Lecture 7

Deformable Models

FUNDAMENTALS OF COMPUTER GRAPHICS

Animation & Simulation

Stanford CS248B, Fall 2022

PROFS. KAREN LIU & DOUG JAMES

Announcements

- Update: Just 3 programming assignments
  - Pinball + deformation + characters
  - P1: Pinball!
    - Due on Tues, Oct 25 (extended!)
  - P2: Blob Factory
    - Out on Tues, Oct 25
    - Due on Thurs, Nov 10
  - P3: Inverse Kinematics
    - Out on Tues, Nov 15
- HW2:
  - Out Thurs, Oct 20
  - Due Thurs, Oct 27
- CS248B face-mask policy
  - Still required in this class by instructor.
  - If you need a mask please ask.

Stanford CS 248B Fall 2022 – Programming Assignment #1

Pinball!

Colliding Particles for Fun

Summary

In your first programming assignment you will design, model, simulate and play a virtual game of pinball. The main challenge is to simulate the ball motion subject to collisions with an environment composed of rigid obstacles (mostly static but some moving). The environment will be modeled using simple 2D shape primitives, for which we can use signed-distance fields (SDFs) to help process collisions robustly, even for fast-moving balls. Your implementation will be done in Open Processing (OP), and you and your classmates will be able to share and play each others' games (without sharing the code). Yes, this will be a groovy start to CS248B.

https://en.wikipedia.org/wiki/Pinball

2D Particle Simulation

For this first assignment, we will model the ball using a simple 2D point-like circular primitive (with unit mass, m=1, radius r, position **p**, and velocity **v**) falling under a small gravitational acceleration, **g**. This assumption ignores the more complex dynamics of a 3D spherical balling with rolling and frictional contact. The main challenge is not the time-stepping of the free-flight motion, but the detection and robust processing of ball-object collisions. As discussed in class, you can use a Symplectic Euler integrator for the ball dynamics, and use impulses to modify the velocity to resolve collisions. In difficult cases, such as for fast-moving balls, you will need to perform continuous collision processing to avoid missing collisions or having objects interpenetrate (more on that later).

The starter code includes a `Ball class` that can `draw()` a circle, and has a `timestep(dt)` method that performs discrete collision detection using the scene's signed-distance field, and applies a suitable collision impulse. You will need to modify the time-stepping scheme to handle collisions more robustly, as discussed below.

Announcements

- Re: Pinball questions and feedback
- Suggestion:
  - Do non-ray-marching parts first
  - Then do ray marching
- Ray marching:
  - Process all collisions on each timestep
  - Multiple collisions possible per timestep
    - May want to limit # for performance
  - Tip: Debug on simple scene first
    - 1D test problem w/ known solution
    - E.g., pinball in empty box
  - See discussion on depthMin and vn>0

Stanford CS 248B Fall 2022 – Programming Assignment #1

Pinball!

Colliding Particles for Fun

Summary

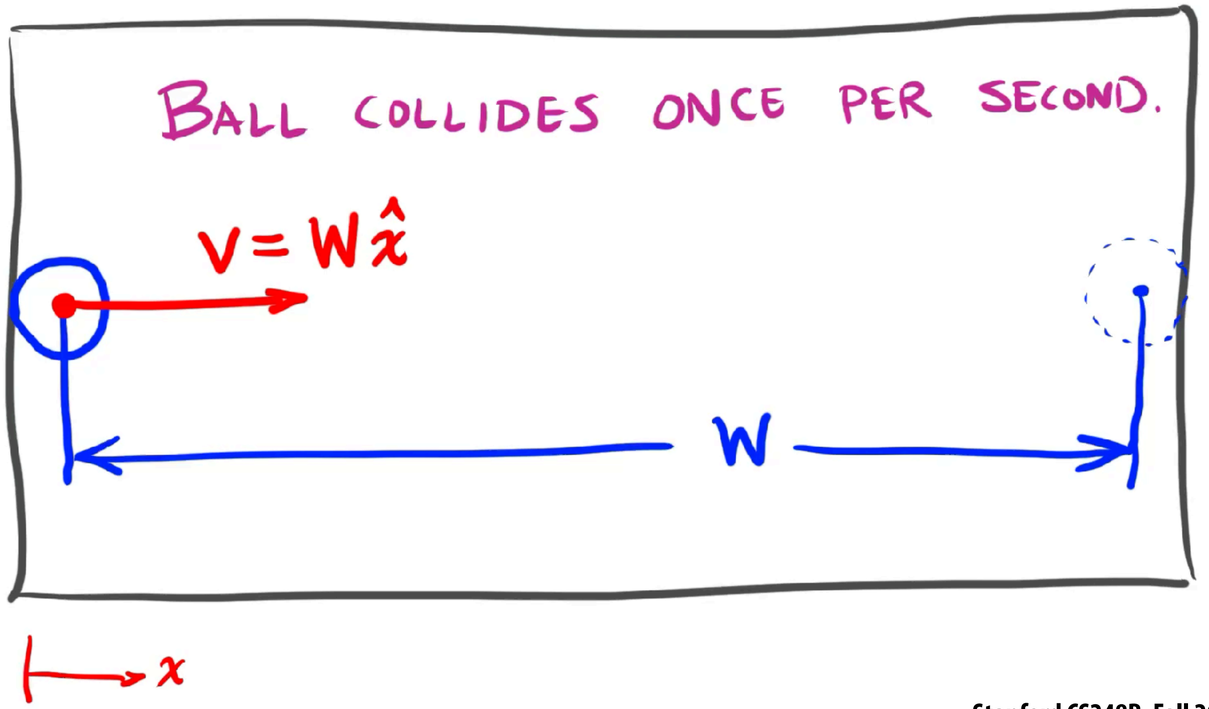
In your first programming assignment you will design, model, simulate and play a virtual game of pinball. The main challenge is to simulate the ball motion subject to collisions with an environment composed of rigid obstacles (mostly static but some moving). The environment will be modeled using simple 2D shape primitives, for which we can use signed-distance fields (SDFs) to help process collisions robustly, even for fast-moving balls. Your implementation will be done in Open Processing (OP), and you and your classmates will be able to share and play each others' games (without sharing the code). Yes, this will be a groovy start to CS248B.

