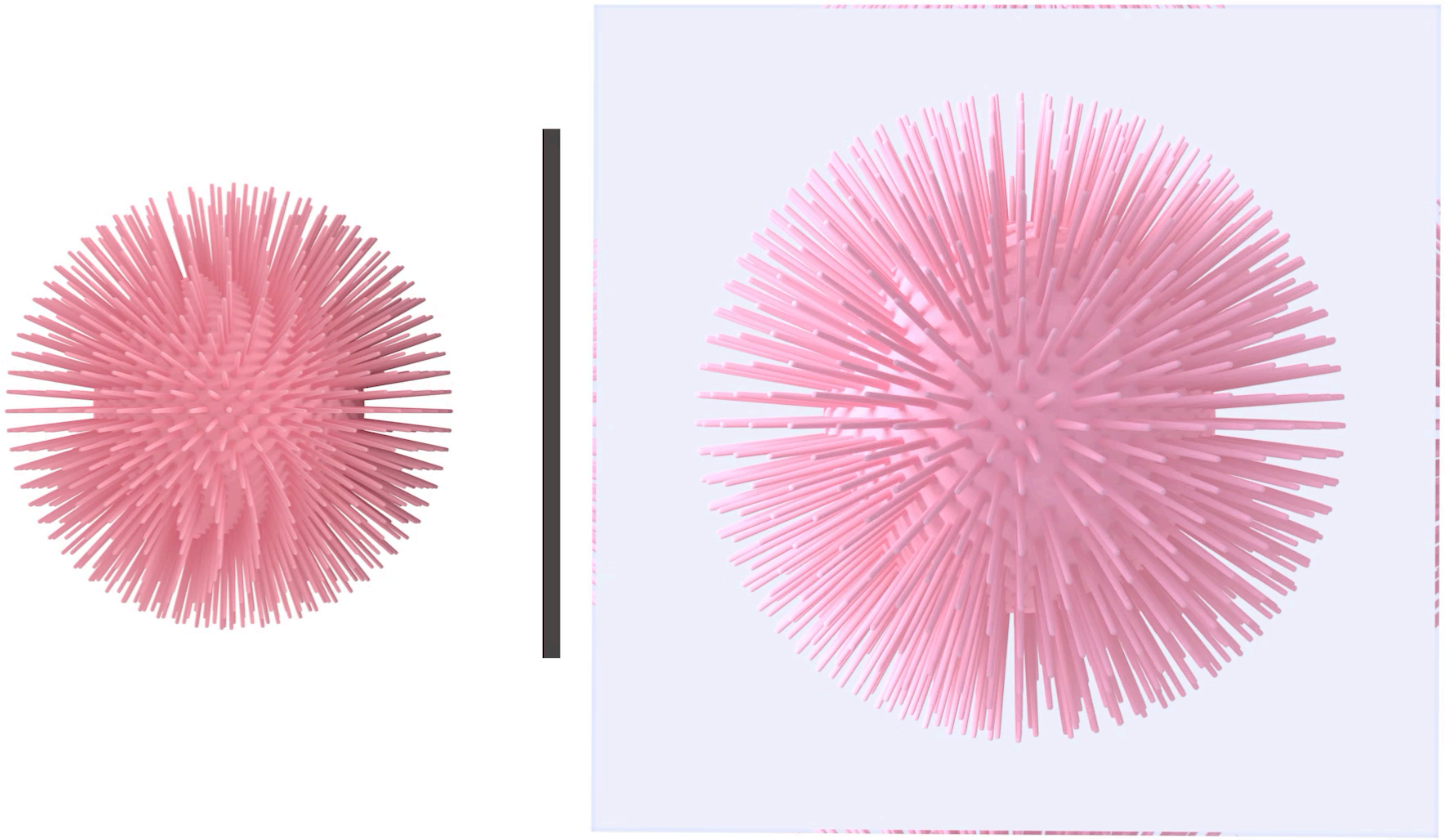
Instructor: Minchen Li



Lec 4: Normal Contact with Distance Barrier

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

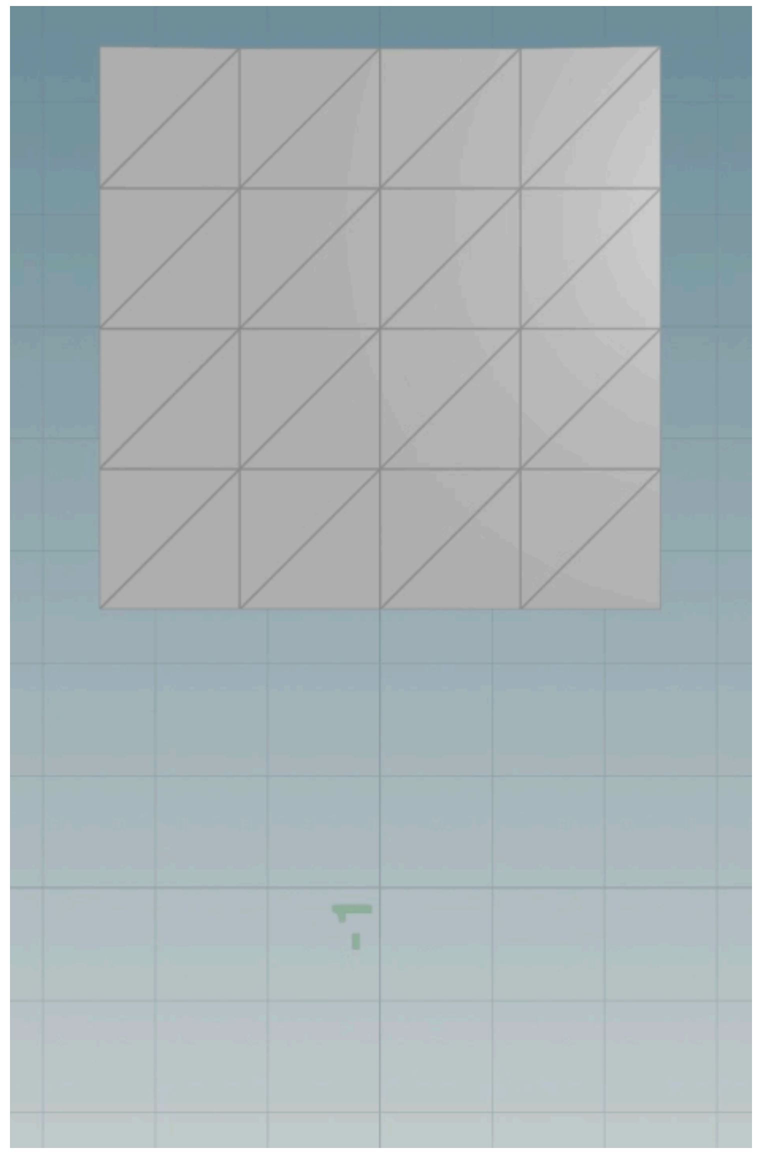
Recap: Dirichlet Boundary Conditions

- Equality constraints

Elasticity | Selecting BC DOF

$$\min_x \; E(x) = \boxed{\frac{1}{2}\|x - (x^n + hv^n)\|_M^2} + \boxed{h^2 P(x)}. \quad \text{s.t.} \quad \boxed{A}x = \boxed{b}$$

Inertia term | Prescribing BC values



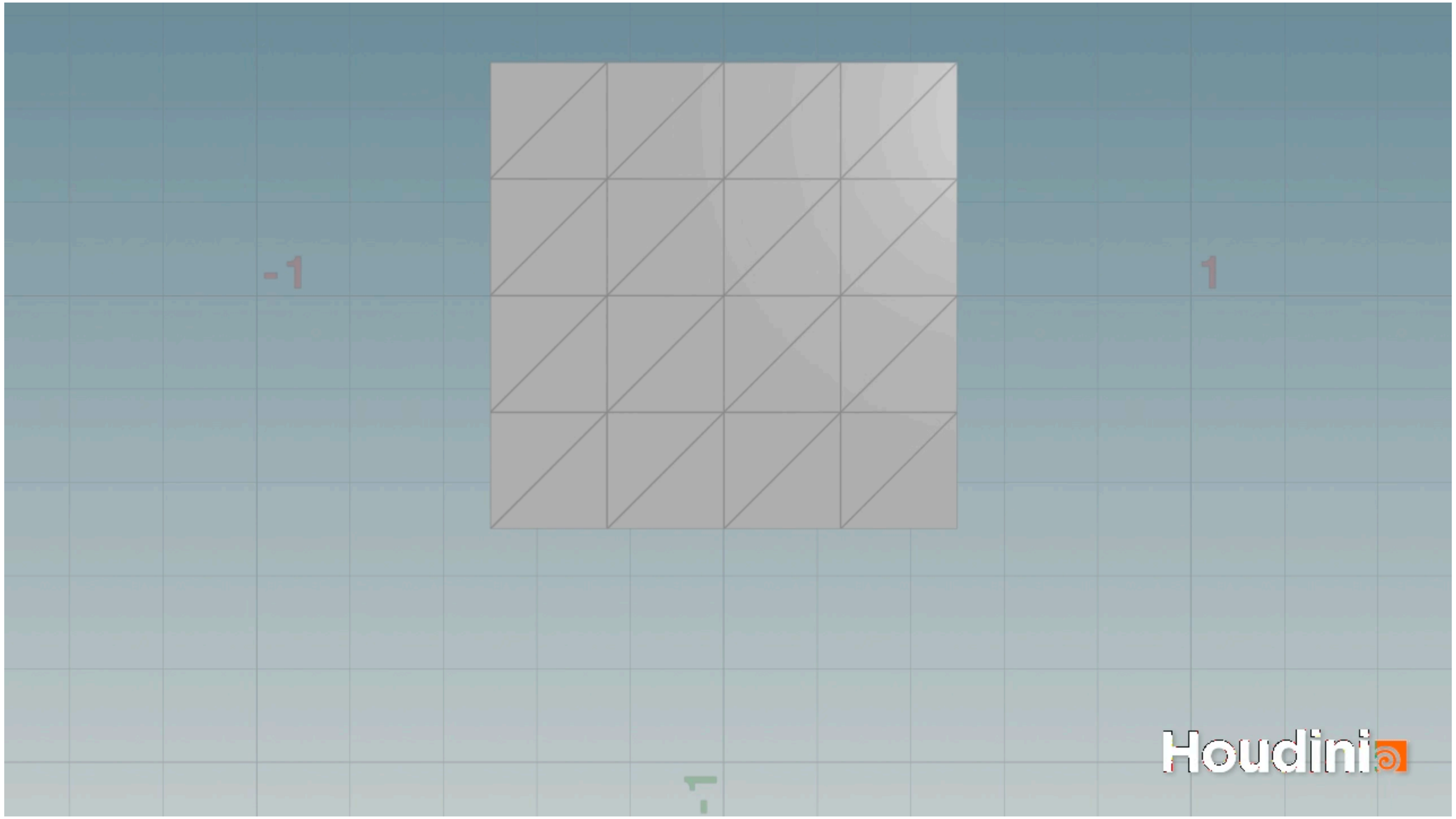
- Sticky v.s. Slip
- DOF Elimination Method

$$H = \begin{bmatrix} 4 & -1 & -1 & -1 \\ -1 & 4 & -1 & -1 \\ -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & 4 \end{bmatrix}, \quad \text{and} \quad g = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Fix node x_2 →

$$\begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_{11} \\ \Delta x_{12} \\ \Delta x_{21} \\ \Delta x_{22} \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \\ 0 \\ 0 \end{bmatrix}$$

