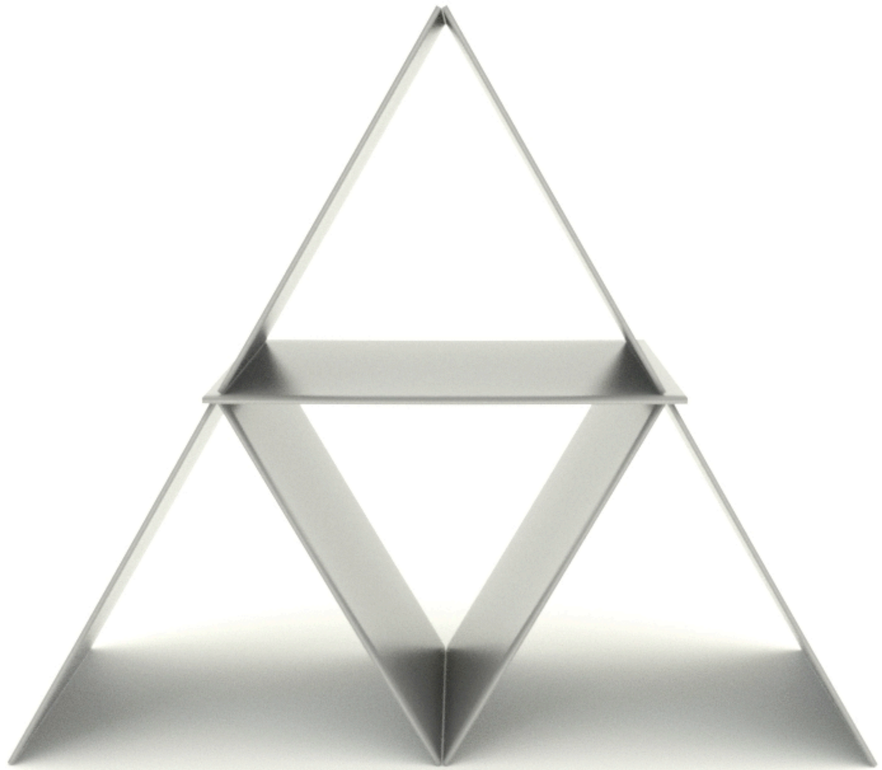
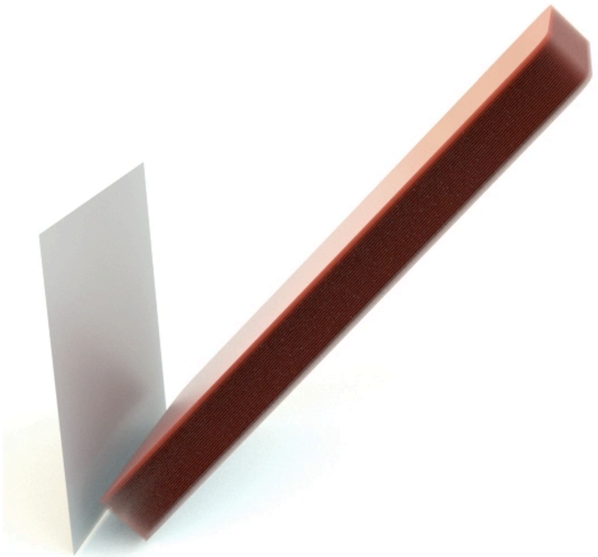
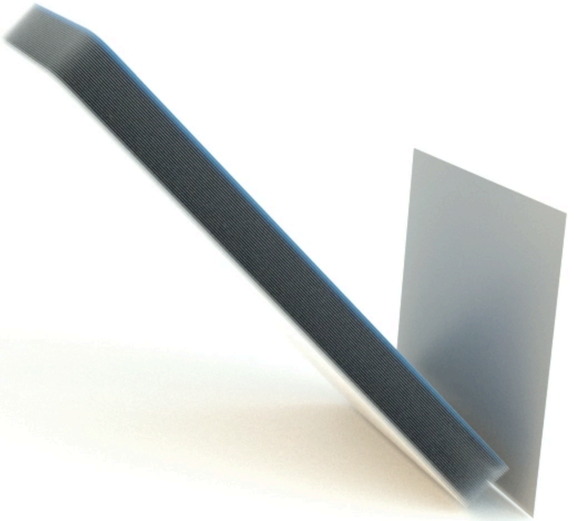
Instructor: Minchen Li







Lec 5: Friction

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

Recap: Normal Contact Force

- Inequality constraints

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

Elasticity

Inertia term

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

- Barrier method: $\min_x E(x) + h^2 P_b(x)$
- Filtered line search

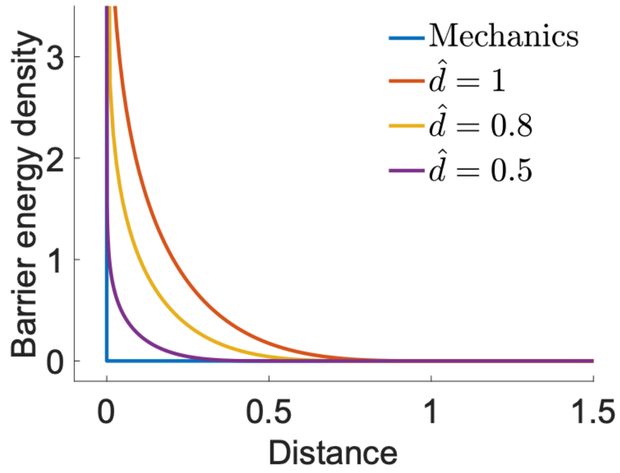
Algorithm 4: Filter Backtracking Line Search

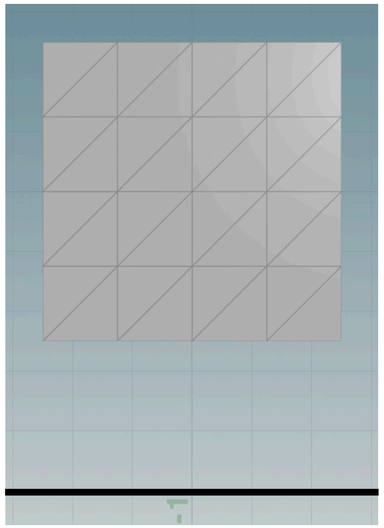
Result: α

$\alpha \leftarrow \text{CCD}(x^i, p)$; // the only different line from algorithm 2