https://en.wikipedia.org/wiki/Pinball

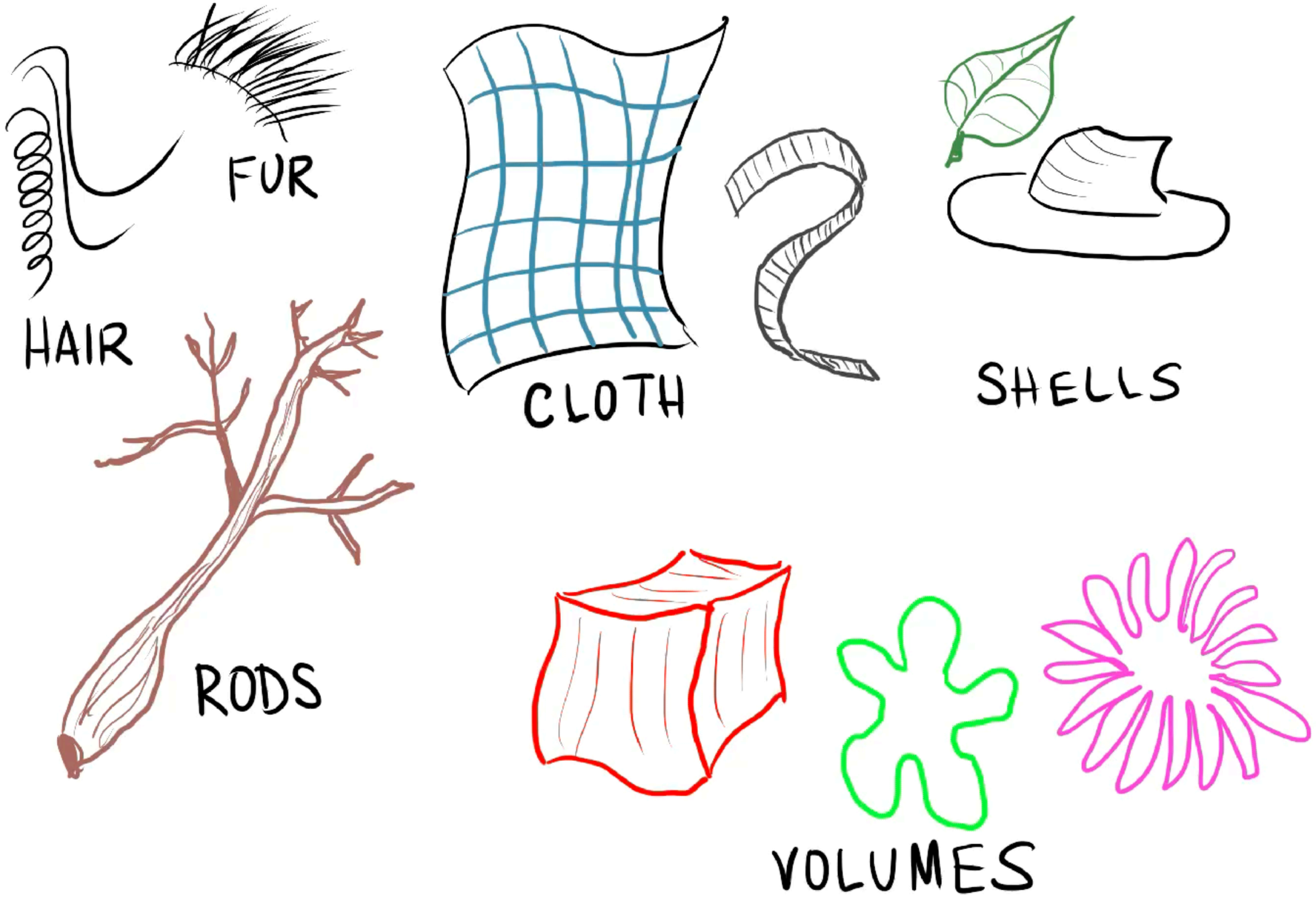
2D Particle Simulation

For this first assignment, we will model the ball using a simple 2D point-like circular primitive (with unit mass, m=1, radius r, position **p**, and velocity **v**) falling under a small gravitational acceleration, **g**. This assumption ignores the more complex dynamics of a 3D spherical balling with rolling and frictional contact. The main challenge is not the time-stepping of the free-flight motion, but the detection and robust processing of ball-object collisions. As discussed in class, you can use a Symplectic Euler integrator for the ball dynamics, and use impulses to modify the velocity to resolve collisions. In difficult cases, such as for fast-moving balls, you will need to perform continuous collision processing to avoid missing collisions or having objects interpenetrate (more on that later).