Simulating Normal Contact



Today:

- Formulation
- Barrier Method
- Filtered Line Search
- Implementation & Demo
- Remarks

Today:

- Formulation
- Barrier Method
- Filtered Line Search
- Implementation & Demo
- Remarks

Formulation of Normal Contact

- With normal contact, add inequality constraints:

$$\min_x \; E(x) = \frac{1}{2}\|x - (x^n + hv^n)\|_M^2 + h^2 P(x). \quad \textbf{s.t.} \quad Ax = b \quad \textbf{and} \quad \forall k, \; d_k(x) \geq 0$$

e.g. for distances between any distinct points on the solid

- d_k: signed distance
  - e.g. signed distance from y to a sphere $\{x \mid \|x - c\| \leq r\}$:
    - $d(y) = \|y - c\| - r$
- (Unsigned distances to be covered in future lectures)

Formulation of Normal Contact

Solution Property

- At the solution x^* of

$$\min_x \; E(x) = \frac{1}{2}\|x - (x^n + hv^n)\|_M^2 + h^2 P(x). \quad \textbf{s.t.} \quad Ax = b \quad \textbf{and} \quad \forall k, \;\; d_k(x) \geq 0$$

- KKT Conditions are satisfied:

Contact force * h^2

$$\nabla E(x) - A^T\lambda - \sum_k \gamma_k \nabla d_l(x) = 0$$

$$Ax = b$$

Dual feasibility

$$\forall k, \;\; d_k(x) \geq 0, \;\; \gamma_k \geq 0, \;\; \gamma_k d_k(x) = 0$$

Primal feasibility **Complementary Slackness**

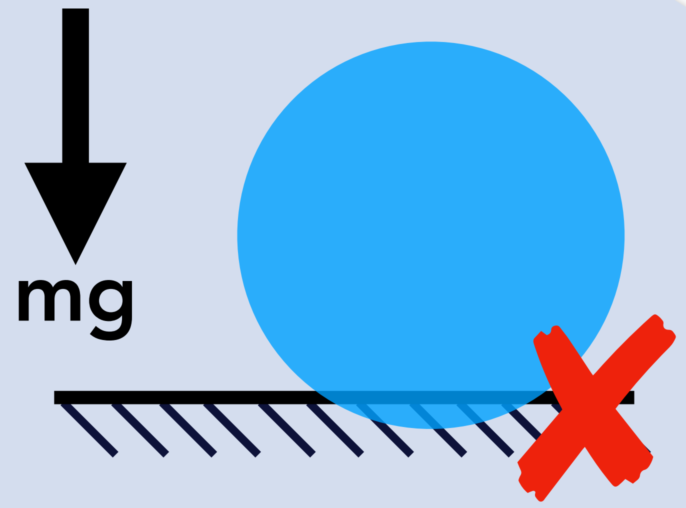
Formulation of Normal Contact

Solution Property — Intuition of KKT Conditions

Primal feasibility

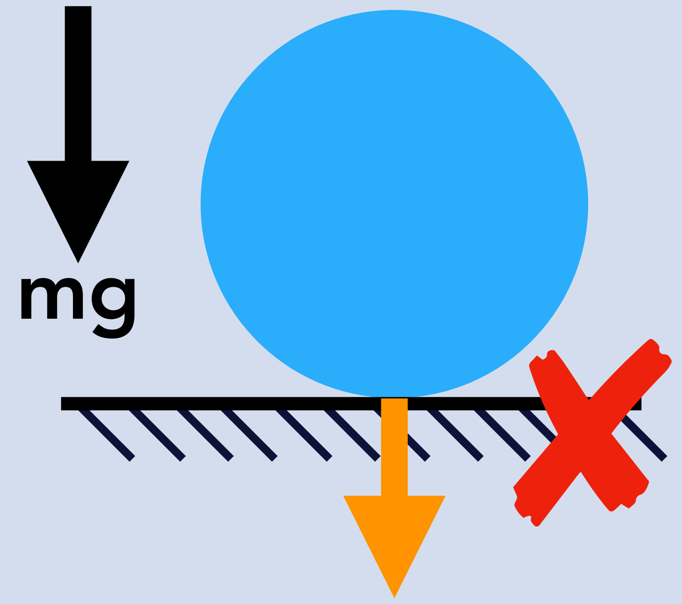
$$\forall k, \quad d_k \geq 0$$

Nonpenetration



Dual feasibility

$$\forall k, \quad \gamma_k \geq 0$$

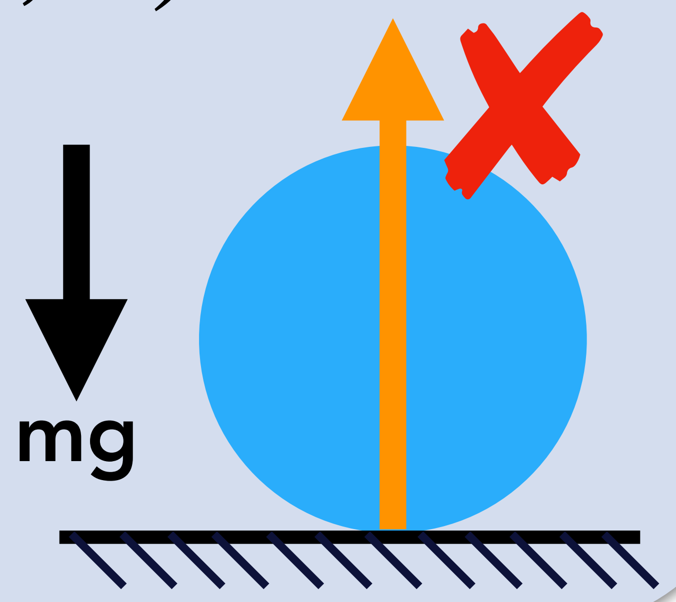


Contact force only push but not pull

Stationarity

$$\nabla E(x) + \kappa \sum_k \nabla b(d_k(x), \hat{d}) = 0$$

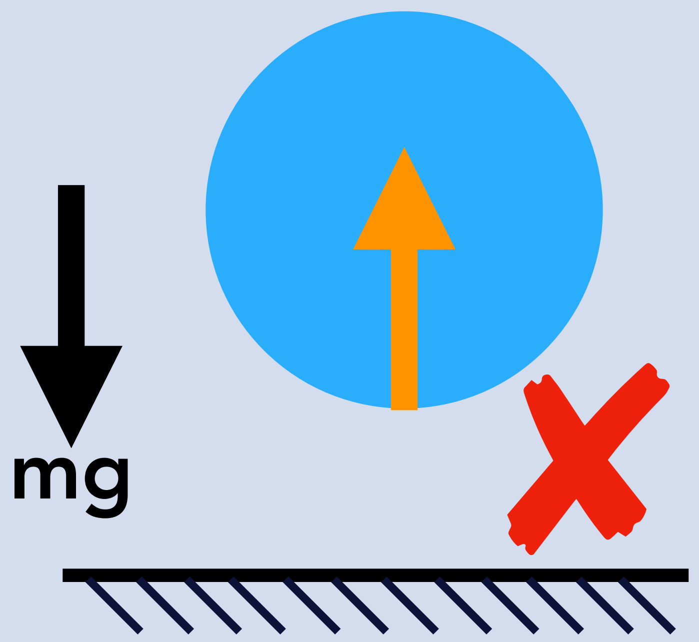
Momentum balance



Complementarity

$$\sum_k \gamma_k d_k = 0$$

Contact force only on touching regions



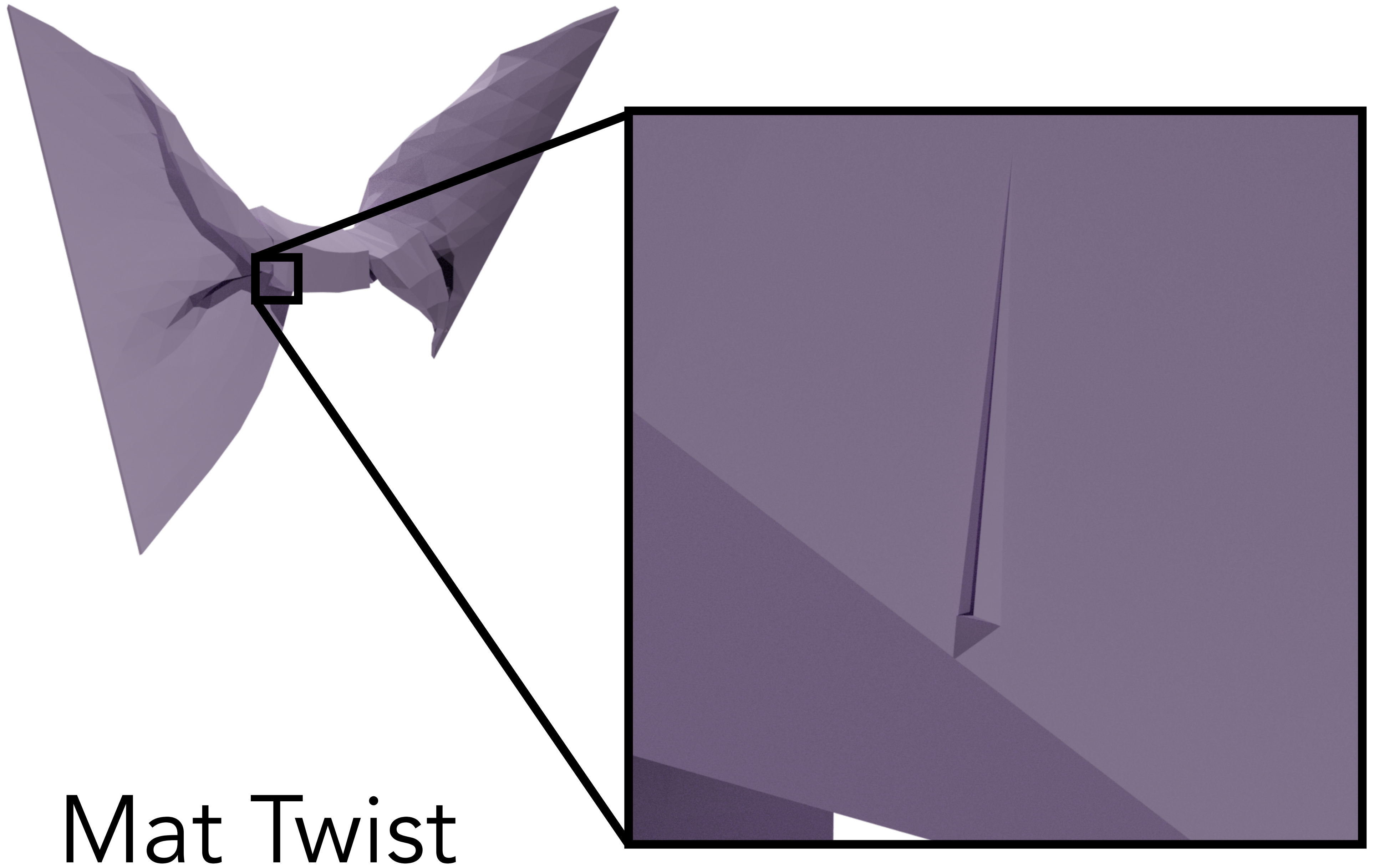
Inequality Constrained Optimization Methods