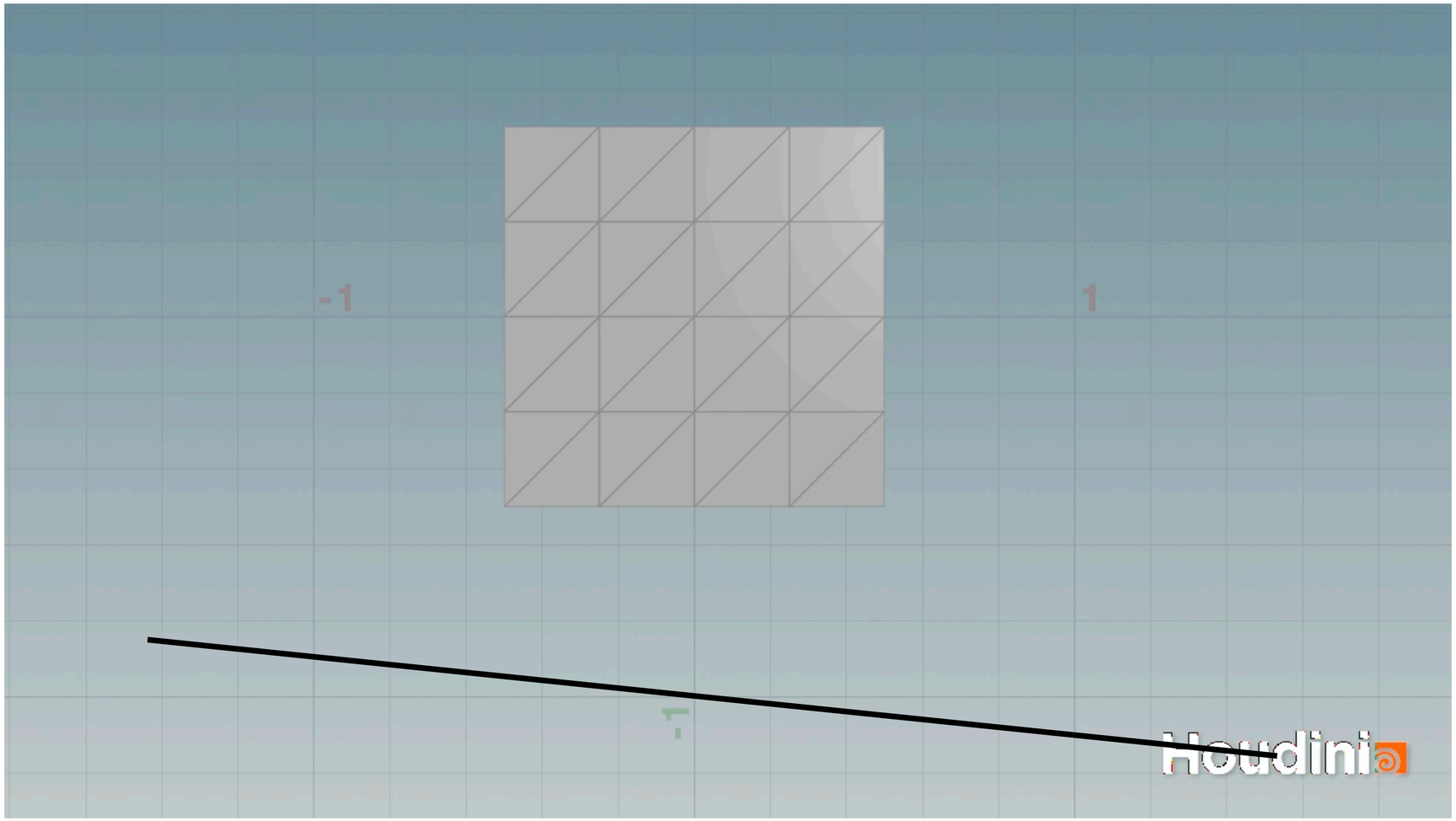
while $E(x^i + \alpha p) > E(x^i)$ **do**

  $\alpha \leftarrow \alpha/2$;





Simulating Tangential Contact Force — Friction



Today:

- Formulation
- Case Study: Square on Slope
- Remarks

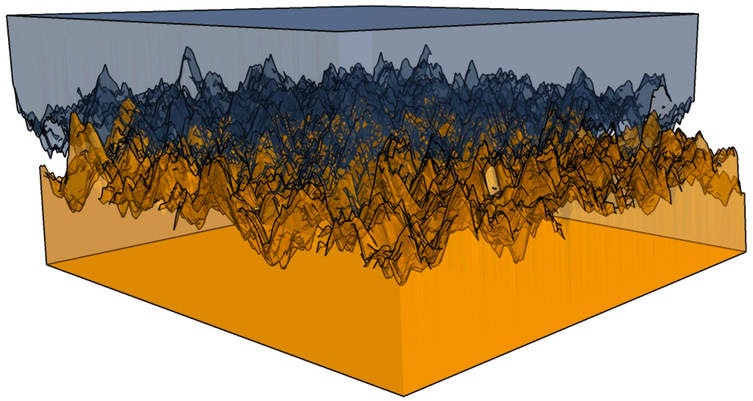
Today:

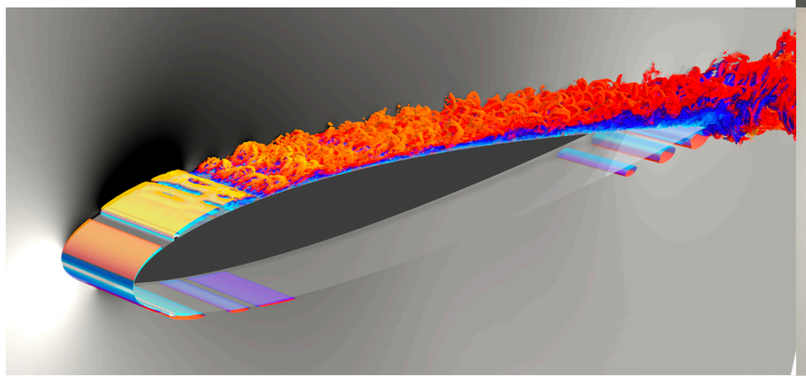
- Formulation
- Case Study: Square on Slope
- Remarks

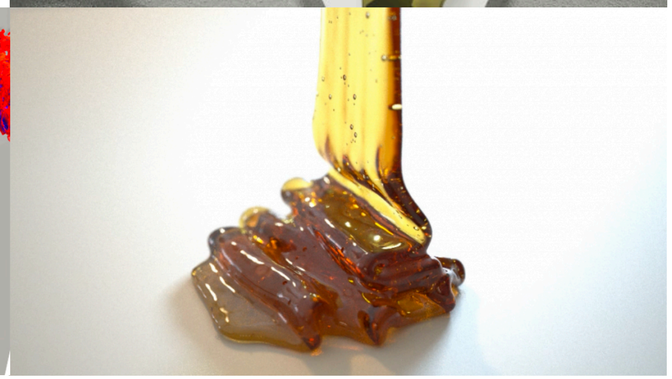
Types of Friction (Macroscopic Views)

- Dry friction
  - Between rough surfaces
- Fluid friction
  - i.e. fluid viscosity
- Lubricated friction
- Skin friction
- …

Our focus today







Properties of Dry Friction

- Non-conservative
  - Work done by friction is **path dependent**
  - Kinetic energy is always transformed to thermal energy (not backwards)
- No potential energy
  - how to incorporate it into the optimization time integration?