The starter code includes a `Ball class` that can `draw()` a circle, and has a `timestep(dt)` method that performs discrete collision detection using the scene's signed-distance field, and applies a suitable collision impulse. You will need to modify the time-stepping scheme to handle collisions more robustly, as discussed below.



Physics-based deformable models in graphics



Modeling deformation forces

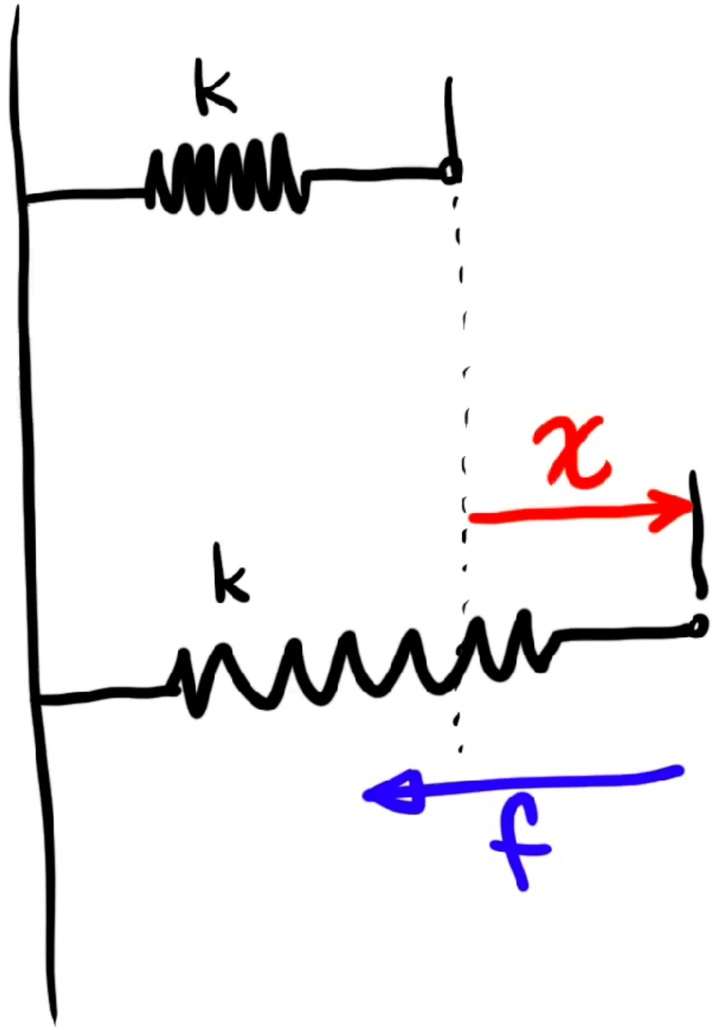
Reference:

David Baraff and Andrew Witkin, Physically Based Modeling, Online SIGGRAPH 2001 Course Notes, 2001.

- https://graphics.pixar.com/pbm2001
- Particle System Dynamics,
- Cloth and Fur Energy Functions

Recall: Hooke's law for 1D linear springs

- 1D elastic force



$$f = -k\chi$$

spring stiffness

displacement

Recall: Hooke's law for 1D linear springs

- Potential energy model

$$E(x) = \frac{1}{2} k x^2$$

$$f(x) = -\frac{d}{dx} E(x) = -kx$$

Conservative forces (preserve energy)

$$\text{WORK} = \int_{x_i}^{x_f} f(x)\, dX = -\int_{x_i}^{x_f} \frac{dE}{dX}\, dX = E(x_i) - E(x_f)$$

