}y = \lambda \bar{M} y$ and take Eigenvectors with the smallest Eigenvalues as more modes.

$\boxed{\text{The Eigenvectors will be orthonormal w.r.t. } \bar{M}\text{, i.e. } (y^i)^T M y^j = \delta_{ij}.}$

Now let $u = x - X = Uz$, where $z \in \mathbf{R}^k$ are the reduced DOF, $U \in \mathbb{R}^{3n\times k}$ formed by the Eigenvectors

Plugging in $M\ddot{u} + Ku = f$, ignoring BCs for now:

$$MU\ddot{z} + KUz = f$$

$$MU\ddot{z} + MU\Lambda z = f$$

$\Lambda \in \mathbb{R}^{k\times k}$ **is a diagonal matrix of Eigenvalues**

$$U^T MU\ddot{z} + U^T MU\Lambda z = U^T f$$

Left-multiply U^T on both sides

$$\ddot{z} + \Lambda z = U^T f$$

Diagonal system! Super fast!

Reduced Simulation of Deformable Solids

Linear Modal Analysis: Effectiveness

Works well for small deformations:

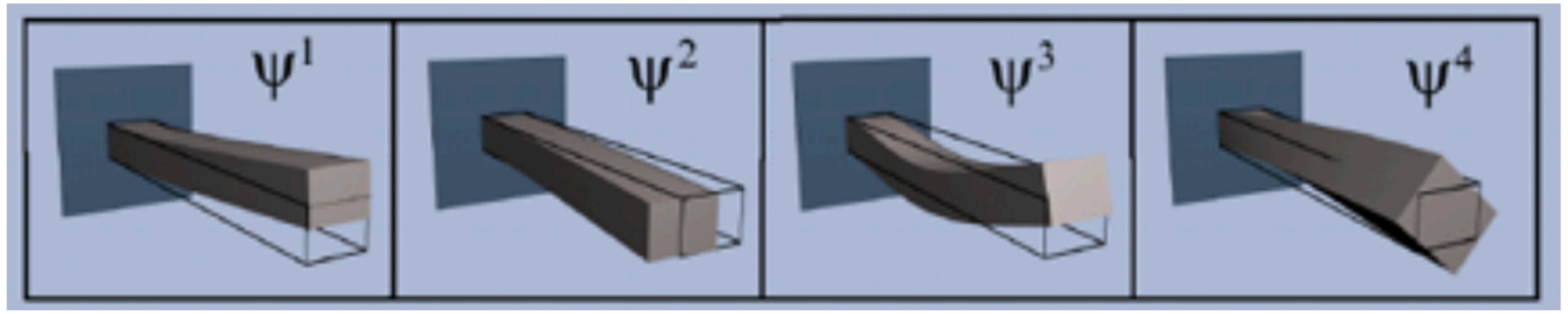


Figure 2: **Linear modes** for a cantilever beam.

— Consistent with our knowledge of linear elasticity

However:

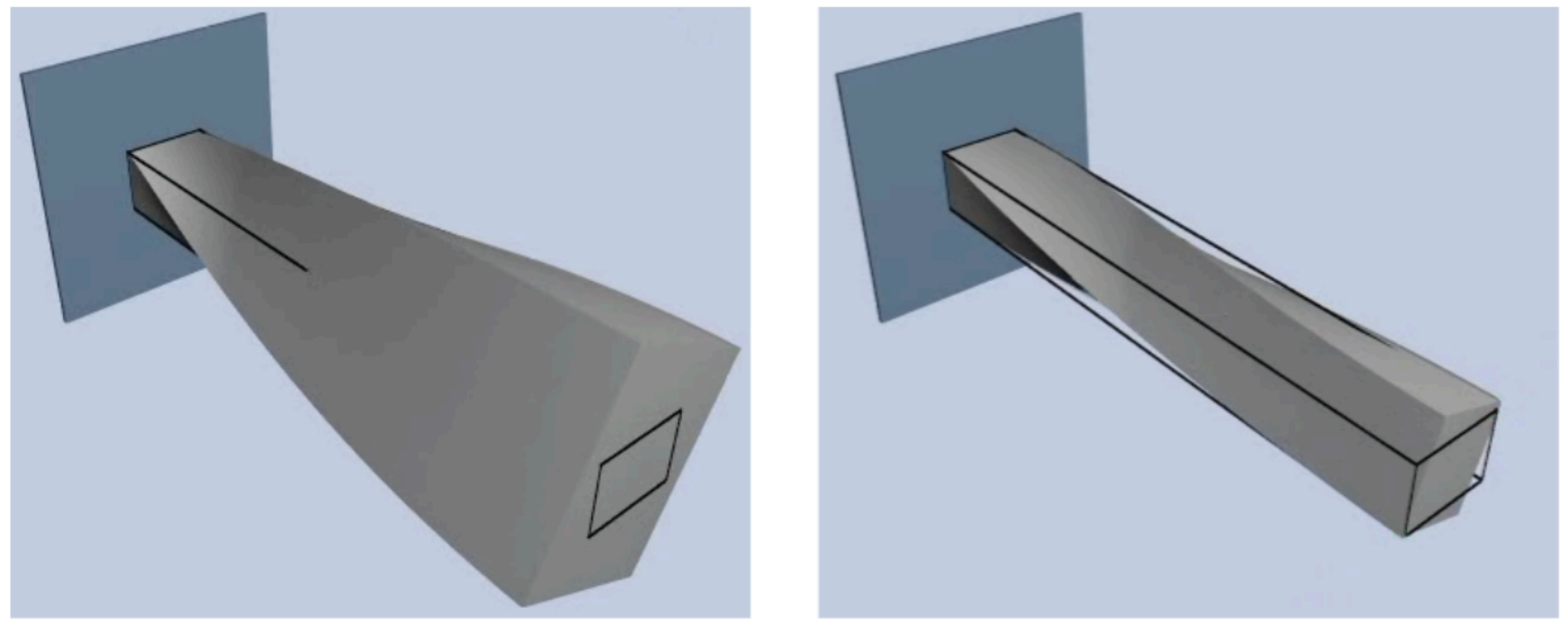


Figure 3: **Model reduction applied to a linear and nonlinear system.**

Reduced Simulation of Deformable Solids

Nonlinear Elasticity, Linear Modes

$M\ddot{u} + f^{int}(u) = f$ **or equivalently, using Incremental Potential:** $\min_x \frac{1}{2}\|x - \tilde{x}^n\|_M^2 + h^2 \sum P(x)$

Plugging in $u = Uz$: $\min_z \frac{1}{2}\|X + Uz - \tilde{x}^n\|_M^2 + h^2 \sum P(X + Uz)$ **(Can compute U using $\nabla^2 P(X)$)**

Gradient: $U^T M(X + Uz - \tilde{x}^n) + h^2 \sum U^T \nabla P(X + Uz)$

Hessian: $U^T M U + h^2 \sum U^T \nabla^2 P(X + Uz) U$

Issue 1: Hessian can be dense!

Solution: use locally supported modes, e.g. Cage-based deformation, Medial Axis Mesh [Lan et al. 2021], etc.

Issue 2: Calculating ∇P and $\nabla^2 P$ are still slow (requiring full space computations)

Solution: use numerical integration to approximate Gradient and Hessian, minimizing the number of quadratures [An et al. 2008]

Lei Lan, Yin Yang, Danny M. Kaufman, Junfeng Yao, Minchen Li, Chenfanfu Jiang. Medial IPC: Accelerated Incremental Potential Contact With Medial Elastics. SIGGRAPH 2021.

Steven S. An, Theodore Kim and Doug L. James, Optimizing Cubature for Efficient Integration of Subspace Deformations. SIGGRAPH Asia 2008.

Reduced Simulation of Deformable Solids

Nonlinear Elasticity, Linear Modes

$$\min_z \frac{1}{2}\|X + Uz - \tilde{x}^n\|_M^2 + h^2 \sum P(X + Uz)$$

Issue 3: modes computed at rest shape (using $\nabla^2 P(X)$)
can result in artificial stiffening at large deformation

Solution 1: use simulated poses/deformed configurations as data, and perform PCA to construct U

Solution 2: use nonlinear modes $u = f(z)$ where f is a nonlinear function

e.g. in rigid body dynamics, $u = f(\theta)$ is nonlinear

Use modal derivatives to construct a quadratic function $u = f(z)$ [*]