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

Formulation of Dry Friction

Tangential Relative Velocity

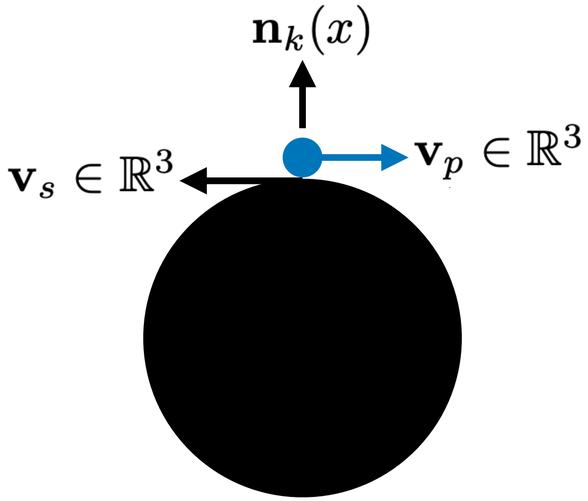
- Add friction to each contact pair k

  Tangential relative velocity $\mathbf{v}_k = T_k(x)^T v \in \mathbb{R}^d$

  Tangent operator $T_k(x) \in \mathbb{R}^{dm \times d}$

  Current configuration $\{x, v\}$

- Example:

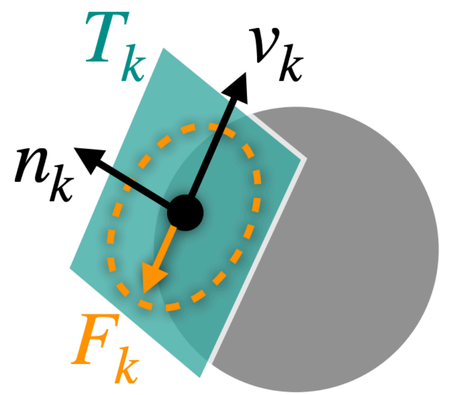


$$\mathbf{v}_k = (\mathbf{v}_p - \mathbf{v}_s) - \mathbf{n}_k \cdot (\mathbf{v}_p - \mathbf{v}_s)\mathbf{n}_k = (\mathbf{I}_3 - \mathbf{n}_k\mathbf{n}_k^T)(\mathbf{v}_p - \mathbf{v}_s)$$

Denote $v = [\mathbf{v}_p^T, \mathbf{v}_s^T]^T \in \mathbb{R}^6$ $\quad T_k(x) = \begin{bmatrix} \mathbf{I}_3 - \mathbf{n}_k(x)\mathbf{n}_k(x)^T \\ \mathbf{n}_k(x)\mathbf{n}_k(x)^T - \mathbf{I}_3 \end{bmatrix} \in \mathbb{R}^{6\times 3}$

Formulation of Dry Friction

Coulomb Friction



- Add friction to each contact pair k

Coefficient of friction

$$F_k(x) = T_k(x) \operatorname*{argmin}_{\boldsymbol{\beta} \in \mathbb{R}^d} \boldsymbol{\beta}^T \mathbf{v}_k \quad \text{s.t.} \quad \|\boldsymbol{\beta}\| \leq \mu \lambda_k \quad \text{and} \quad \boldsymbol{\beta} \cdot \mathbf{n}_k = 0$$

$$\lambda_k = -w_k \frac{\partial b}{\partial d_k} \text{ is the contact force magnitude}$$

$$\text{contact force is } -w_k \nabla_{\mathbf{x}_k} b(d_k(x)) = -w_k \frac{\partial b}{\partial d_k} \nabla_{\mathbf{x}_k} d_k(x)$$

$$\lambda_k = \| - w_k \frac{\partial b}{\partial d_k} \nabla_{\mathbf{x}_k} d_k(x) \| = -w_k \frac{\partial b}{\partial d_k} \text{ since } \frac{\partial b}{\partial d_k} < 0 \text{ and } \|\nabla_{\mathbf{x}_k} d_k(x)\| = 1$$

$$F_k(x) = -\mu \lambda_k T_k(x) f(\|\mathbf{v}_k\|) \mathbf{s}(\mathbf{v}_k)$$

Non-smooth!

$$\mathbf{s}(\mathbf{v}_k) = \frac{\mathbf{v}_k}{\|\mathbf{v}_k\|} \text{ when } \|\mathbf{v}_k\| > 0$$

Any unit vector $\perp n_k$ when $\|\mathbf{v}_k\| = 0$

$$f(\|\mathbf{v}_k\|) = 1 \text{ when } \|\mathbf{v}_k\| > 0$$

$$f(\|\mathbf{v}_k\|) \in [0, 1] \text{ when } \|\mathbf{v}_k\| = 0$$

Formulation of Dry Friction

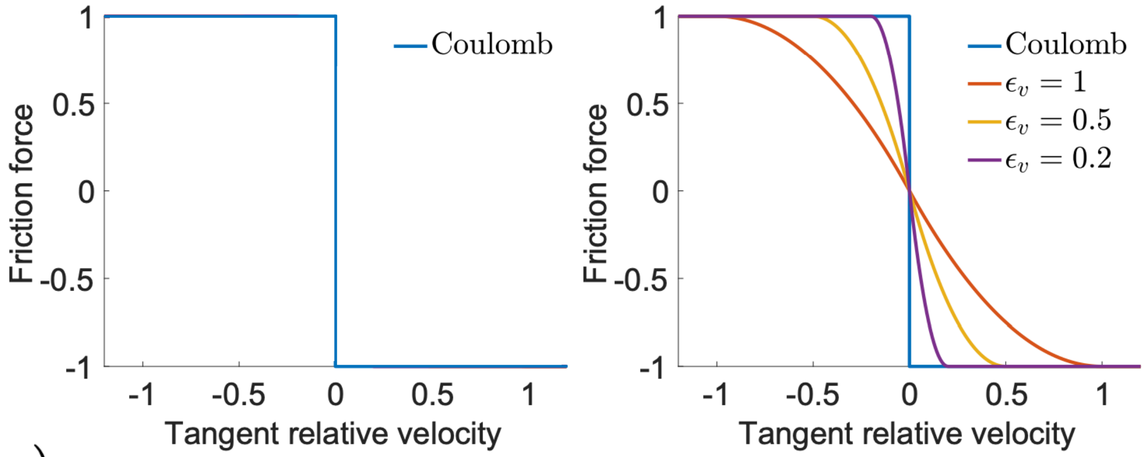
Approximation: Smooth Dynamic-Static Transition

$$F_k(x) = -\mu\lambda_k T_k(x) f(\|\mathbf{v}_k\|)\mathbf{s}(\mathbf{v}_k)$$

Non-smooth!

$$f(\|\mathbf{v}_k\|) = 1 \text{ when } \|\mathbf{v}_k\| > 0$$

$$f(\|\mathbf{v}_k\|) \in [0,1] \text{ when } \|\mathbf{v}_k\| = 0$$



Approximate f with $f_1(y) = \begin{cases} -\frac{y^2}{\epsilon_v^2} + \frac{2y}{\epsilon_v}, & y \in [0, \epsilon_v) \\ 1, & y \geq \epsilon_v, \end{cases}$

What about s ?