Hooke's Law for nD linear springs (zero rest-length)

$$E(p) = \frac{1}{2} k \| p - \bar{p} \|^2$$

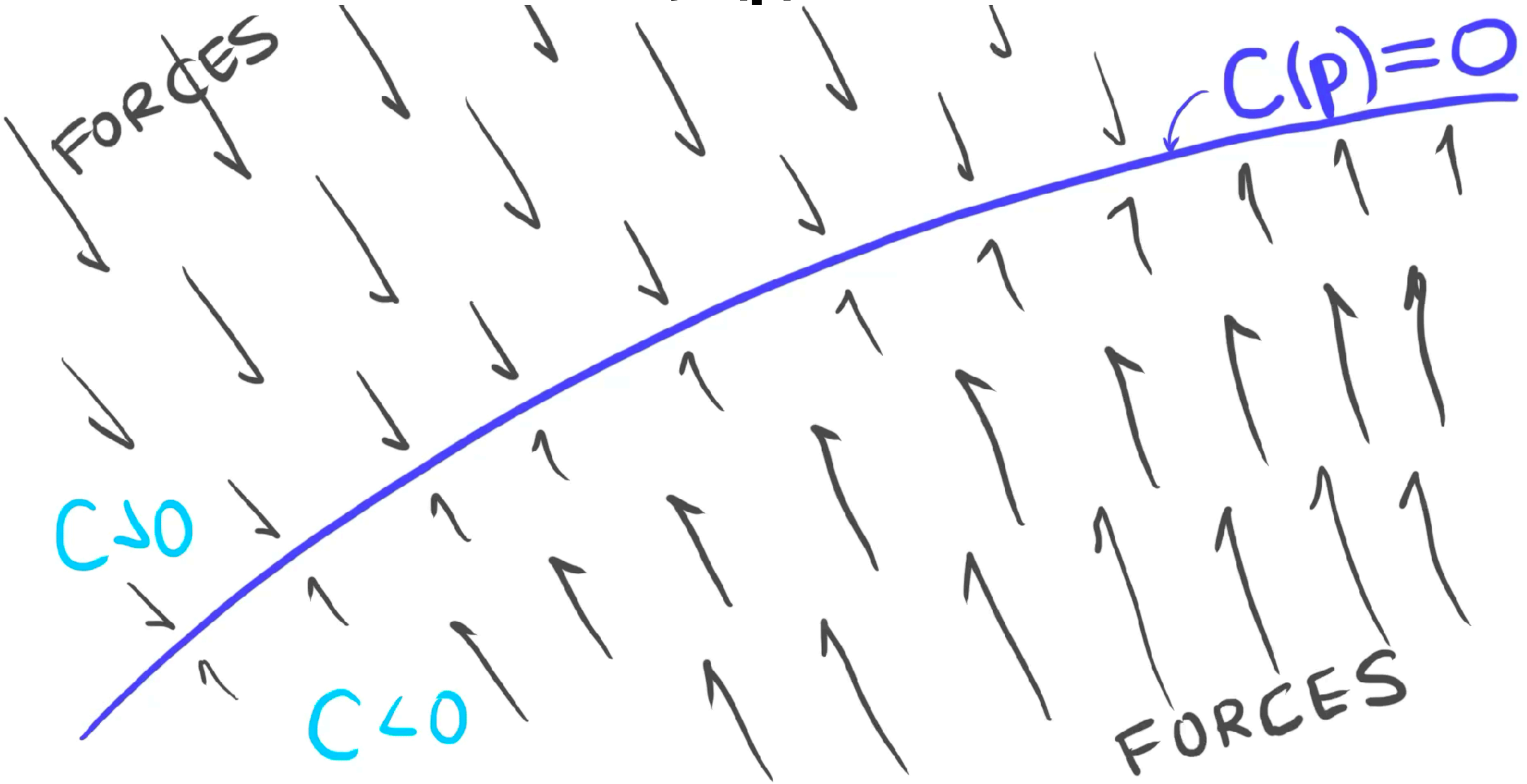
↑ rest position of particle

$$f(p) = -\frac{\partial}{\partial p} E(p) = -\frac{1}{2} k \frac{\partial}{\partial p} \| p - \bar{p} \|^2$$