- Penalty methods
    - Convert to “unconstrained” optimization — **No guarantees of feasibility**
- Active set methods
    - Convert to equality-constrained optimization — **No guarantees of convergence**
- Primal-dual methods
    - Solving the KKT system by alternating x-update and γ-update — **Slow convergence**
- Barrier methods (Interior-point method)
    - Convert to “unconstrained” optimization — **Guarantees fast convergence and feasibility**

Inequality Constrained Optimization Methods

Sequential Quadratic Programming (SQP) for Contact Simulation

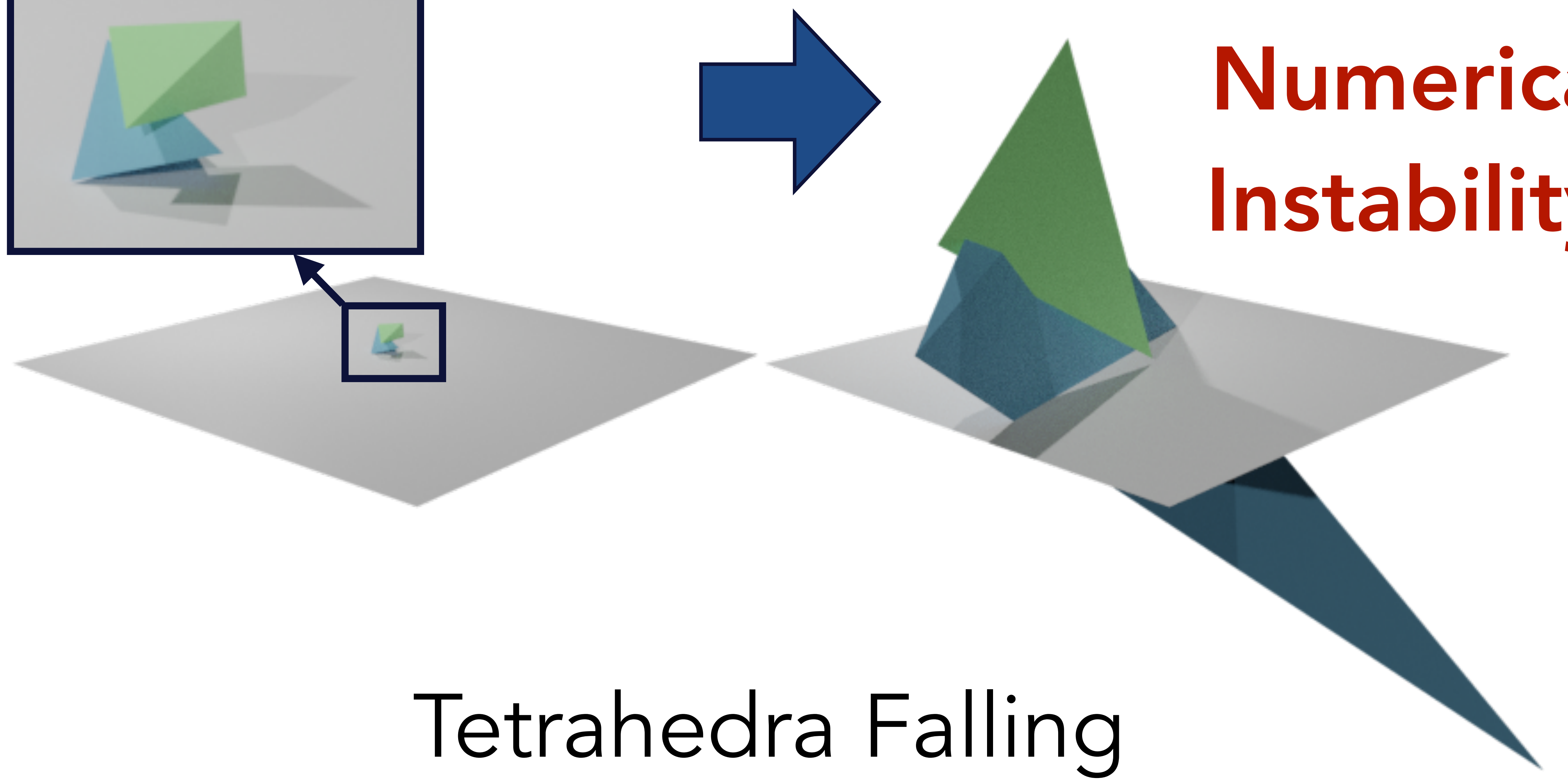
Common failures:



Mat Twist

Interpenetration!

a few frames later



Numerical Instability!

Tetrahedra Falling

Today:

- Formulation
  - *Distance Constraints & KKT System*
- Barrier Method
- Filtered Line Search
- Implementation & Demo
- Remarks

Barrier Method

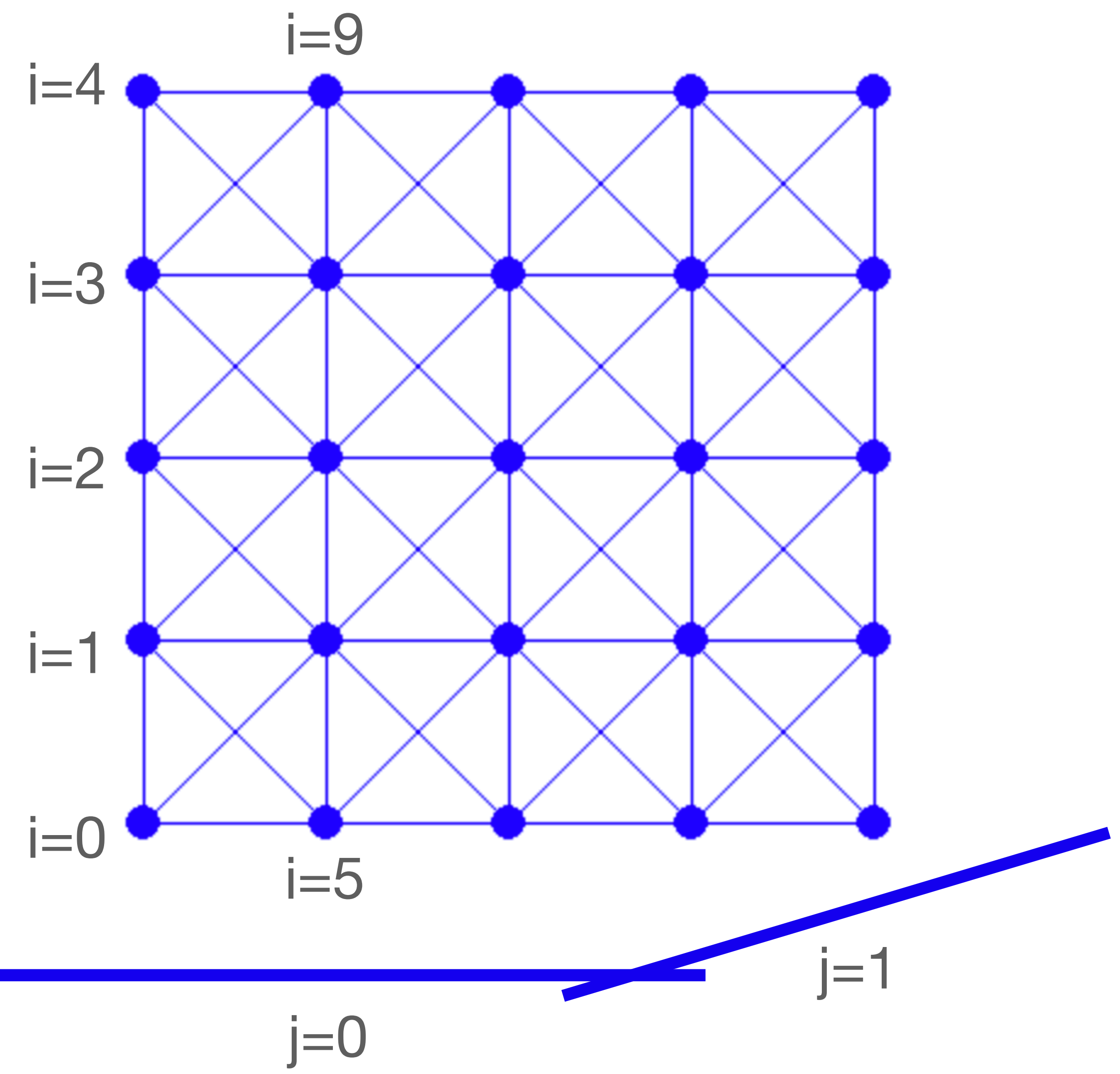
Solid-to-Obstacle Contact

$$\min_x E(x) \quad \text{s.t.} \quad d_{ij} \geq 0 \;\; \forall \text{ node } i \text{ and obstacle } j.$$