Let $f(x)$ be a function we wish to smooth. It is C^1 continuous everywhere except at $x = a$ where it is only C^0 continuous. Applying a function $g(x)$ that is C^1 continuous everywhere with $g(a) = 0$, we have a smoothed function $f(x)g(x)$ that is C^1 continuous everywhere.

Mollification Lemma

$$\Longrightarrow \quad F_k(x) = -\mu\lambda_k T_k(x) f_1(\|\mathbf{v}_k\|)\mathbf{s}(\mathbf{v}_k)$$

is C^1-continuous!

Formulation of Dry Friction

Approximation: Semi-Implicit Temporal Discretization

$F_k(x) = -\mu\lambda_k T_k(x) f_1(\|\mathbf{v}_k\|)\mathbf{s}(\mathbf{v}_k)$ **is smooth, but, still non-conservative…**

Motivation: in explicit time integration, forces are constant (within a time step), thus integrable

Solution: discretize λ and T to time step n $\quad F_k(x) \approx -\mu\lambda_k^n T_k^n f_1(\|\bar{\mathbf{v}}_k\|)\mathbf{s}(\bar{\mathbf{v}}_k)$

where $\bar{\mathbf{v}}_k = (T_k^n)^T v$ $\quad T^n = T(x^n)$ $\quad \lambda^n = \lambda(x^n)$

$$\boxed{P_f(x) = \sum_k \mu\lambda_k^n f_0(\|\bar{\mathbf{v}}_k\hat{h}\|)}$$

$\hat{h}$ is a constant multiple of the time step size

$$\boxed{-\nabla P_f(x) = -\sum_k \mu\lambda_k^n T_k^n f_1(\|\bar{\mathbf{v}}_k\|)\mathbf{s}(\bar{\mathbf{v}}_k)}$$

$$f_0(y) = \begin{cases} -\frac{y^3}{3\epsilon_v^2\hat{h}^2} + \frac{y^2}{\epsilon_v\hat{h}} + \frac{\epsilon_v\hat{h}}{3}, & y \in [0, \epsilon_v\hat{h}) \\ y, & y \geq \epsilon_v\hat{h}, \end{cases} \qquad f_0'(y) = f_1(y/\hat{h}) \qquad \hat{h}I = (\partial v/\partial x)^{-1}$$

$$\boxed{\begin{aligned} &\nabla^2 P_f(x) \\ &= \sum_k \mu\lambda_k^n T_k^n \left(\frac{f_1'(\|\bar{\mathbf{v}}_k\|)\|\bar{\mathbf{v}}_k\| - f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|^3} \bar{\mathbf{v}}_k\bar{\mathbf{v}}_k^T + \frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|}\mathbf{I}_3 \right) T_k^{nT} \frac{\partial v}{\partial x} \end{aligned}}$$

Formulation of Dry Friction

Fixed-Point Iteration

- How to obtain solution with **fully-implicit friction**?

Algorithm: alternate between

$$\min_x : E(x, \{\lambda, T\}) = \frac{1}{2}\|x - \tilde{x}^n\|_M^2 + \Delta t^2 \left(P_e(x) + P_b(x) + P_f(x, \{\lambda, T\})\right)$$

$$\text{s.t.} \quad Ax = b,$$

$$\text{and friction update until convergence}$$

Fixed-point iteration on $f_m \cdot f_u$

Define

$$f_m(\{\lambda, T\}) = \operatorname*{argmin}_x E(x, \{\lambda, T\})$$

$$f_u(x) = \text{FrictionUpdate}(x),$$

$$(f_m \cdot f_u)(x) \equiv f_m(f_u(x))$$

Formulation of Dry Friction

Fixed-Point Iteration (Cont.)

Definition (Fixed-Point Iterations)**.** x is a fixed point of function $f()$ if and only if

$$x = f(x).$$

The fixed-point iterations finds the fixed-point of a function $f()$ starting from x^0 by iteratively updating the estimate

$$x^{i+1} \leftarrow f(x^i)$$

until convergence.

[Li et al 2020]

Algorithm 5: Fixed-Point Iteration for Fully-Implicit Friction

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

$x^j \leftarrow x^n$;
$\{\lambda^j, T^j\} \leftarrow \text{FrictionUpdate}(x^j)$;
while $\|((\nabla^2 E)^{-1} \nabla E)(x^j, \{\lambda^j, T^j\})\| > \epsilon$ **do**
  $x^{j+1} \leftarrow \text{argmin}_x \, E(x, \{\lambda^j, T^j\})$;
  $\{\lambda^{j+1}, T^{j+1}\} \leftarrow \text{FrictionUpdate}(x^{j+1})$;
  $j \leftarrow j + 1$

Converges only when starting sufficiently close.

No numerical explosion issues here.

Today:

- Formulation

  Mollification, Semi-Implicit Approx., Fixed-Point Iteration

- **Case Study: Square on Slope**
- Remarks

Case Study: Square on Slope

From Ground to Slope

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

General half-space (including slope):

$$d(\mathbf{x}) = \mathbf{n}^T(\mathbf{x} - \boldsymbol{o}), \quad \nabla d(\mathbf{x}) = \mathbf{n}, \quad \text{and} \quad \nabla^2 d(\mathbf{x}) = \mathbf{0}$$

simulator.py

```
17 ground_n = np.array([0.1, 1.0])      # normal of the slope
18 ground_n /= np.linalg.norm(ground_n)    # normalize ground
       normal vector just in case
19 ground_o = np.array([0.0, -1.0])     # a point on the slope
```

```
58     pygame.draw.aaline(screen, (0, 0, 255), screen_projection
       ([ground_o[0] - 3.0 * ground_n[1], ground_o[1] + 3.0 *
       ground_n[0]]),
59         screen_projection([ground_o[0] + 3.0 * ground_n[1],
       ground_o[1] - 3.0 * ground_n[0]]))   # slope
```