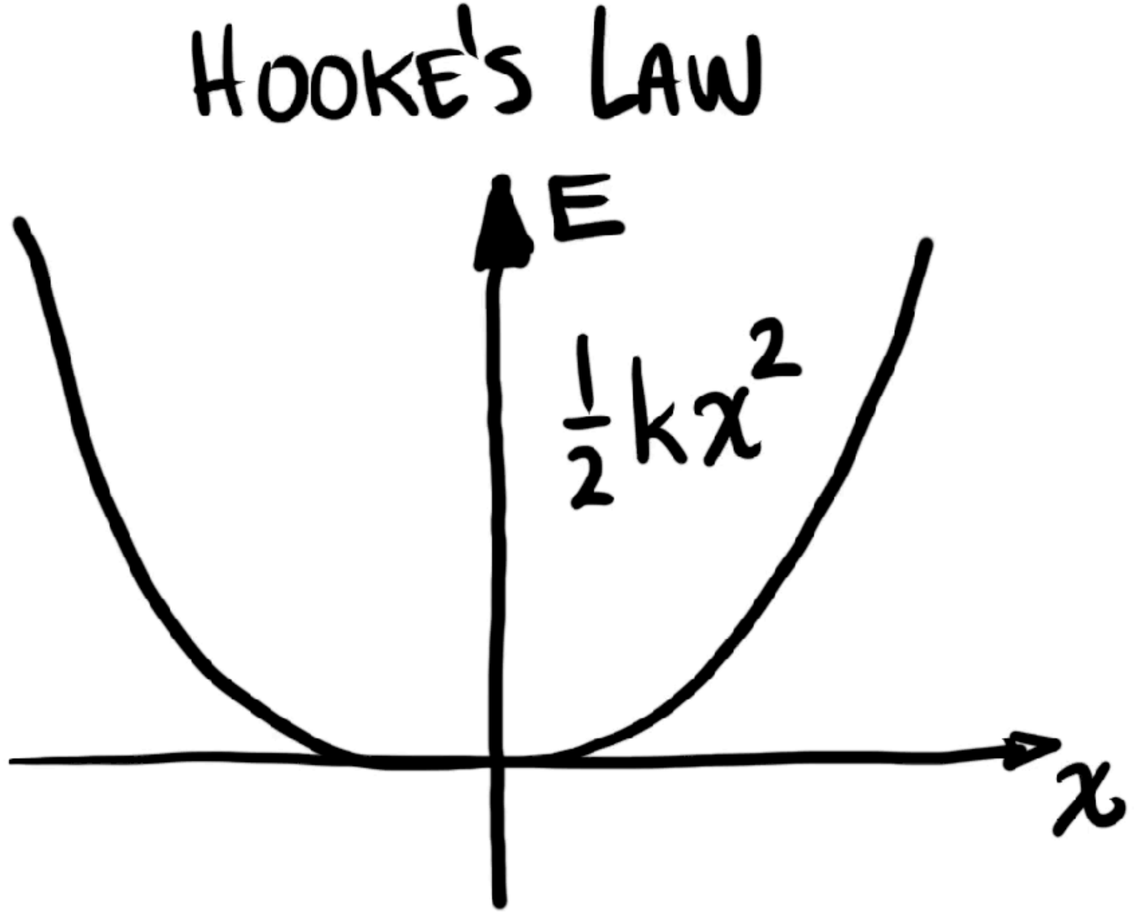
$$= -k(p - \bar{p})$$

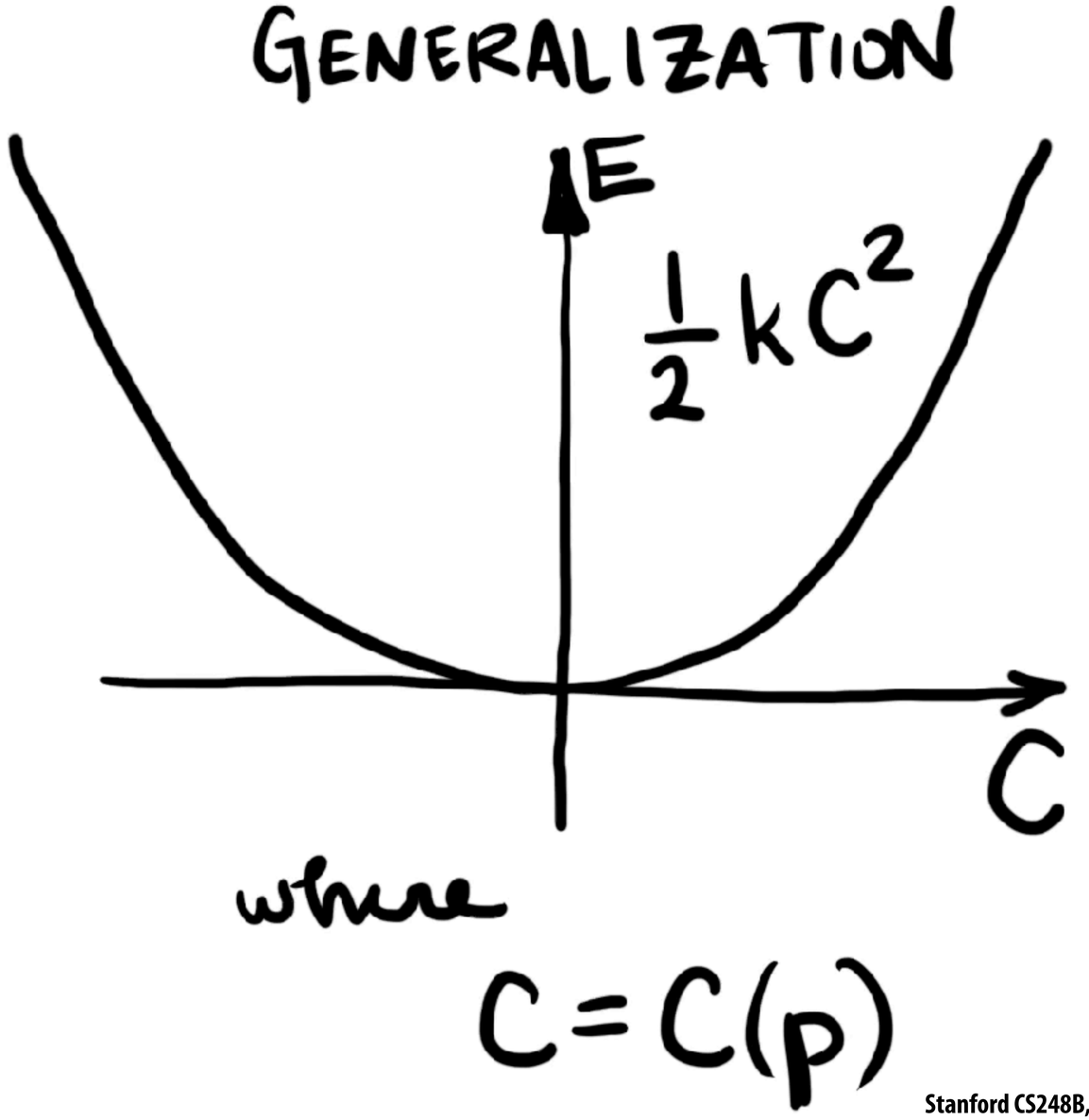
(pulls p back to $\bar{p}$)