Approximate constraints with potential energy

$$\min_x E(x) + h^2 P_b(x)$$

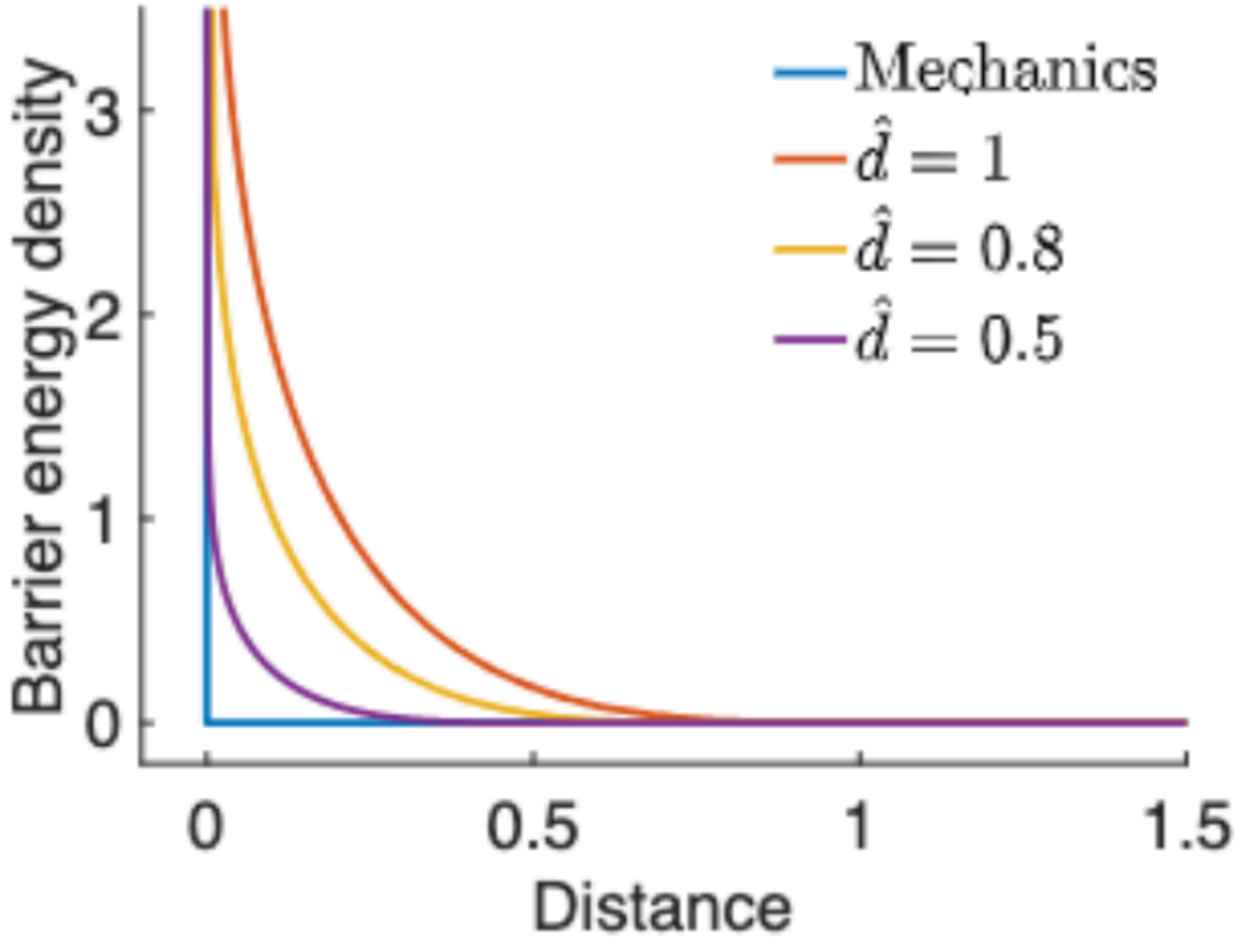
— Incremental Potential Contact (IPC) [Li et al 2020]



$$P_b(x) = \sum_{i,j} A_i \hat{d}\, b(d_{ij}(x))$$

Volume weighting

$$b(d_{ij}(x)) = \begin{cases} \frac{\kappa}{2}\left(\frac{d_{ij}}{\hat{d}} - 1\right) \ln \frac{d_{ij}}{\hat{d}} & d_{ij} < \hat{d} \\ 0 & d_{ij} \geq \hat{d}. \end{cases}$$



Feasibility guaranteed due to the barrier;

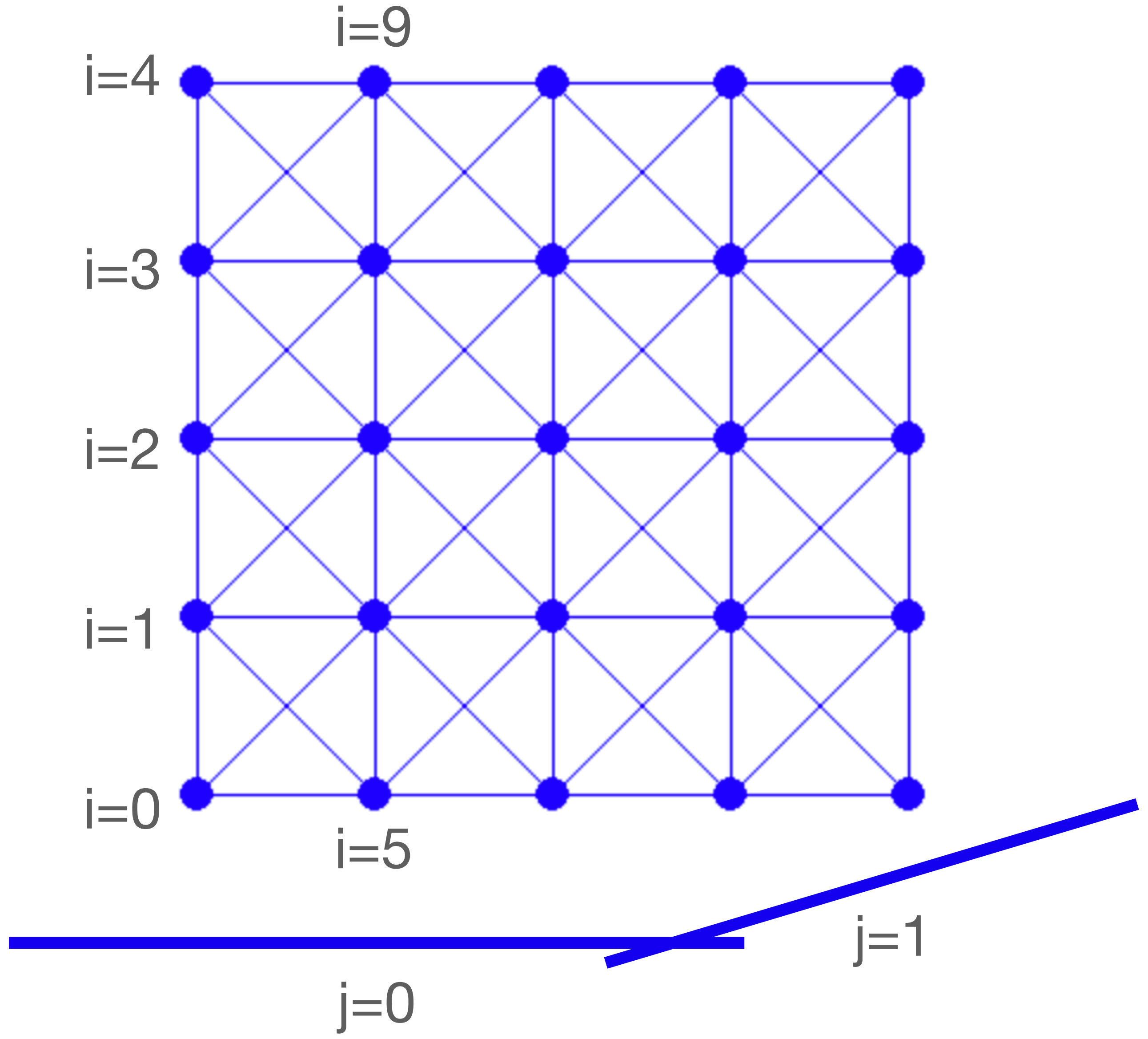
Convergence guaranteed by line search.

Barrier Method

Solid-to-Obstacle Contact — Derivatives

$$\min_x E(x) + h^2 P_b(x)$$

$$P_b(x) = \sum_{i,j} \underbrace{A_i \hat{d}}_{w_i} b(d_{ij}(x)) \qquad b(d_{ij}(x)) = \begin{cases} \frac{\kappa}{2}\left(\frac{d_{ij}}{\hat{d}} - 1\right) \ln \frac{d_{ij}}{\hat{d}} & d_{ij} < \hat{d} \\ 0 & d_{ij} \geq \hat{d} \end{cases}$$

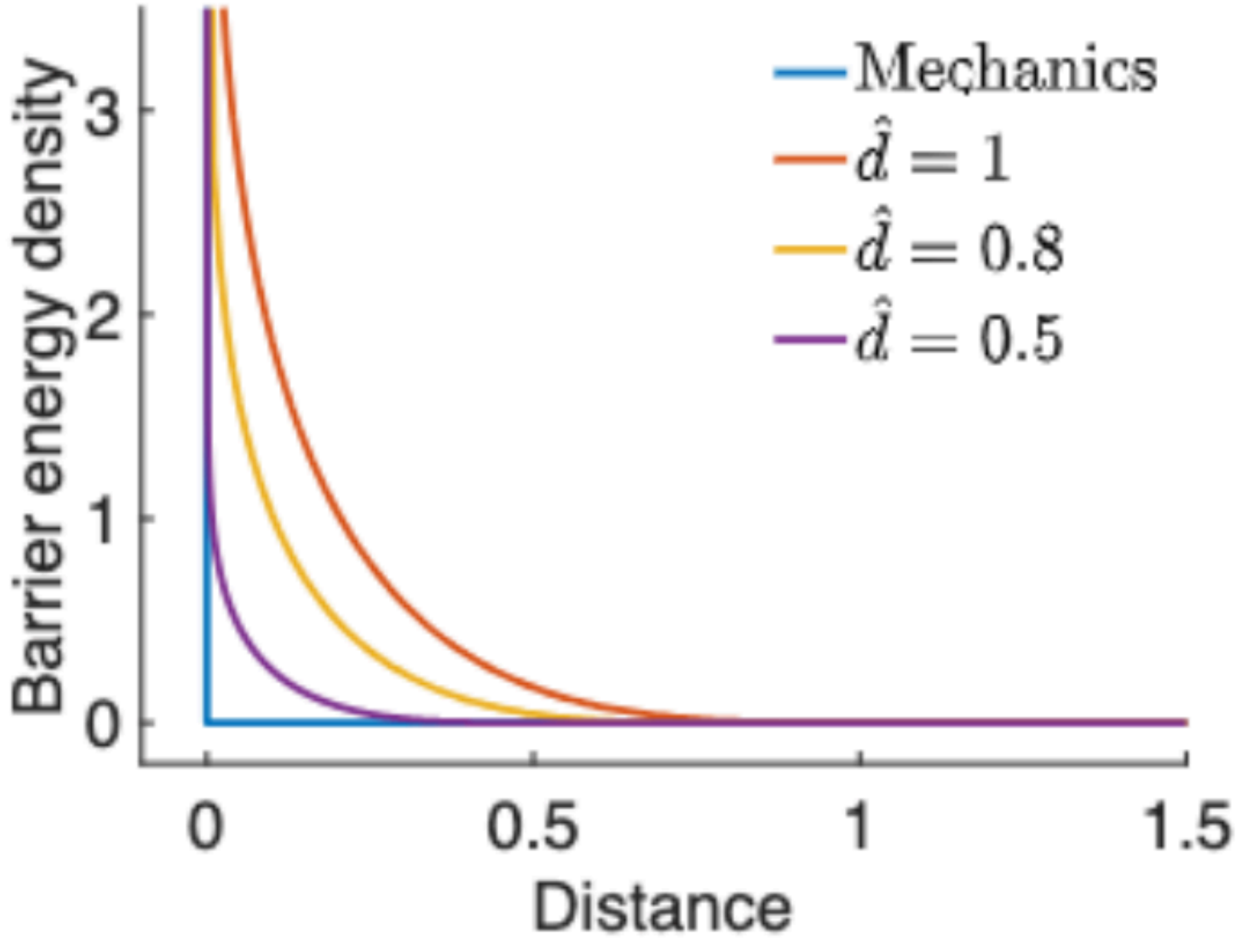


- Gradient

$$\nabla P_b(x) = \sum_{i,j} w_i \frac{\partial b}{\partial d}(d_{ij}(x)) \nabla d_{ij}(x)$$