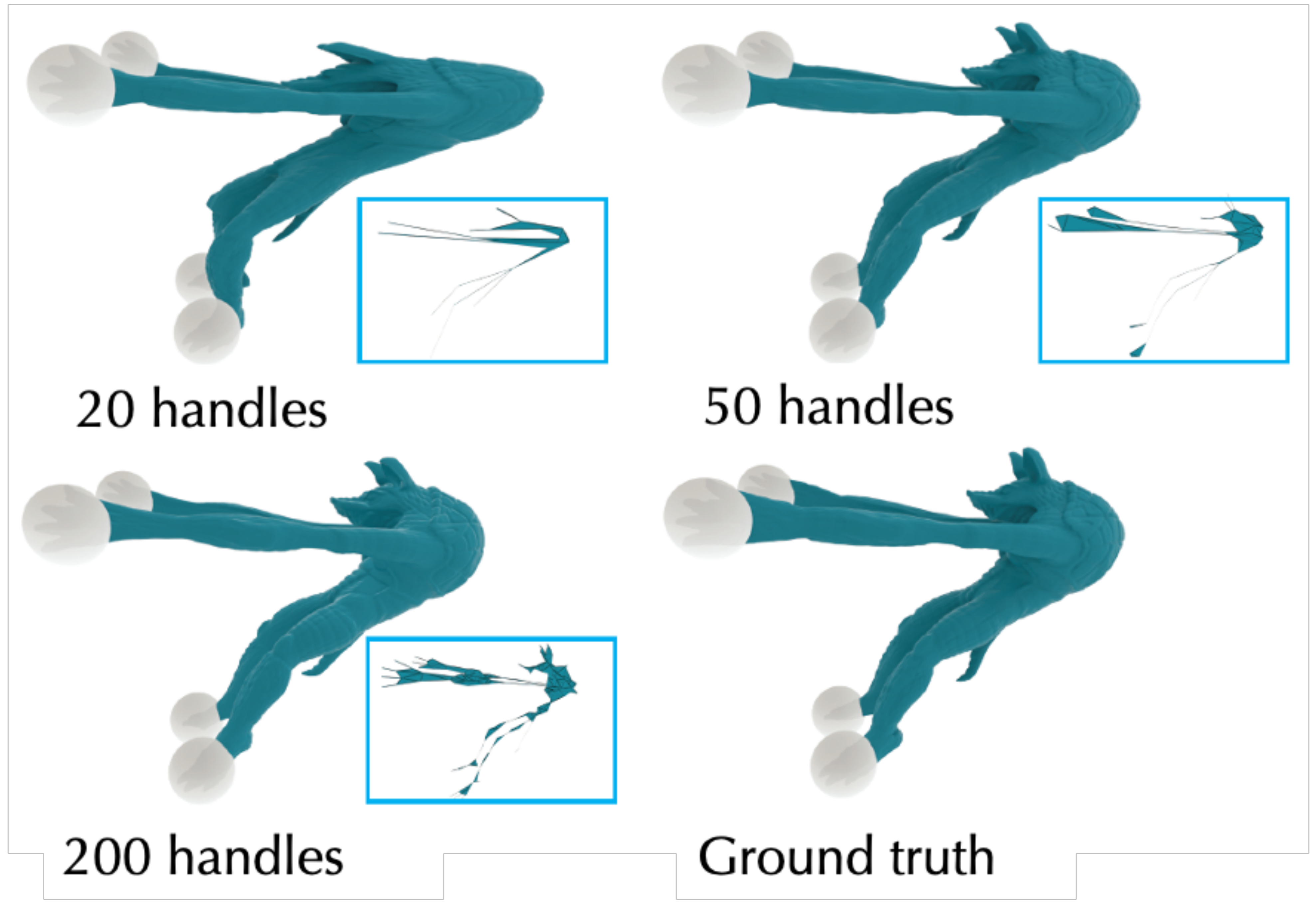
Use neural networks to learn $u = f(z)$

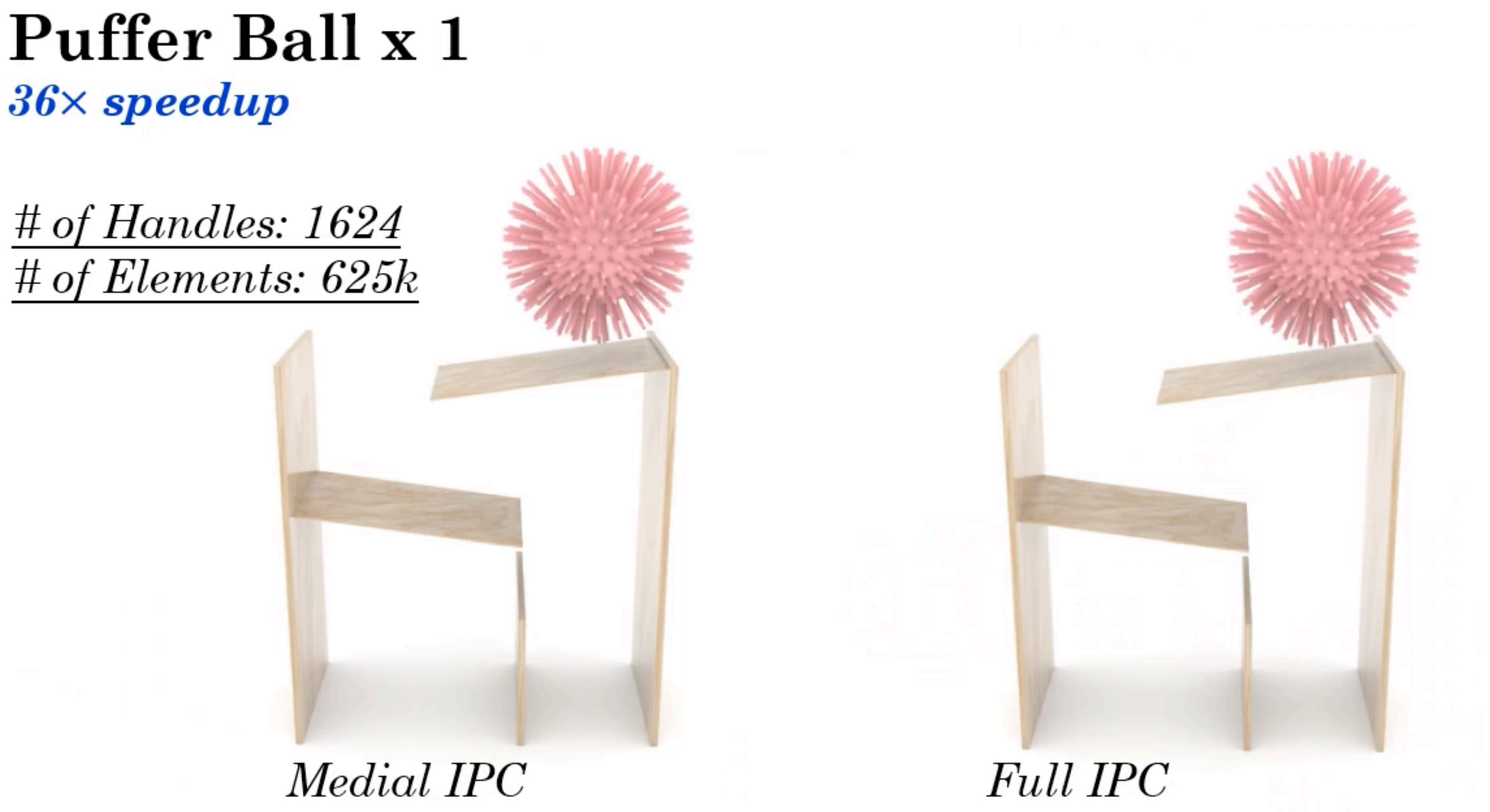
Remarks: Affine modes are linear modes, and are spatially linear;
PCA and Eigen modes are linear modes, but can be spatially nonlinear.