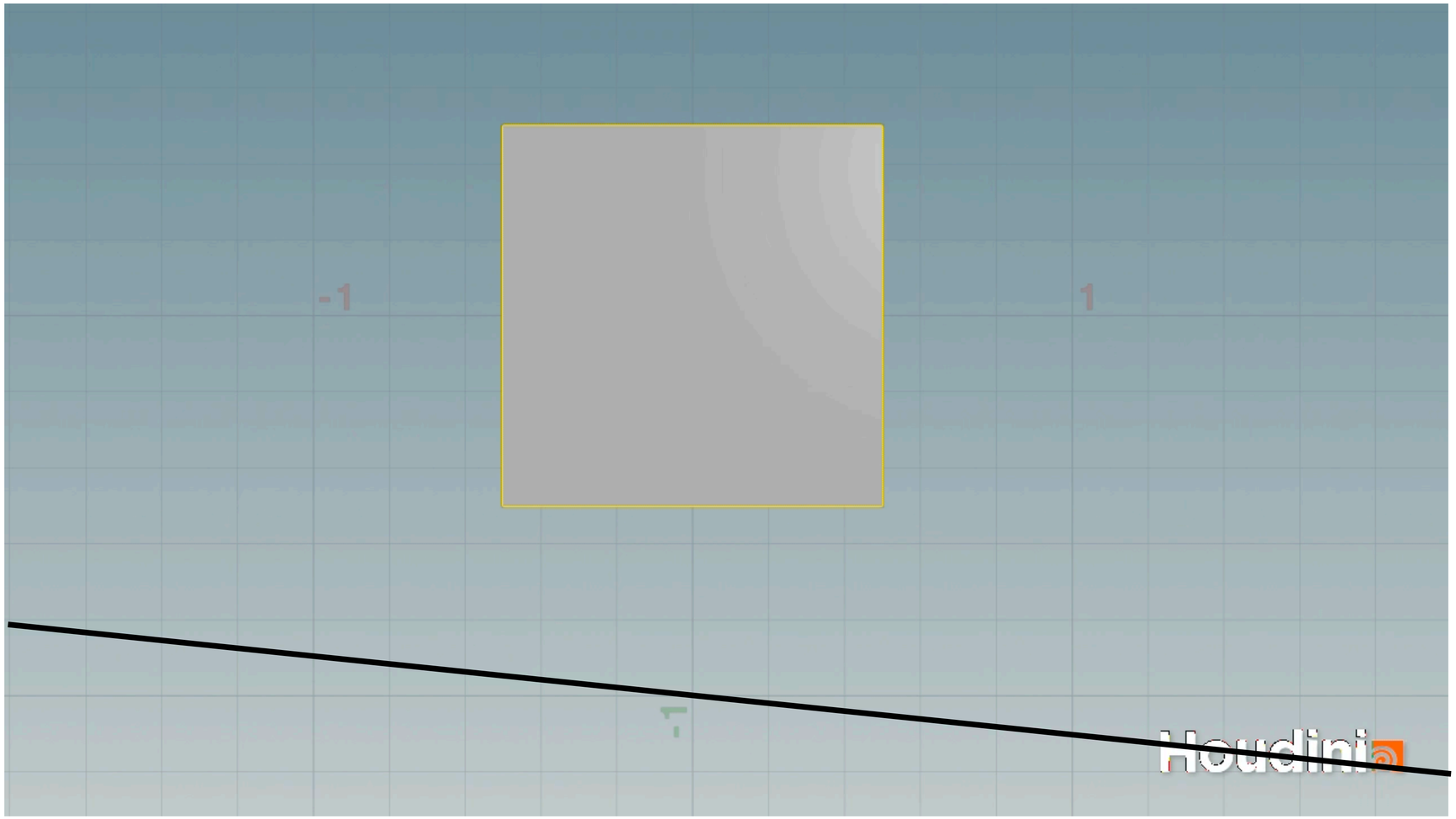
BarrierEnergy.py

```
7  def val(x, n, o, contact_area):
8      sum = 0.0
9      for i in range(0, len(x)):
10         d = n.dot(x[i] - o)
11         if d < dhat:
12             s = d / dhat
13             sum += contact_area[i] * dhat * kappa / 2 * (s -
       1) * math.log(s)
14     return sum
15
16 def grad(x, n, o, contact_area):
17     g = np.array([[0.0, 0.0]] * len(x))
18     for i in range(0, len(x)):
19         d = n.dot(x[i] - o)
20         if d < dhat:
21             s = d / dhat
22             g[i] = contact_area[i] * dhat * (kappa / 2 * (math
       .log(s) / dhat + (s - 1) / d)) * n
23     return g
24
25 def hess(x, n, o, contact_area):
26     IJV = [[0] * 0, [0] * 0, np.array([0.0] * 0)]
27     for i in range(0, len(x)):
28         d = n.dot(x[i] - o)
29         if d < dhat:
30             local_hess = contact_area[i] * dhat * kappa / (2 *
        d * d * dhat) * (d + dhat) * np.outer(n, n)
31             for c in range(0, 2):
```

```
38 def init_step_size(x, n, o, p):
39     alpha = 1
40     for i in range(0, len(x)):
41         p_n = p[i].dot(n)
42         if p_n < 0:
43             alpha = min(alpha, 0.9 * n.dot(x[i] - o) / -p_n)
44     return alpha
```

Case Study: Square on Slope

Without Friction



Case Study: Square on Slope

Friction Implementation

$$\boxed{-\nabla P_{f,k}(x) = -\mu\lambda_k^n T_k^n \frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|}\bar{\mathbf{v}}_k}$$

$$\frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|} = \begin{cases} -\frac{\|\bar{\mathbf{v}}_k\|}{\epsilon_v^2} + \frac{2}{\epsilon_v}, & \|\bar{\mathbf{v}}_k\| \in [0, \epsilon_v) \\ 1/\|\bar{\mathbf{v}}_k\|, & \|\bar{\mathbf{v}}_k\| \geq \epsilon_v, \end{cases}$$

$$-\nabla P_{f,k}(x) = -\mu\lambda_k^n T_k^n \frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|}\bar{\mathbf{v}}_k = 0 \quad \text{when} \quad \|\bar{\mathbf{v}}_k\| = 0$$

$$\boxed{\begin{aligned} &\nabla^2 P_f(x) \\ &= \sum_k \mu\lambda_k^n T_k^n \left(\frac{f_1'(\|\bar{\mathbf{v}}_k\|)\|\bar{\mathbf{v}}_k\| - f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|^3}\bar{\mathbf{v}}_k\bar{\mathbf{v}}_k^T + \frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|}\mathbf{I}_3 \right) T_k^{nT} \frac{\partial v}{\partial x} \end{aligned}}$$

$$\frac{f_1'(\|\bar{\mathbf{v}}_k\|)\|\bar{\mathbf{v}}_k\| - f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|^2} = \begin{cases} -1/\epsilon_v^2, & \|\bar{\mathbf{v}}_k\| \in [0, \epsilon_v) \\ -1/\|\bar{\mathbf{v}}_k\|^2, & \|\bar{\mathbf{v}}_k\| \geq \epsilon_v, \end{cases}$$

$$\nabla^2 P_{f,k}(x) = \mu\lambda_k^n T_k^n \left(\frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|}\mathbf{I}_3 \right) T_k^{nT} \frac{\partial v}{\partial x} \quad \text{when} \quad \|\bar{\mathbf{v}}_k\| = 0$$

FrictionEnergy.py

```python
import numpy as np
import utils

epsv = 1e-3

def f0(vbarnorm, epsv, hhat):
    if vbarnorm >= epsv:
        return vbarnorm * hhat
    else:
        vbarnormhhat = vbarnorm * hhat
        epsvhhat = epsv * hhat
        return vbarnormhhat * vbarnormhhat * (-vbarnormhhat / 3.0 + epsvhhat) / (epsvhhat * epsvhhat) + epsvhhat / 3.0

def f1_div_vbarnorm(vbarnorm, epsv):
    if vbarnorm >= epsv:
        return 1.0 / vbarnorm
    else:
        return (-vbarnorm + 2.0 * epsv) / (epsv * epsv)

def f_hess_term(vbarnorm, epsv):
    if vbarnorm >= epsv:
        return -1.0 / (vbarnorm * vbarnorm)
    else:
        return -1.0 / (epsv * epsv)
```