- Hessian

$$\nabla^2 P_b(x) = \sum_{i,j} w_i \left(\frac{\partial^2 b}{\partial d^2}(d_{ij}(x)) \nabla d_{ij}(x) \nabla d_{ij}(x)^T + \frac{\partial b}{\partial d}(d_{ij}(x)) \nabla^2 d_{ij}(x) \right)$$



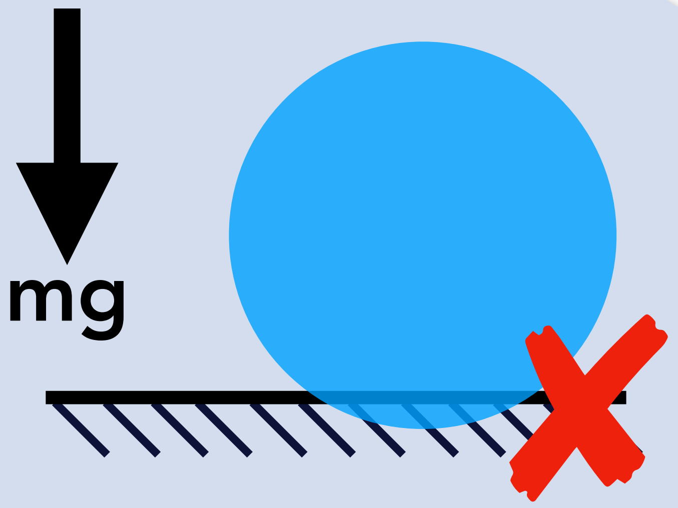
Barrier Method

Satisfaction of KKT Conditions

Primal feasibility

$$\forall k, \quad d_k \geq 0$$

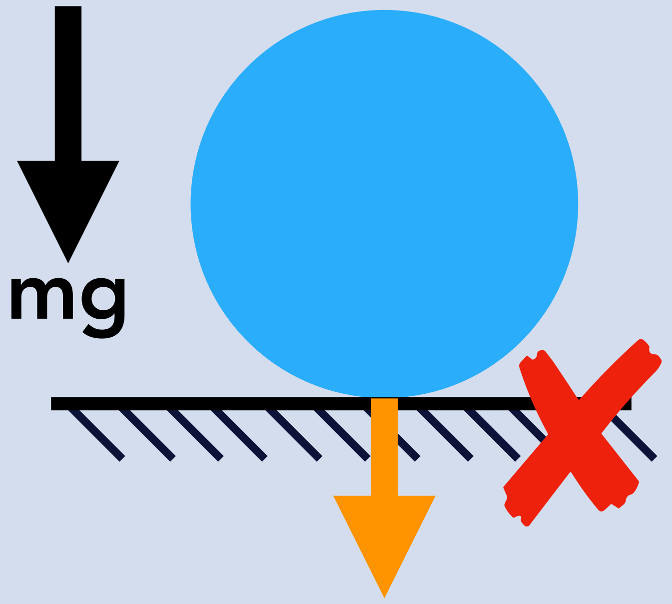
Nonpenetration



Satisfied by construction!

Dual feasibility

$$\forall k, \quad -\kappa \frac{\partial b}{\partial d_k} \geq 0$$



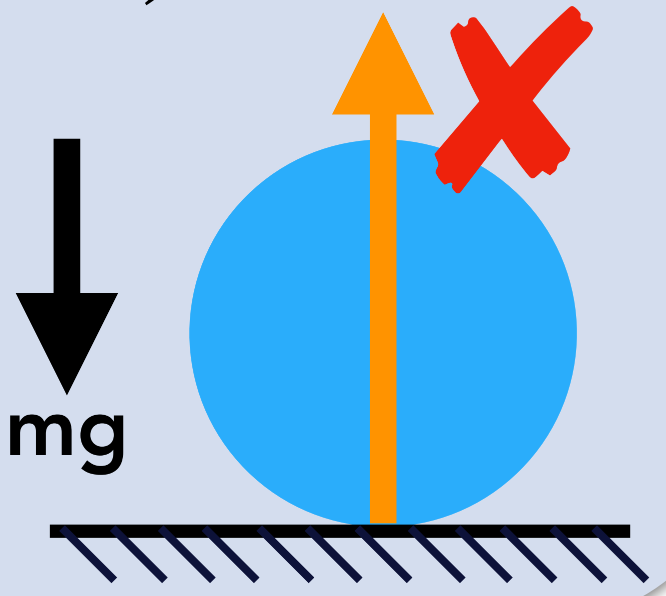
Contact force only push but not pull

Satisfied by construction!

Stationarity

$$\nabla E(x) + \kappa \sum_k \nabla b(d_k(x), \hat{d}) = 0$$

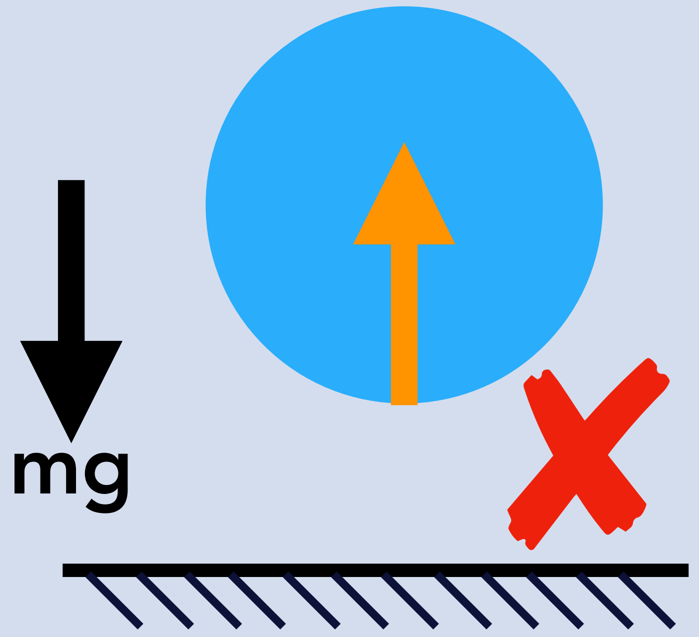
Momentum balance



Accuracy controlled by Newton tolerance ϵ

Complementarity

$$\sum_k d_k(-\kappa \frac{\partial b}{\partial d_k}) = 0$$



Contact force only on touching regions

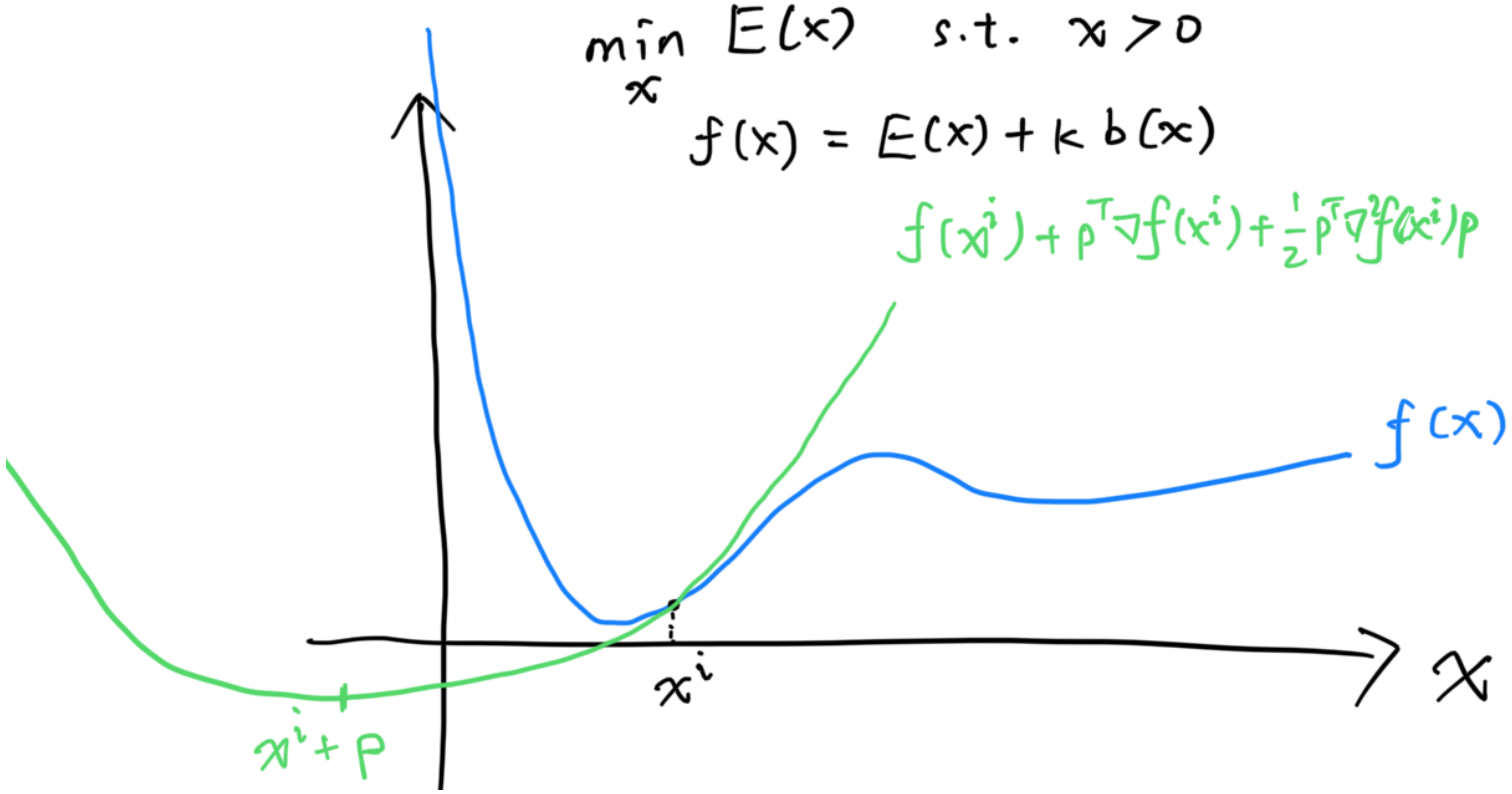
Accuracy controlled by $\hat{d}$

Today:

- Formulation
  - *Distance Constraints & KKT System*
- Barrier Method
  - *Constrained problem -> unconstrained solves*
- **Filtered Line Search**
- Implementation & Demo
- Remarks

Issues of Newton's Method on Barrier Energies

- Newton's search direction $p = -H^{-1}g$ is not fully aware of the barrier
- Needs to make sure step size α is feasible



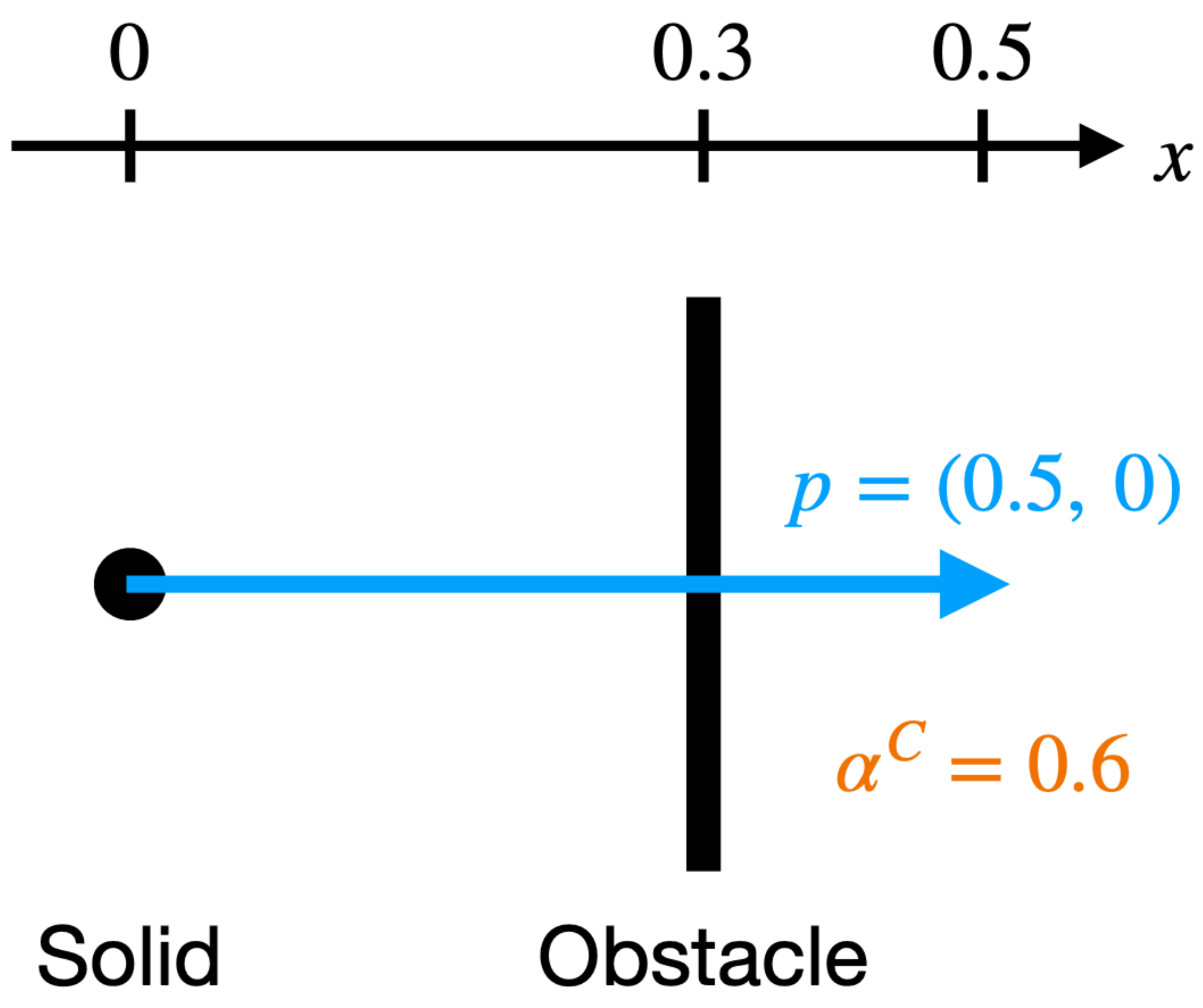
Filtered Line Search

- Line search from α rather than 1 such that

$$d_{jk}(x^i + \beta p) \geq 0 \quad \forall \text{ node } j, \text{ obstacle } k, \text{ and } \beta \in [0, \alpha]$$

- Continuous Collision Detection (CCD)
  - For a distance function $d_{jk}(x + \alpha p)$
  - CCD calculates α_{jk}^C such that $d_{jk}(x + \alpha p) > 0 \quad \forall\, \alpha \in [0, \alpha_{jk}^C)$
  - α_{jk}^C is the smallest positive real root of $d_{jk}^a(\alpha) = d_{jk}(x + \alpha p)$
  - If there is not any positive real root, then $\alpha_{jk}^C = \infty$
  - $\alpha = \min_{j,k} \alpha_{jk}^C$

CCD Example



Barrier Aware Newton's Method

Algorithm 3: Projected Newton Method for Backward Euler Time Integration

Result: x^{n+1}, v^{n+1}

```
x^i ← x^n;
do
    P ← SPDProjection(∇²E(x^i));
    p ← −P^{-1}∇E(x^i);
    α ← BackTrackingLineSearch(x^i, p);  //
    x^i ← x^i + αp;
while ‖p‖_∞/h > ε;
x^{n+1} ← x^i;
v^{n+1} ← (x^{n+1} − x^n)/Δt;
```

Algorithm 4: Filter Backtracking Line Search

Result: α

```
α ← CCD(x^i, p);  // the only different line
while E(x^i + αp) > E(x^i) do
    α ← α/2;
```

Today:

- Formulation
  - *Distance Constraints & KKT System*
- Barrier Method
  - *Constrained problem -> unconstrained solves*
- Filtered Line Search
  - *Continuous Collision Detection (CCD)*
- Implementation & Demo
- Remarks

Implementation

Solid-to-Ground Signed Distance Functions

- For horizontal plane $y = y_0$

$$d(\mathbf{x}) = \mathbf{x}_y - y_0, \quad \nabla d(\mathbf{x}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \nabla^2 d(\mathbf{x}) = \mathbf{0}$$

$$P_b(x) = \sum_{i,j} A_i \hat{d} b(d_{ij}(x)) \quad \text{and} \quad b(d_{ij}(x)) = \begin{cases} \frac{\kappa}{2}(\frac{d_{ij}}{\hat{d}} - 1) \ln \frac{d_{ij}}{\hat{d}} & d_{ij} < \hat{d} \\ 0 & d_{ij} \geq \hat{d} \end{cases}$$

$$\nabla P_b(x) = \sum_{i,j} w_i \frac{\partial b}{\partial d}(d_{ij}(x)) \nabla d_{ij}(x)$$

$$\nabla^2 P_b(x) = \sum_{i,j} w_i \left(\frac{\partial^2 b}{\partial d^2}(d_{ij}(x)) \nabla d_{ij}(x) \nabla d_{ij}(x)^T + \frac{\partial b}{\partial d}(d_{ij}(x)) \nabla^2 d_{ij}(x) \right)$$

BarrierEnergy.py

```python
import math
import numpy as np

dhat = 0.01
kappa = 1e5

def val(x, y_ground, contact_area):
    sum = 0.0
    for i in range(0, len(x)):
        d = x[i][1] - y_ground
        if d < dhat:
            s = d / dhat
            sum += contact_area[i] * dhat * kappa / 2 * (s - 1) * math.log(s)
    return sum

def grad(x, y_ground, contact_area):
    g = np.array([[0.0, 0.0]] * len(x))
    for i in range(0, len(x)):
        d = x[i][1] - y_ground
        if d < dhat:
            s = d / dhat
            g[i][1] = contact_area[i] * dhat * (kappa / 2 * (math.log(s) / dhat + (s - 1) / d))
    return g

def hess(x, y_ground, contact_area):
    IJV = [[0] * len(x), [0] * len(x), np.array([0.0] * len(x))]
    for i in range(0, len(x)):
        IJV[0][i] = i * 2 + 1
        IJV[1][i] = i * 2 + 1
        d = x[i][1] - y_ground
        if d < dhat:
            IJV[2][i] = contact_area[i] * dhat * kappa / (2 * d * d * dhat) * (d + dhat)
        else:
            IJV[2][i] = 0.0
    return IJV
```

Implementation

Solid-to-Ground CCD

- For horizontal plane $y = y_0$

$$d(\mathbf{x}) = \mathbf{x}_y - y_0 \qquad d(\mathbf{x} + \alpha \boldsymbol{p}) = \mathbf{x}_y + \alpha \boldsymbol{p}_y - y_0$$

$d^a_{jk}(\alpha) = d_{jk}(x + \alpha p)$ **has only one positive real root** $(y_0 - \mathbf{x}_y)/\boldsymbol{p}_y$ when $\boldsymbol{p}_y < 0$ since $\mathbf{x}_y > y_0$

BarrierEnergy.py

```
def init_step_size(x, y_ground, p):
    alpha = 1
    for i in range(0, len(x)):
        if p[i][1] < 0:
            alpha = min(alpha, 0.9 * (y_ground - x[i][1]) / p[
    i][1])
    return alpha
```

To avoid exact 0 distance

Demo!

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

GPU Version: github.com/phys-sim-book/solid-sim-tutorial-gpu

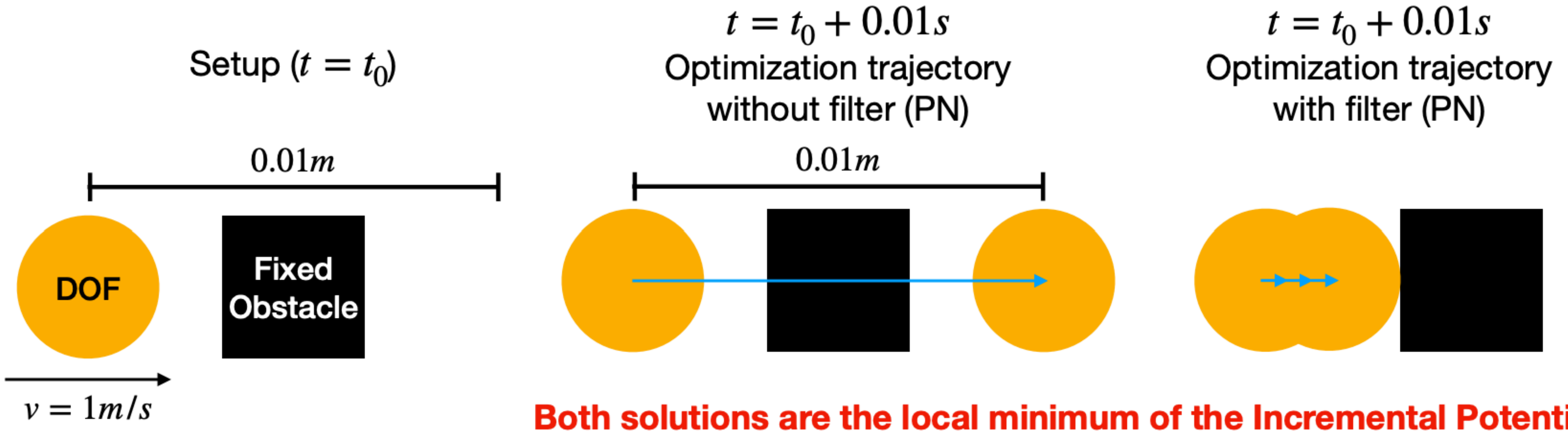
Online Book: phys-sim-book.github.io

Today:

- Formulation
  - *Distance Constraints & KKT System*
- Barrier Method
  - *Constrained problem -> unconstrained solves*
- Filtered Line Search
  - *Continuous Collision Detection (CCD)*
- Implementation & Demo
- Remarks

Remarks

Local Minimums of Incremental Potential



Both solutions are the local minimum of the Incremental Potential!