*Jernej Barbicˇ, Doug James. Real-Time Subspace Integration for St.Venant-Kirchhoff Deformable Models. SIGGRAPH 2005.

Reduced Simulation of Deformable Solids

Results from Medial IPC [Lan et al. 2021]





Lei Lan, Yin Yang, Danny M. Kaufman, Junfeng Yao, Minchen Li, Chenfanfu Jiang. Medial IPC: Accelerated Incremental Potential Contact With Medial Elastics. SIGGRAPH 2021.

Today:

- Simulating Stiff Elastic Solids

  SE(3), Affine Body, Deformation Modes

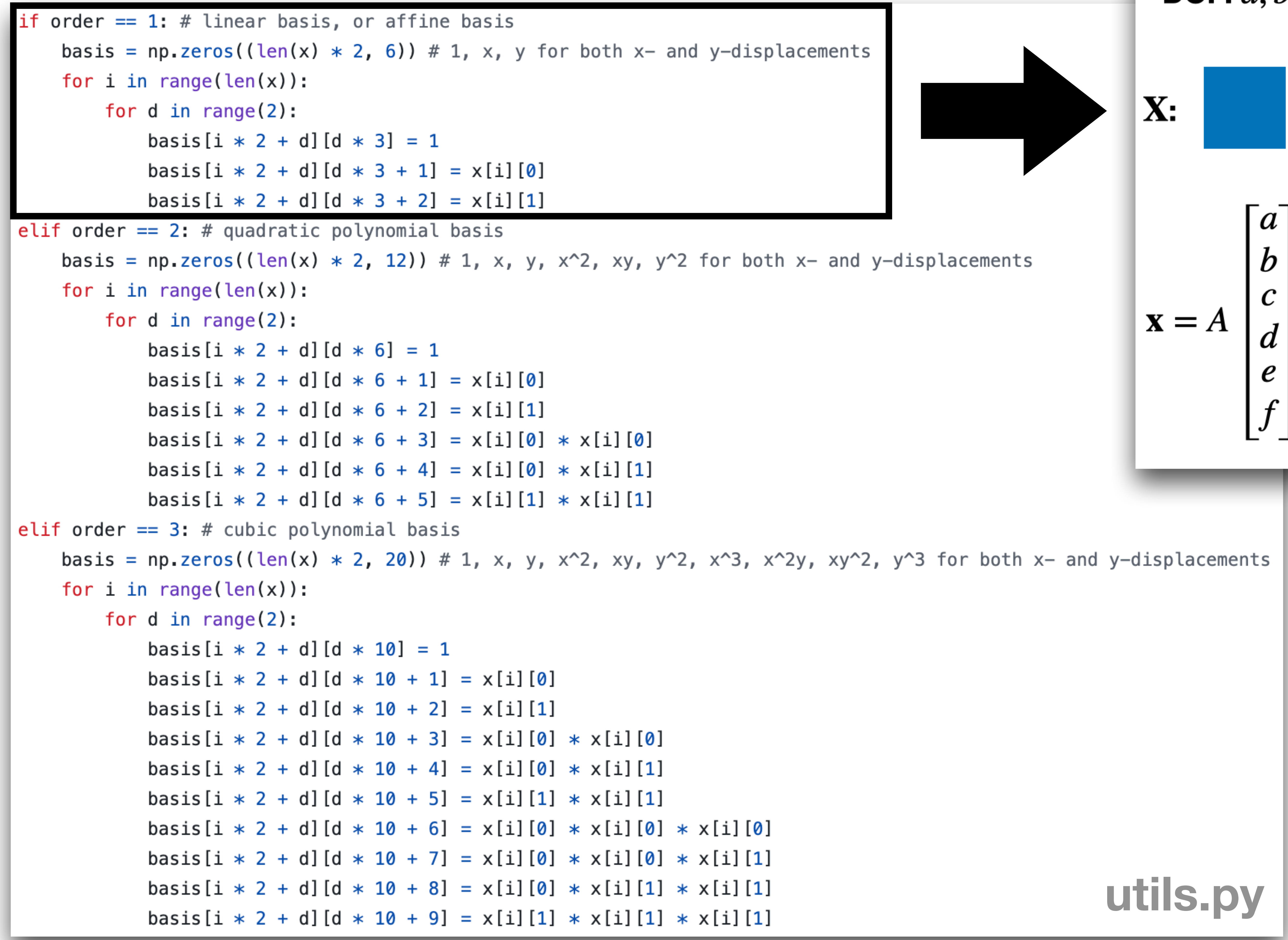
- Modal-Order Reduction

  Linear Modal Analysis for Linear and Nonlinear Elasticity

- Implementation & Demo

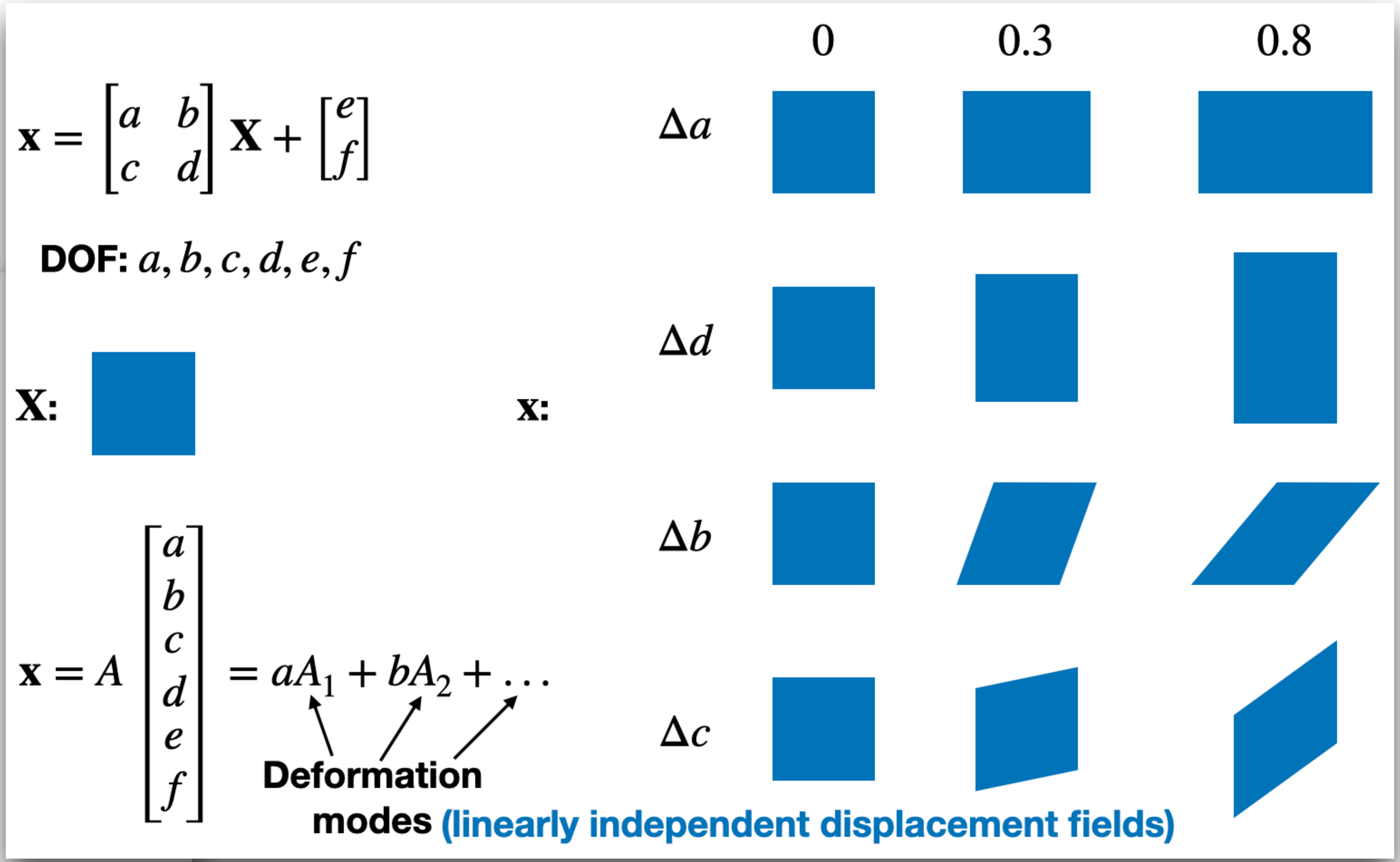
Implementation

Compute Basis (Polynomial)



```python
if order == 1: # linear basis, or affine basis
    basis = np.zeros((len(x) * 2, 6)) # 1, x, y for both x- and y-displacements
    for i in range(len(x)):
        for d in range(2):
            basis[i * 2 + d][d * 3] = 1
            basis[i * 2 + d][d * 3 + 1] = x[i][0]
            basis[i * 2 + d][d * 3 + 2] = x[i][1]