Case Study: Square on Slope

Friction Implementation (Cont.)

$$\boxed{-\nabla P_{f,k}(x) = -\mu\lambda_k^n T_k^n \frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|}\bar{\mathbf{v}}_k}$$

$$\frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|} = \begin{cases} -\frac{\|\bar{\mathbf{v}}_k\|}{\epsilon_v^2} + \frac{2}{\epsilon_v}, & \|\bar{\mathbf{v}}_k\| \in [0, \epsilon_v) \\ 1/\|\bar{\mathbf{v}}_k\|, & \|\bar{\mathbf{v}}_k\| \geq \epsilon_v, \end{cases}$$

$$-\nabla P_{f,k}(x) = -\mu\lambda_k^n T_k^n \frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|}\bar{\mathbf{v}}_k = 0 \quad \text{when} \quad \|\bar{\mathbf{v}}_k\| = 0$$

$$\boxed{\begin{aligned} &\nabla^2 P_f(x) \\ &= \sum_k \mu\lambda_k^n T_k^n \left(\frac{f_1'(\|\bar{\mathbf{v}}_k\|)\|\bar{\mathbf{v}}_k\| - f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|^3}\bar{\mathbf{v}}_k\bar{\mathbf{v}}_k^T + \frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|}\mathbf{I}_3 \right) T_k^{nT} \frac{\partial v}{\partial x} \end{aligned}}$$

$$\frac{f_1'(\|\bar{\mathbf{v}}_k\|)\|\bar{\mathbf{v}}_k\| - f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|^2} = \begin{cases} -1/\epsilon_v^2, & \|\bar{\mathbf{v}}_k\| \in [0, \epsilon_v) \\ -1/\|\bar{\mathbf{v}}_k\|^2, & \|\bar{\mathbf{v}}_k\| \geq \epsilon_v, \end{cases}$$

$$\nabla^2 P_{f,k}(x) = \mu\lambda_k^n T_k^n \left(\frac{f_1(\|\bar{\mathbf{v}}_k\|)}{\|\bar{\mathbf{v}}_k\|}\mathbf{I}_3 \right) T_k^{nT} \frac{\partial v}{\partial x} \quad \text{when} \quad \|\bar{\mathbf{v}}_k\| = 0$$

FrictionEnergy.py

```
def val(v, mu_lambda, hhat, n):
    sum = 0.0
    T = np.identity(2) - np.outer(n, n) # tangent of slope is constant
    for i in range(0, len(v)):
        if mu_lambda[i] > 0:
            vbar = np.transpose(T).dot(v[i])
            sum += mu_lambda[i] * f0(np.linalg.norm(vbar), epsv, hhat)
    return sum

def grad(v, mu_lambda, hhat, n):
    g = np.array([[0.0, 0.0]] * len(v))
    T = np.identity(2) - np.outer(n, n) # tangent of slope is constant
    for i in range(0, len(v)):
        if mu_lambda[i] > 0:
            vbar = np.transpose(T).dot(v[i])
            g[i] = mu_lambda[i] * f1_div_vbarnorm(np.linalg.norm(vbar), epsv) * T.dot(vbar)
    return g

def hess(v, mu_lambda, hhat, n):
    IJV = [[0] * 0, [0] * 0, np.array([0.0] * 0)]
    T = np.identity(2) - np.outer(n, n) # tangent of slope is constant
    for i in range(0, len(v)):
        if mu_lambda[i] > 0:
            vbar = np.transpose(T).dot(v[i])
            vbarnorm = np.linalg.norm(vbar)
            inner_term = f1_div_vbarnorm(vbarnorm, epsv) * np.identity(2)
            if vbarnorm != 0:
                inner_term += f_hess_term(vbarnorm, epsv) / vbarnorm * np.outer(vbar, vbar)
            local_hess = mu_lambda[i] * T.dot(utils.make_PD(inner_term)).dot(np.transpose(T)) / hhat
            for c in range(0, 2):
                for r in range(0, 2):
                    IJV[0].append(i * 2 + r)
                    IJV[1].append(i * 2 + c)
                    IJV[2] = np.append(IJV[2], local_hess[r, c])
    return IJV
```

Case Study: Square on Slope

Calculating Normal Force Magnitude

$$\lambda_k = -w_k \frac{\partial b}{\partial d_k} \text{ is the contact force magnitude}$$

```python
def compute_mu_lambda(x, n, o, contact_area, mu):
    mu_lambda = np.array([0.0] * len(x))
    for i in range(0, len(x)):
        d = n.dot(x[i] - o)
        if d < dhat:
            s = d / dhat
            mu_lambda[i] = mu * -contact_area[i] * dhat * (kappa / 2 * (math.log(s) / dhat + (s - 1) / d))
    return mu_lambda
```

BarrierEnergy.py

```python
def step_forward(x, e, v, m, l2, k, n, o, contact_area, mu, is_DBC, h, tol):
    x_tilde = x + v * h      # implicit Euler predictive position
    x_n = copy.deepcopy(x)
    mu_lambda = BarrierEnergy.compute_mu_lambda(x, n, o, contact_area, mu)  # compute mu * lambda for each node using x^n

    # Newton loop
```