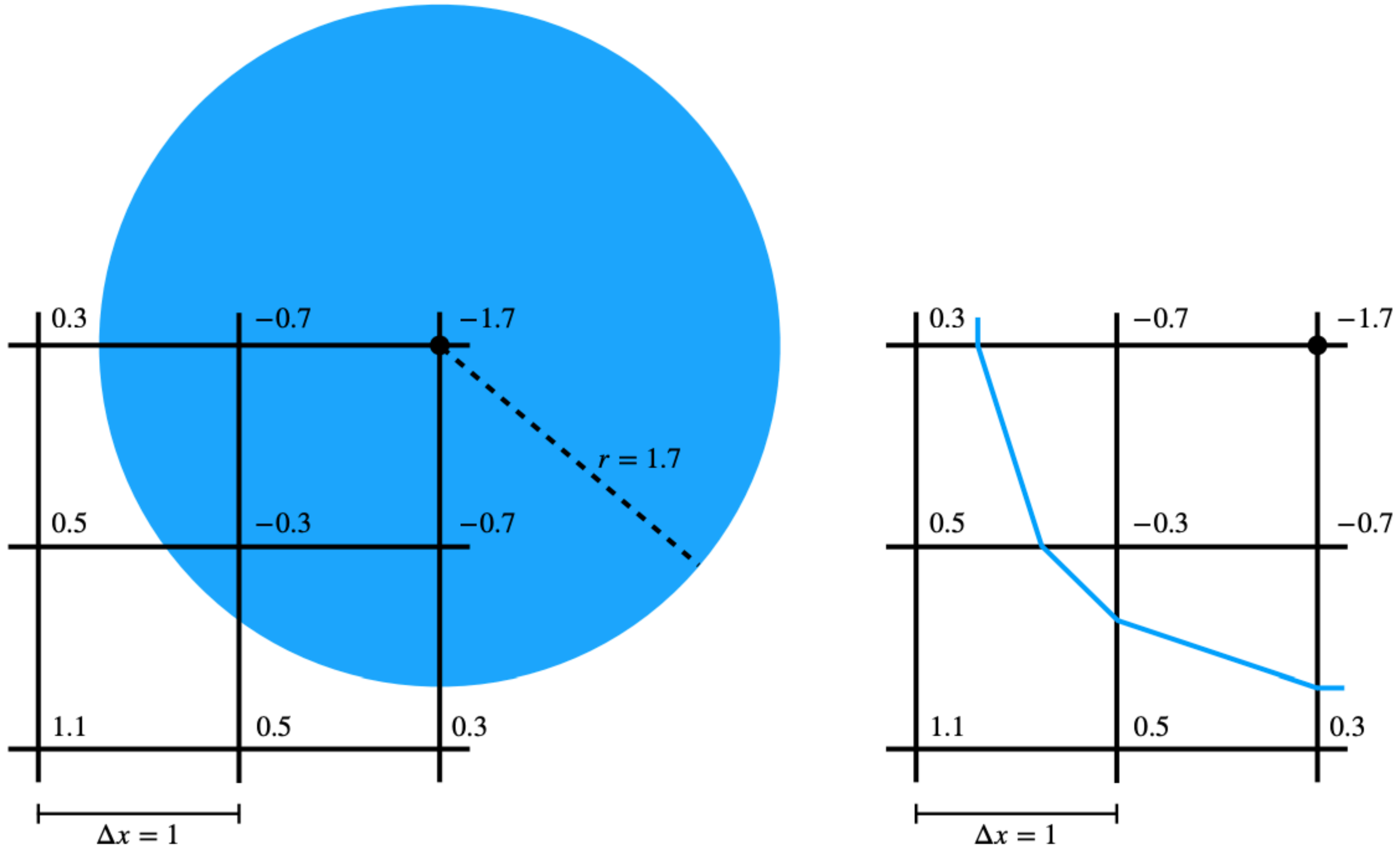
We want to locally minimize (find nearby optimum) the IP for physical simulation

Remarks

Grid-Based Signed Distance Field

- For $\mathbf{x} = (x, y)$ where $x = x_i + \alpha\Delta x$ and $y = y_i + \beta\Delta x$
- Linear interpolation gives

$$d(\mathbf{x}) = (1-\beta)((1-\alpha)d(\mathbf{x}_{i,i})+\alpha d(\mathbf{x}_{i+1,i}))+\beta((1-\alpha)d(\mathbf{x}_{i,i+1})+\alpha d(\mathbf{x}_{i+1,i+1}))$$



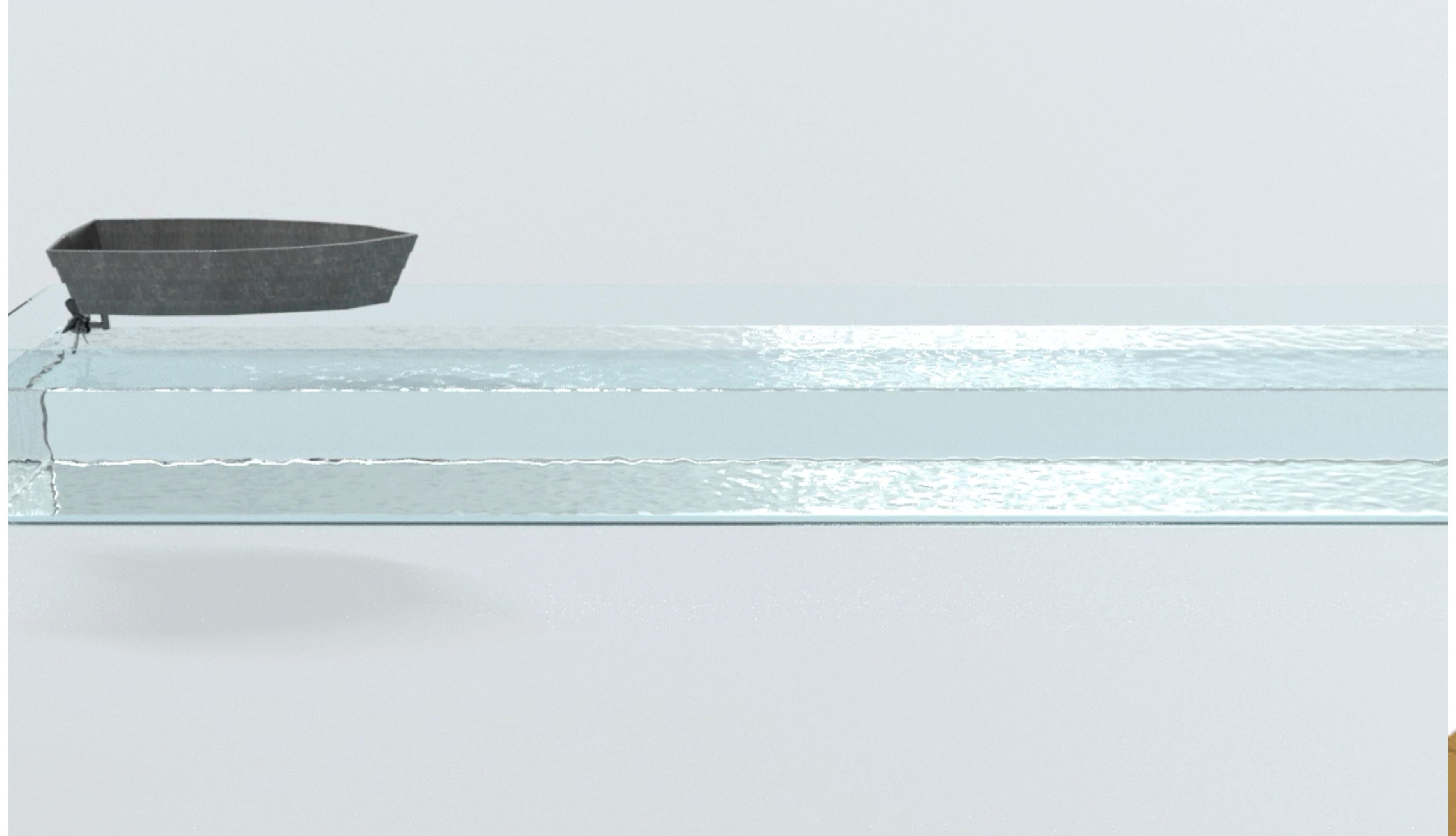
Not smooth when crossing cells!