elif order == 2: # quadratic polynomial basis
    basis = np.zeros((len(x) * 2, 12)) # 1, x, y, x^2, xy, y^2 for both x- and y-displacements
    for i in range(len(x)):
        for d in range(2):
            basis[i * 2 + d][d * 6] = 1
            basis[i * 2 + d][d * 6 + 1] = x[i][0]
            basis[i * 2 + d][d * 6 + 2] = x[i][1]
            basis[i * 2 + d][d * 6 + 3] = x[i][0] * x[i][0]
            basis[i * 2 + d][d * 6 + 4] = x[i][0] * x[i][1]
            basis[i * 2 + d][d * 6 + 5] = x[i][1] * x[i][1]
elif order == 3: # cubic polynomial basis
    basis = np.zeros((len(x) * 2, 20)) # 1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3 for both x- and y-displacements
    for i in range(len(x)):
        for d in range(2):
            basis[i * 2 + d][d * 10] = 1
            basis[i * 2 + d][d * 10 + 1] = x[i][0]
            basis[i * 2 + d][d * 10 + 2] = x[i][1]
            basis[i * 2 + d][d * 10 + 3] = x[i][0] * x[i][0]
            basis[i * 2 + d][d * 10 + 4] = x[i][0] * x[i][1]
            basis[i * 2 + d][d * 10 + 5] = x[i][1] * x[i][1]