time_integrator.py

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

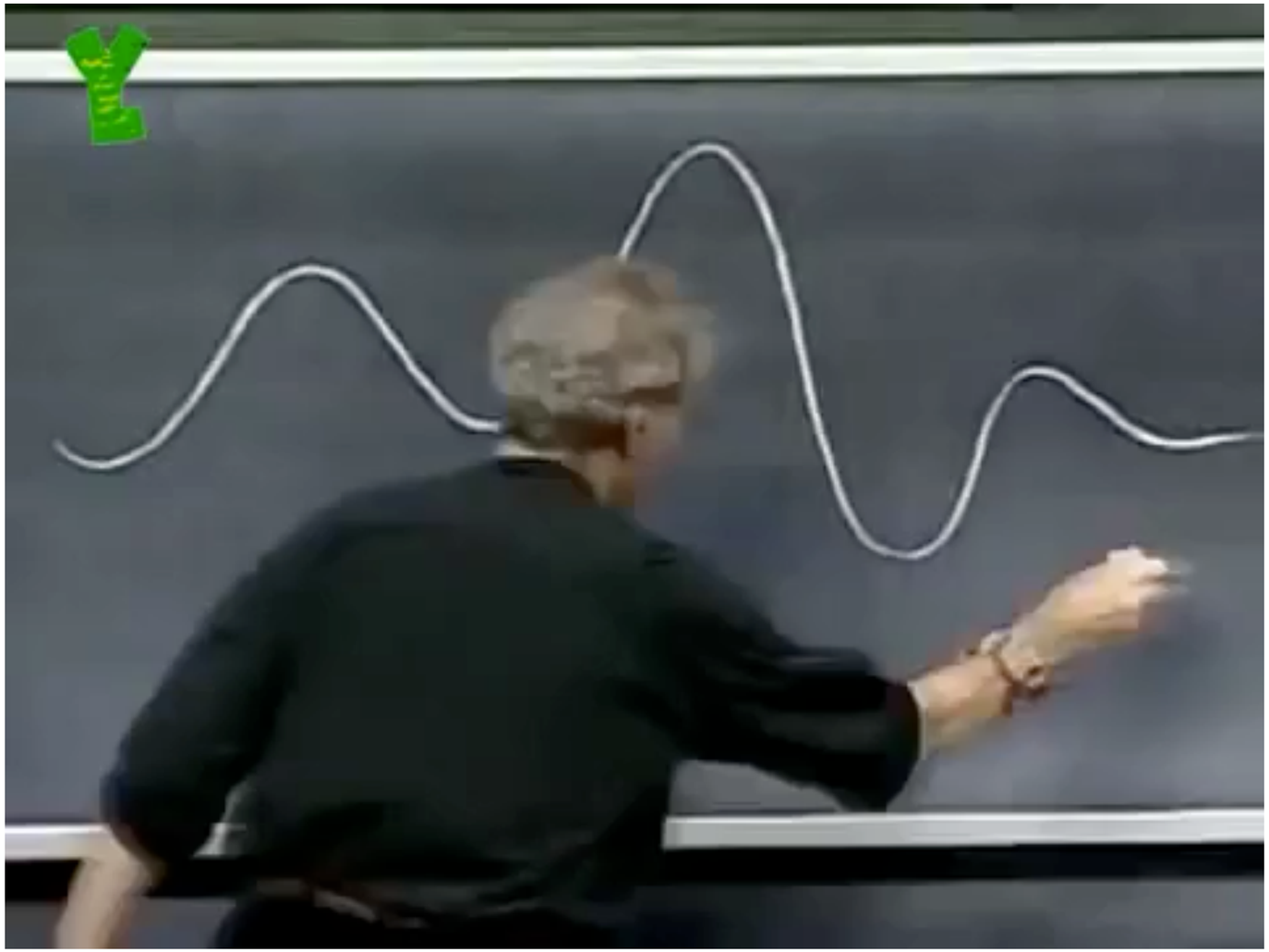
  Mollification, Semi-Implicit Approx., Fixed-Point Iteration

- Case Study: Square on Slope

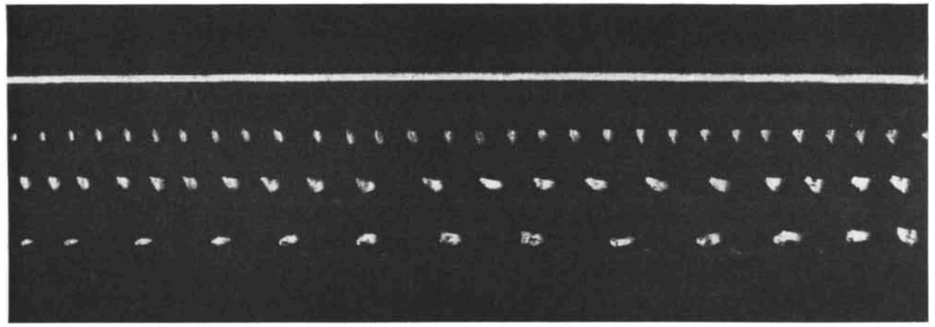
  Avoid Numerical Issues

- Remarks

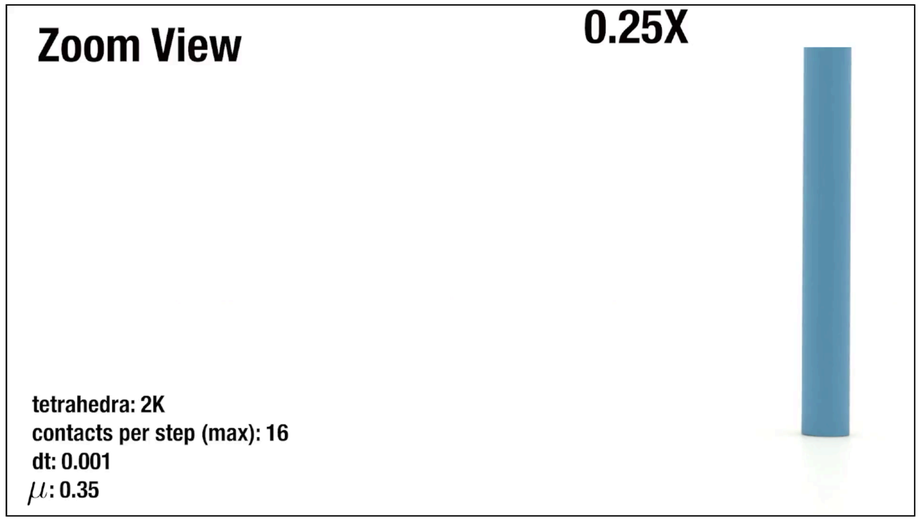
Remarks — Stick-Slip Instability Effect



Rabinowicz, Ernest. "Stick and slip." Scientific American 194, no. 5 (1956): 109-119.



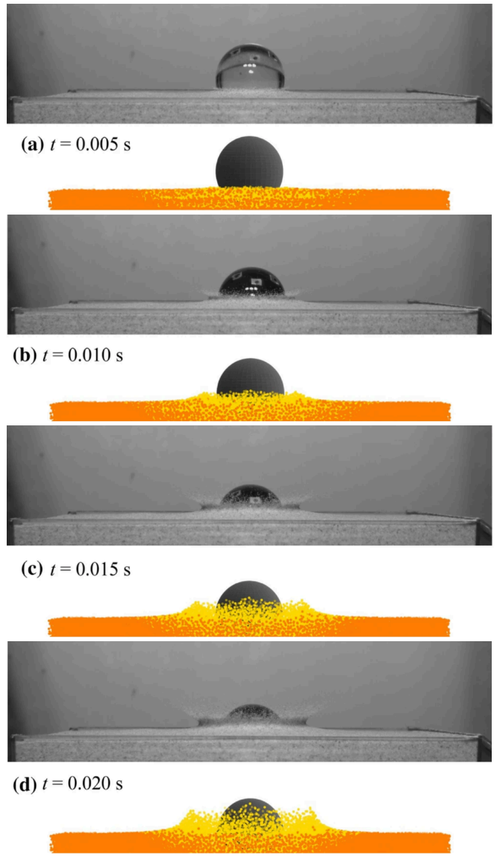
CHALK MARKS on a blackboard demonstrate stick-slip. The top mark was made by a piece of chalk held at an acute angle to the direction of motion; the marks below it, by pieces of chalk held at an obtuse angle to this direction. In the latter marks the chalk stuck to the blackboard, then slipped, then stuck again and so on. The more tightly the chalk is held, the smaller the distance of slip.