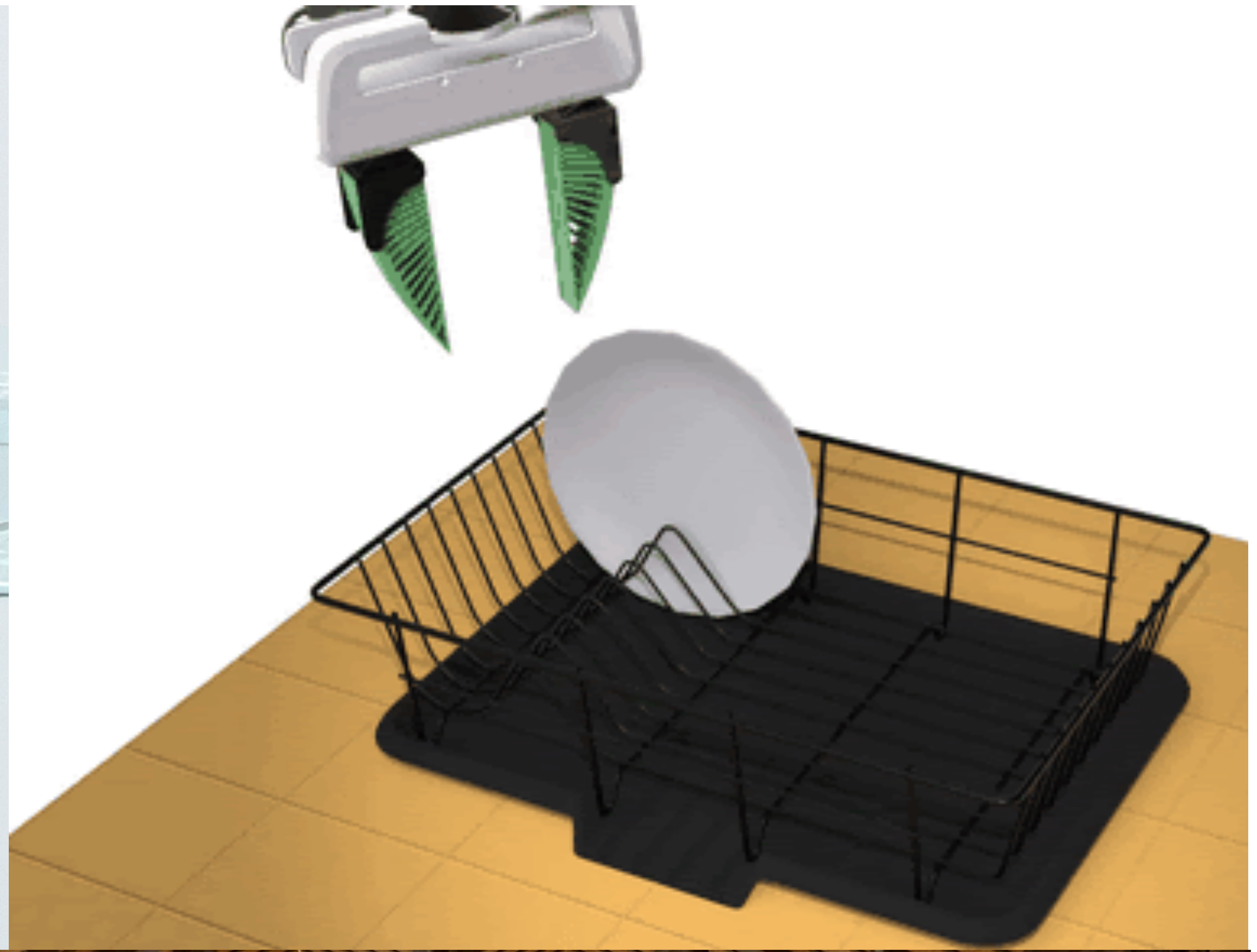
Solution:
- **Use high-order B-Splines, or**
- **Keep using the same interpolant**

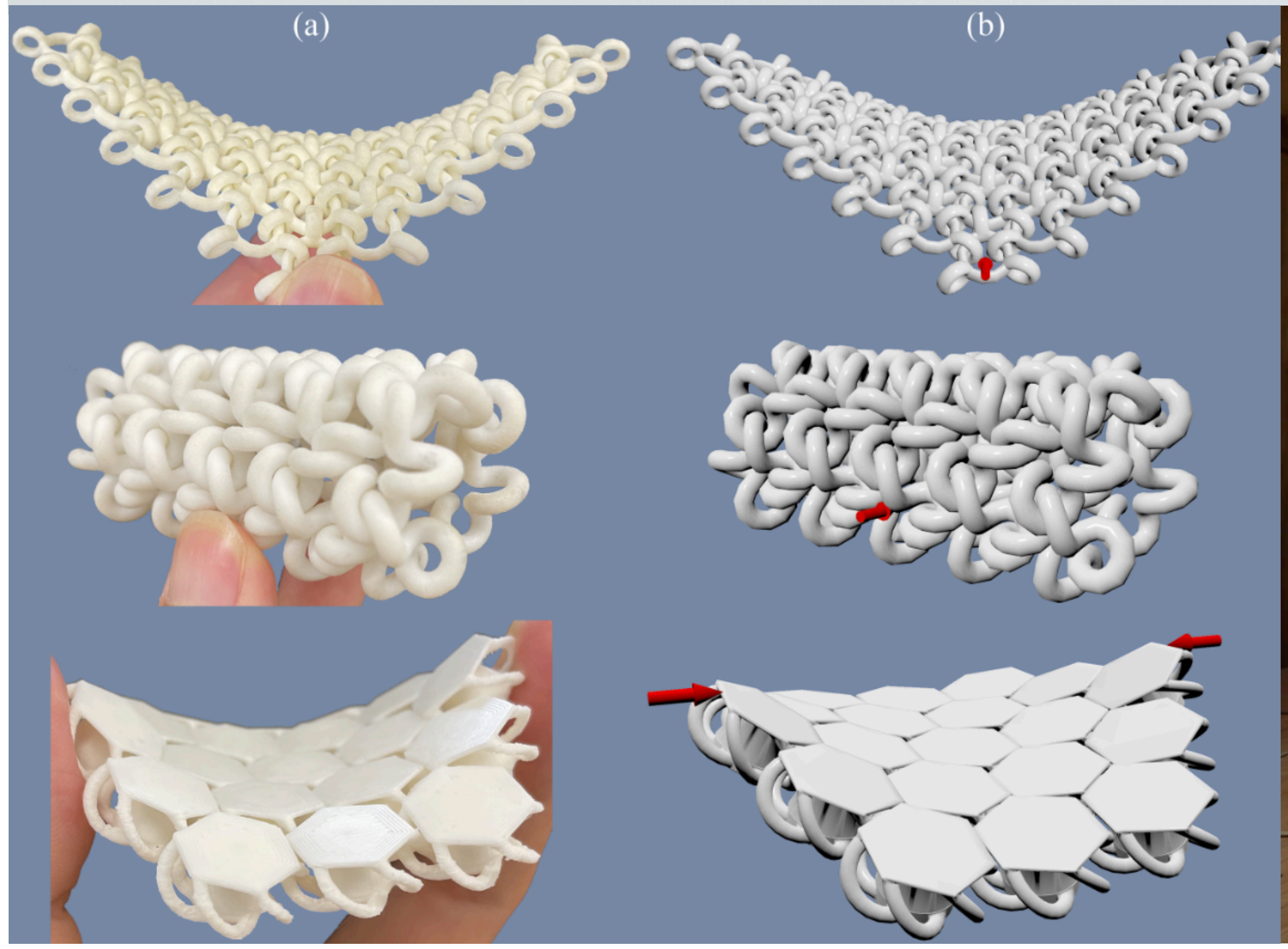
Remarks

Applications

- Robot learning [Kim et al 2022] [Du et al 2024]
- Cloth reconstruction [Zheng et al 2024]
- Material modeling [Tang et al 2023]
- Multi-material coupled simulations [Jiang et al 2022] [Xie et al 2023] [Li et al 2024]







Today:

- Formulation
  - *Distance Constraints & KKT System*
- Barrier Method
  - *Constrained problem -> unconstrained solves*
- Filtered Line Search
  - *Continuous Collision Detection (CCD)*
- Implementation & Demo
- Remarks
  - *Local Minimum, SDF, Applications*

Next Lecture: Friction

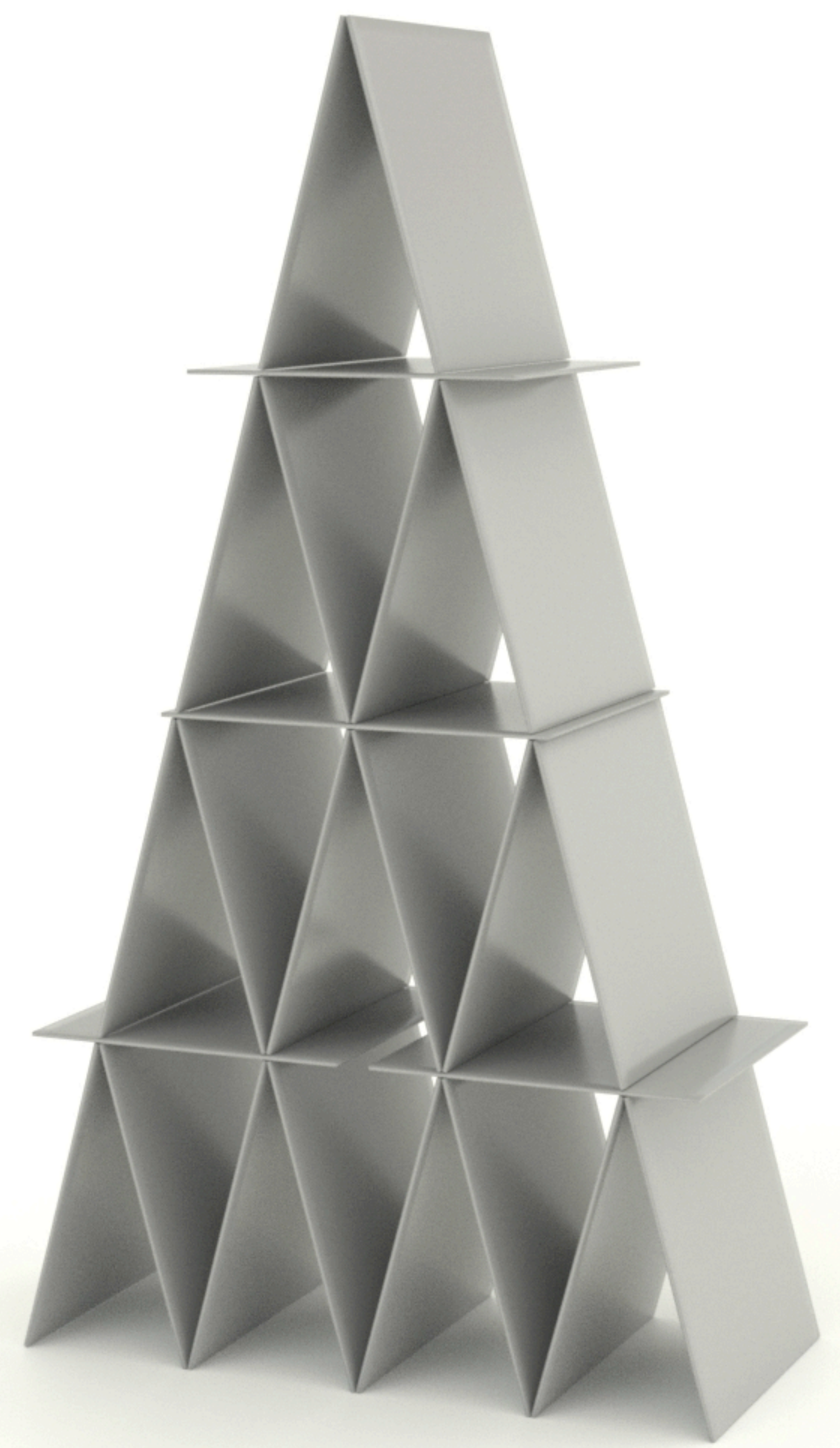


Image Sources

- https://www.nvidia.com/en-us/geforce/graphics-cards/50-series/rtx-5090/
- https://tangpengbin.github.io/publications/DIM/DIM.html
- https://sites.google.com/view/embedded-ipc
- https://qingqing-zhao.github.io/PhysAvatar