            basis[i * 2 + d][d * 10 + 6] = x[i][0] * x[i][0] * x[i][0]
            basis[i * 2 + d][d * 10 + 7] = x[i][0] * x[i][0] * x[i][1]
            basis[i * 2 + d][d * 10 + 8] = x[i][0] * x[i][1] * x[i][1]
            basis[i * 2 + d][d * 10 + 9] = x[i][1] * x[i][1] * x[i][1]
```

utils.py



$\mathbf{x} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \mathbf{X} + \begin{bmatrix} e \\ f \end{bmatrix}$
DOF: a, b, c, d, e, f
X:
x:
$\mathbf{x} = A \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = aA_1 + bA_2 + \ldots$
Deformation modes (linearly independent displacement fields)
0
0.3
0.8
Δa
Δd
Δb
Δc

For each 2 rows of A, find the corresponding node and its material space coordinates $(x,\ y)$, then use the following function to calculate A's entry in each column:

$$col_a(x, y) = (x,\ 0) \qquad col_d(x, y) = (0,\ y)$$

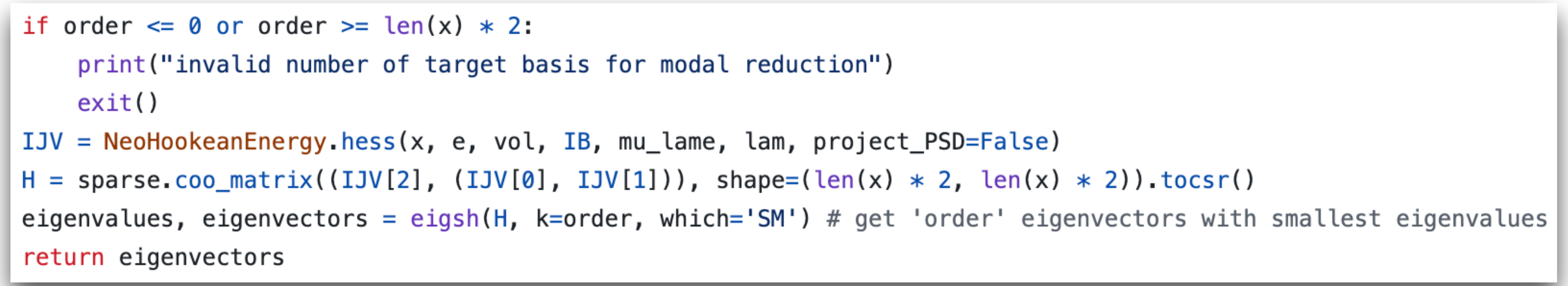
$$col_b(x, y) = (y,\ 0) \qquad col_e(x, y) = (1,\ 0)$$

$$col_c(x, y) = (0,\ x) \qquad col_f(x, y) = (0,\ 1)$$

Implementation

Compute Basis (Modal)