Soft Constraint Functions, C(p)



Generalizing Hooke's Law: From $x \to 0$ to $C(\mathbf{p}) \to 0$.





Generalized spring model (one constraint)

$C(p) \in \mathbb{R}$: Generalized displacement

(soft constraint function)

Want $C \to 0$

$$E(p) = \frac{1}{2} k C^2(p)$$

$$f_i(p) = -\nabla_{p_i} E(p) = -\frac{\partial E(p)}{\partial p_i} = -k\, C\, \frac{\partial C}{\partial p_i}$$

stiffness, displacement, direction

Generalized spring model (M constraints)

Consider M constraint functions,

$$\mathbf{C} = \begin{pmatrix} C_1(\mathbf{p}) \\ C_2(\mathbf{p}) \\ \vdots \\ C_M(\mathbf{p}) \end{pmatrix} : \mathbb{R}^{nN} \longrightarrow \mathbb{R}^M$$

Generalized spring model (M constraints)

TOTAL ENERGY IS SUM OF SPRING ENERGIES:

$$E(\mathbf{p}) = \sum_{i=1}^{M} \frac{1}{2} k_i C_i^2(\mathbf{p}) \xrightarrow{k=k_i} \frac{1}{2} k\, \mathbf{C} \cdot \mathbf{C}$$

$$= \frac{1}{2} k\, \mathbf{C}^T \mathbf{C} = \frac{1}{2} k \|\mathbf{C}\|^2$$

Generalized spring model (M constraints)

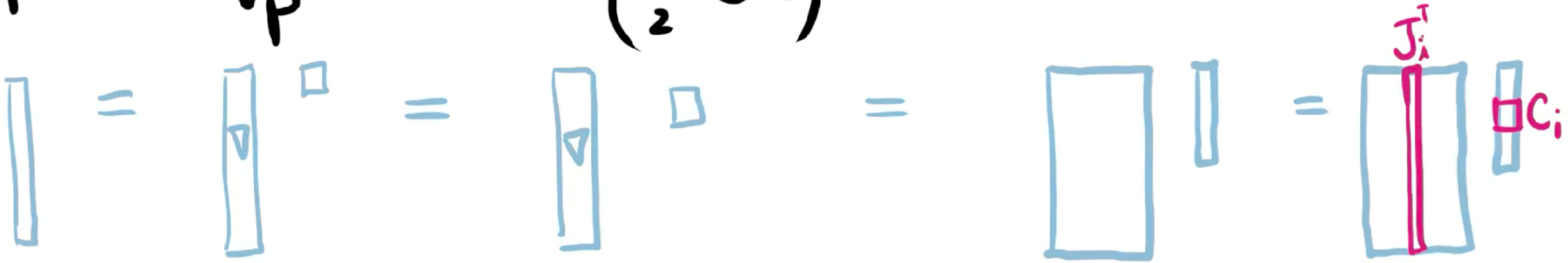
Force on particle i,

$$f_i = -\frac{\partial E}{\partial p_i} = -\frac{\partial}{\partial p_i}\left(\frac{1}{2}k\sum_{j=1}^{M} C_j^2(p)\right) = -k\sum_{j=1}^{M} C_j \frac{\partial C_j}{\partial p_i}$$