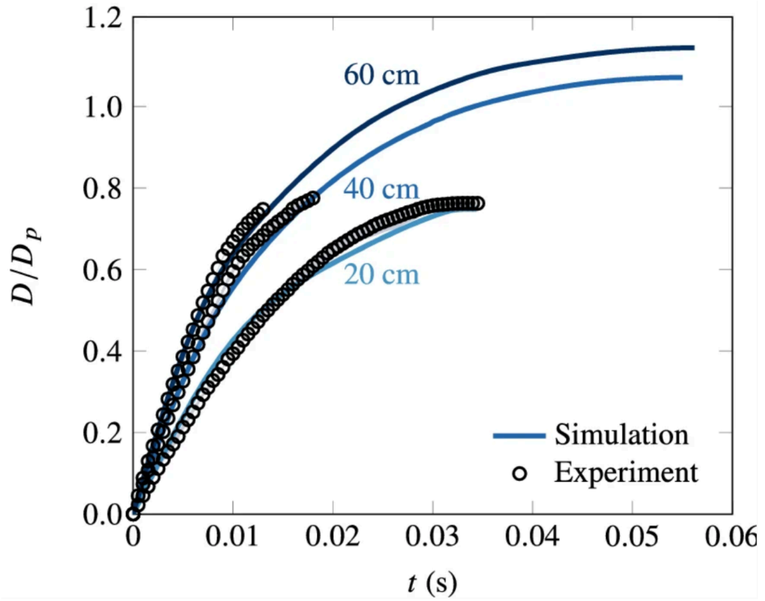


Remarks — Applications

Computational Mechanics

- Jiang et al. [2022]
  - Uses IPC to couple discrete element method (DEM) and material point method (MPM)
- …

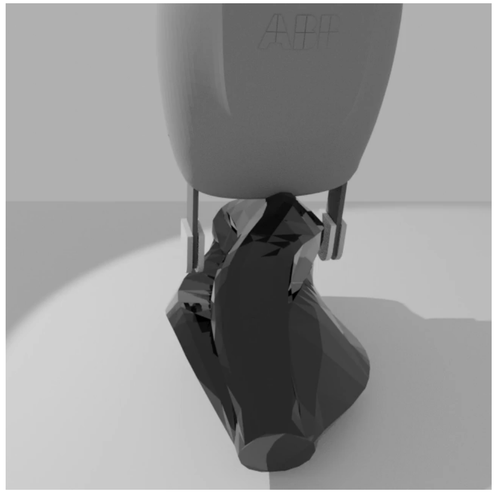




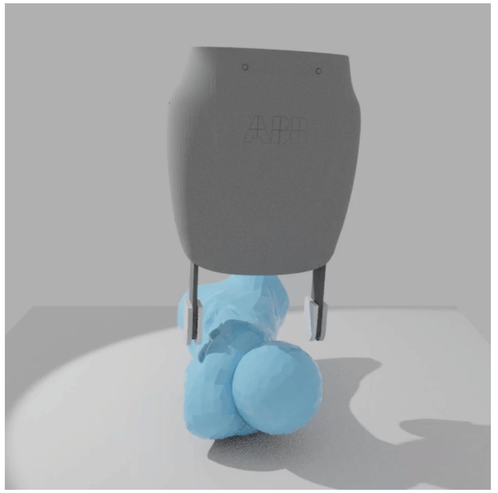
Remarks — Applications

Robotics

- IPC-GraspSim [Kim et al. 2022]
  - Compares grasping results between IPC simulations and experiments
- …



Successful Grasp: Vase



Failed Grasp: Pawn

Grasp sets (5 trials/grasp)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
IPC-GraspSim	0.0	0.0	1.0	0.0	0.0	1.0	0.4	0.8	1.0	1.0	1.0	1.0	1.0	1.0	0.6	0.0	0.0	0.0	0.4	0.0	0.0	1.0	1.0	0.0	0.0
Physical Experiments	0.0	1.0	1.0	0.0	0.0	1.0	0.8	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.2	0.0	0.0	0.4	1.0	0.8	0.0	0.0

Today:

- **Formulation**

  Mollification, Semi-Implicit Approx., Fixed-Point Iteration

- **Case Study: Square on Slope**

  Avoid Numerical Issues

- **Remarks**

  Stick-Slip Instability, Applications

Next Lecture: Moving Boundary Conditions

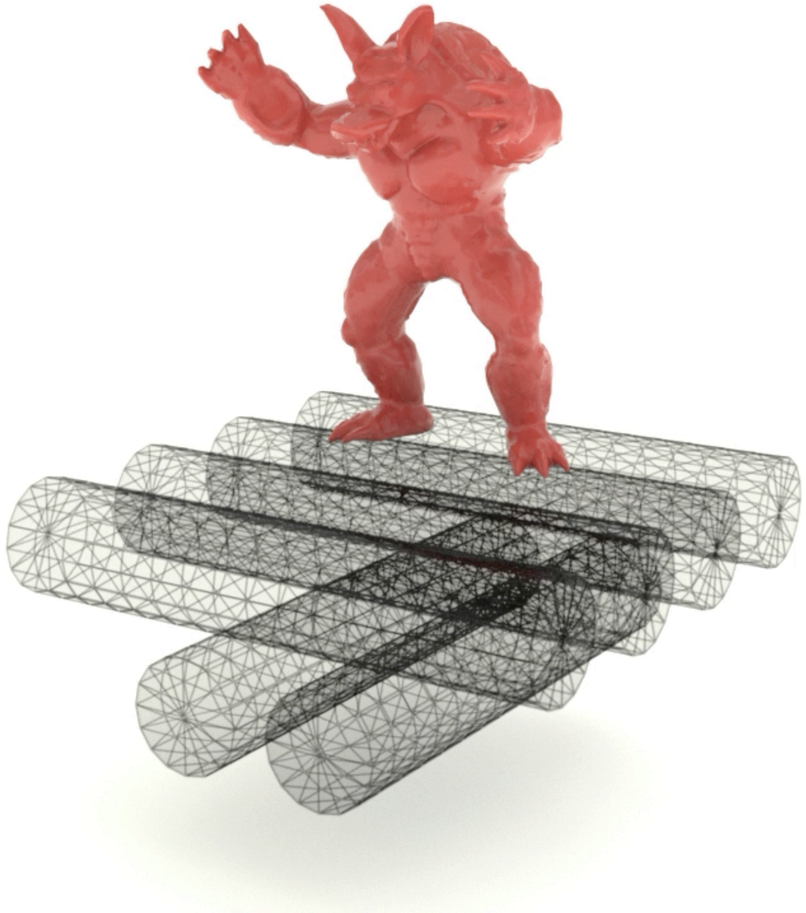


Image Sources

- https://nacho.blogs.uv.es/2017/11/20/conformation-constraints-for-efficient-viscoelastic-fluid-simulation/
- https://www.gauss-centre.eu/results/computational-and-scientific-engineering/dns-of-airfoil-acoustics
- https://iselinc.com/explore-many-benefits-lubrication/
- https://en.wikipedia.org/wiki/Friction#/media/File:Friction_between_surfaces.jpg