- For simplicity, we directly use the Eigenvectors of $\nabla^2\Psi(X)$ (no PSD projection) with the smallest Eigenvalues as basis.



```
if order <= 0 or order >= len(x) * 2:
    print("invalid number of target basis for modal reduction")
    exit()
IJV = NeoHookeanEnergy.hess(x, e, vol, IB, mu_lame, lam, project_PSD=False)
H = sparse.coo_matrix((IJV[2], (IJV[0], IJV[1])), shape=(len(x) * 2, len(x) * 2)).tocsr()
eigenvalues, eigenvectors = eigsh(H, k=order, which='SM') # get 'order' eigenvectors with smallest eigenvalues
return eigenvectors
```

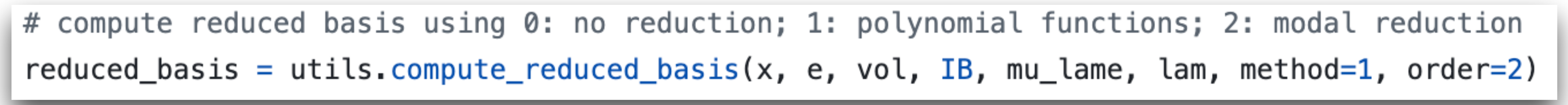
utils.py

Implementation

Simulation in the Subspace

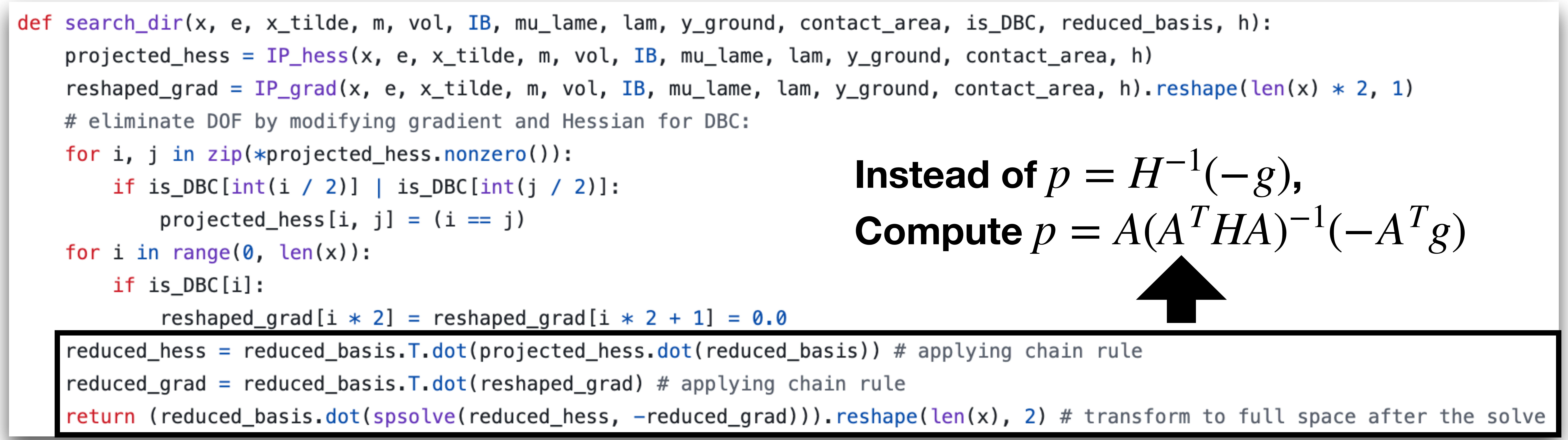
- Pick and compute the basis:

simulator.py



```
# compute reduced basis using 0: no reduction; 1: polynomial functions; 2: modal reduction
reduced_basis = utils.compute_reduced_basis(x, e, vol, IB, mu_lame, lam, method=1, order=2)
```

- Solve the time stepping optimization in the subspace:



```
def search_dir(x, e, x_tilde, m, vol, IB, mu_lame, lam, y_ground, contact_area, is_DBC, reduced_basis, h):
    projected_hess = IP_hess(x, e, x_tilde, m, vol, IB, mu_lame, lam, y_ground, contact_area, h)
    reshaped_grad = IP_grad(x, e, x_tilde, m, vol, IB, mu_lame, lam, y_ground, contact_area, h).reshape(len(x) * 2, 1)
    # eliminate DOF by modifying gradient and Hessian for DBC:
    for i, j in zip(*projected_hess.nonzero()):
        if is_DBC[int(i / 2)] | is_DBC[int(j / 2)]:
            projected_hess[i, j] = (i == j)
    for i in range(0, len(x)):
        if is_DBC[i]:
            reshaped_grad[i * 2] = reshaped_grad[i * 2 + 1] = 0.0
    reduced_hess = reduced_basis.T.dot(projected_hess.dot(reduced_basis)) # applying chain rule
    reduced_grad = reduced_basis.T.dot(reshaped_grad) # applying chain rule
    return (reduced_basis.dot(spsolve(reduced_hess, -reduced_grad))).reshape(len(x), 2) # transform to full space after the solve
```

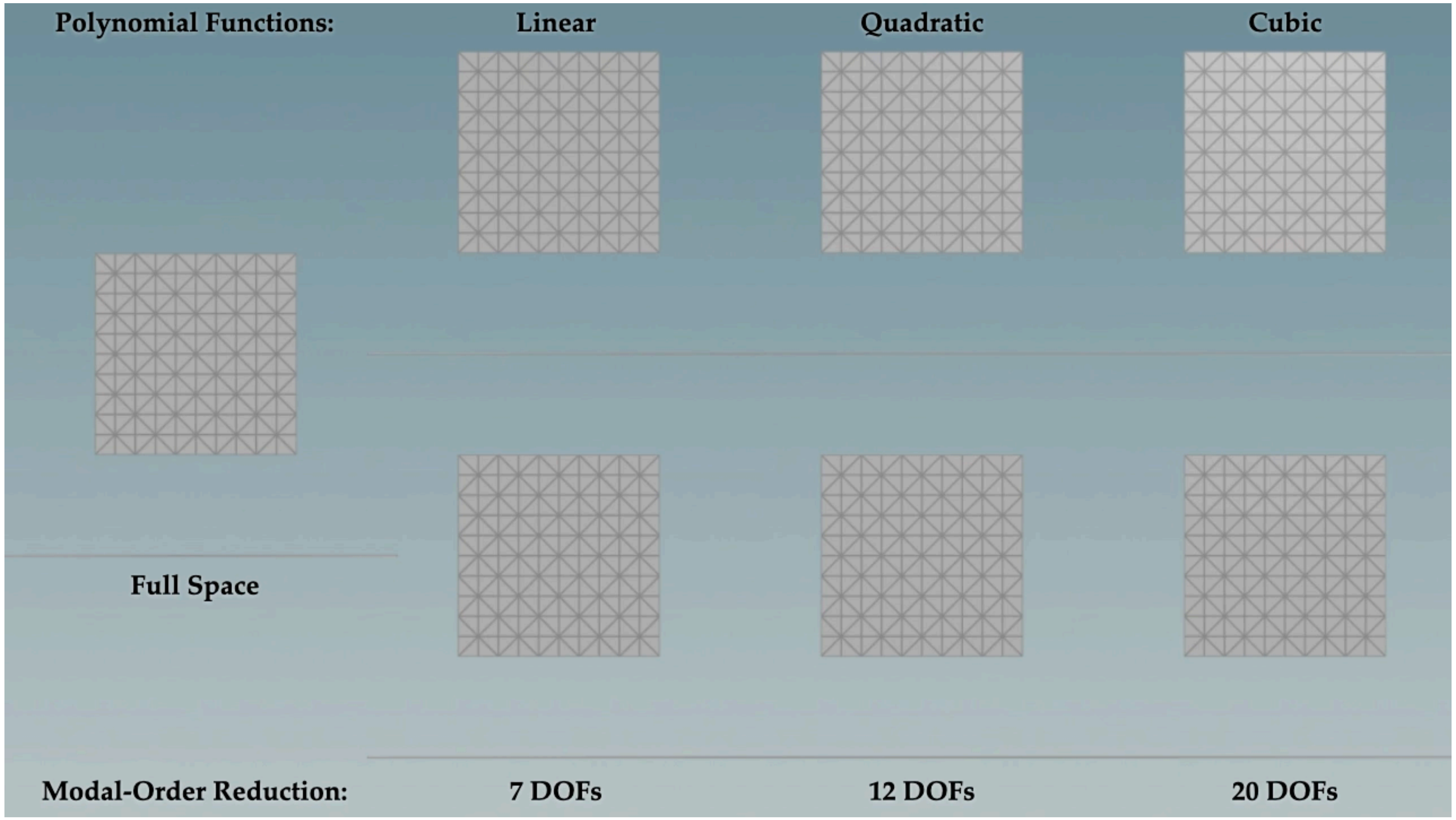
Instead of $p = H^{-1}(-g)$**,**

Compute $p = A(A^T H A)^{-1}(-A^T g)$

time_integrator.py

Demo!

Code: github.com/phys-sim-book/solid-sim-tutorial



Today:

- Simulating Stiff Elastic Solids

  SE(3), Affine Body, Deformation Modes

- Modal-Order Reduction

  Linear Modal Analysis for Linear and Nonlinear Elasticity

- Implementation & Demo

  Compute Basis, Simulation in the Subspace

Next Lecture: Codimensional Solids

Image Sources

- https://padeepz.net/ce6602-syllabus-structural-analysis-2-regulation-2013-anna-university/
- https://en.wikipedia.org/wiki/Young%27s_modulus
- http://viterbi-web.usc.edu/~jbarbic/femdefo/barbic-courseNotes-modelReduction.pdf