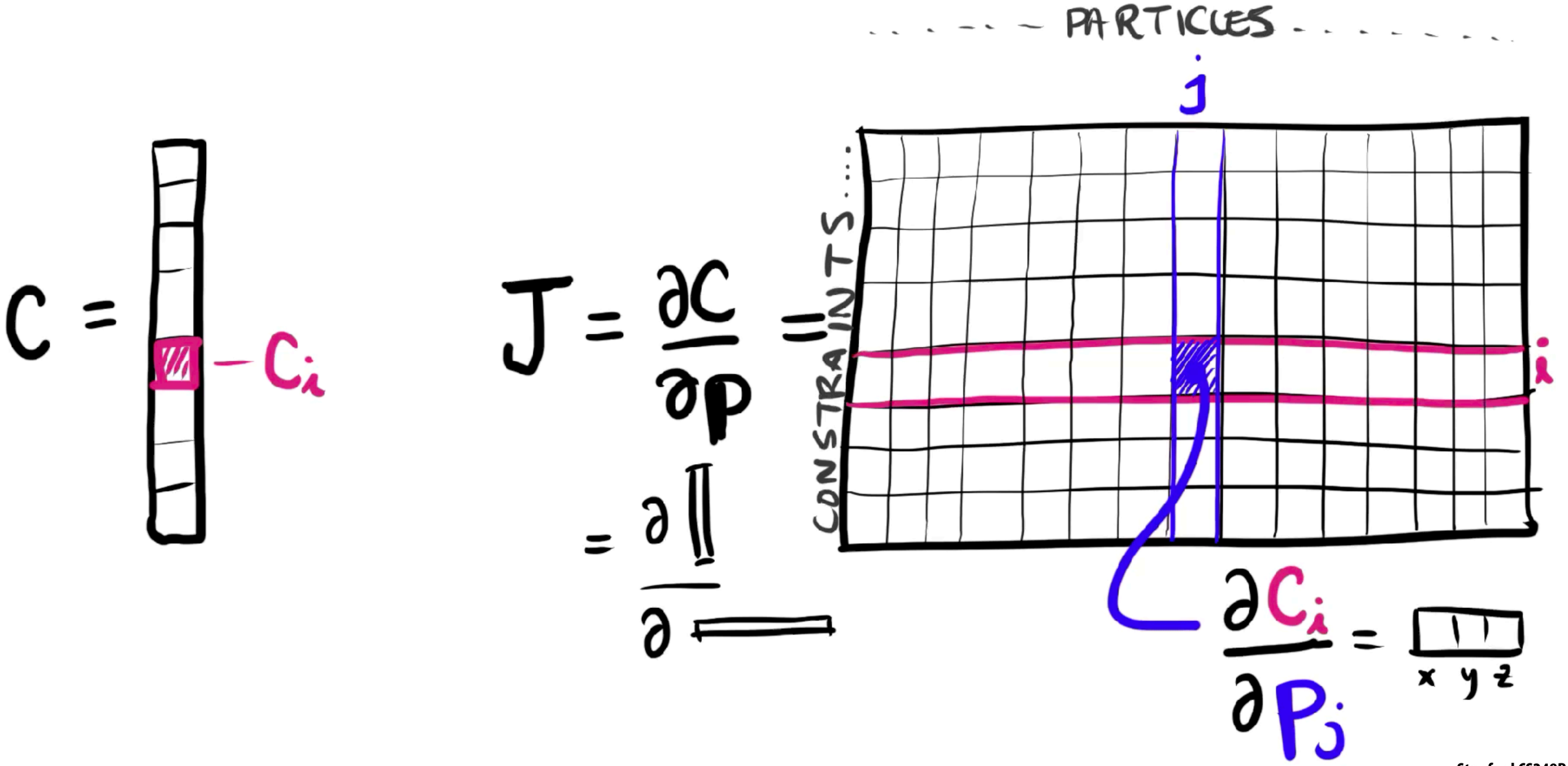
stiffness, displacement, direction

Force on particle system

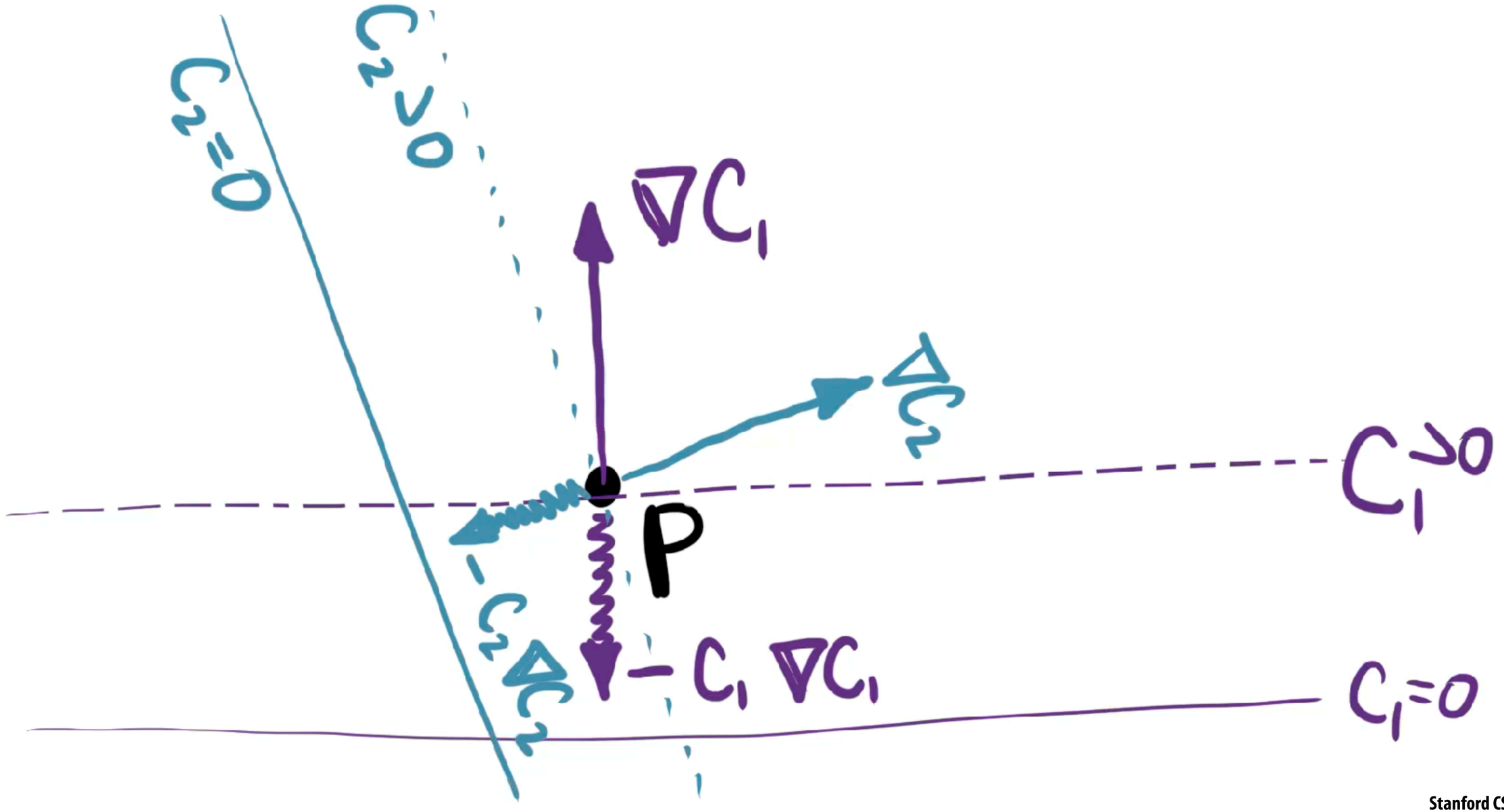
$$f = -\nabla_p E = -\nabla\left(\frac{k}{2} C^T C\right) = -k\,\nabla C^T C = -k J^T C$$



Constraint vector and Jacobian matrix



Geometric picture of generalized spring forces



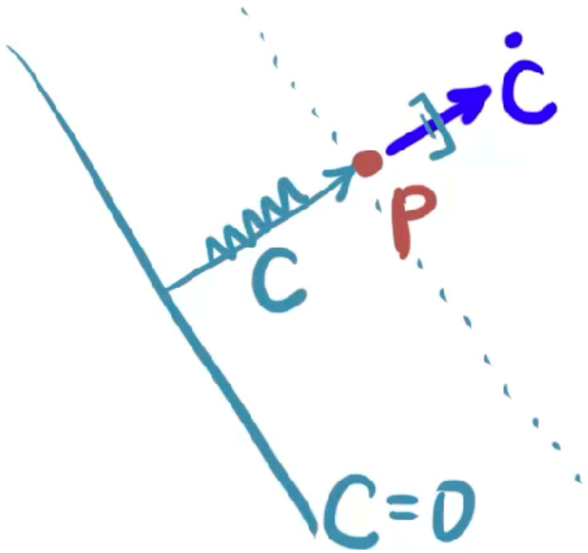
Generalized damped springs

Hooke's law
+ damping

$$f = -k_s x - k_d \dot{x}$$

Generalization
of damped spring

$$f = -J^T(k_s C + k_d \dot{C})$$



Taking derivatives

of things we'll need

PIXAR

FEATURE FILMS SHORT FILMS TECHNOLOGY CAREERS EXTRAS ABOUT

AWWW...DON'T CRY.

Notation: Gradient w.r.t. particle positions

Particle position variables:

$$p_i, \; i = 1, \ldots, N \qquad \text{where} \qquad p_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \in \mathbb{R}^3 \qquad (n = 3)$$

Gradient w.r.t. particle position:

$\frac{\partial}{\partial p_i}$ or ∇_{p_i} or ∂_i is shorthand for $\begin{pmatrix} \frac{\partial}{\partial x_i} \\ \frac{\partial}{\partial y_i} \\ \frac{\partial}{\partial z_i} \end{pmatrix}$ OR $\equiv \begin{pmatrix} \frac{\partial}{\partial x_i} & \frac{\partial}{\partial y_i} & \frac{\partial}{\partial z_i} \end{pmatrix}$

- it's 3 derivatives in 3D.
- arrange them however is convenient.

Gradient of a constant (!)

Function

$$g(P) = c \in \mathbb{R}^1$$

Gradient w.r.t. $P_i \in \mathbb{R}^3$:

$$\partial_{P_i} g(P) = \partial_{P_i} c = \vec{0} \in \mathbb{R}^3 \qquad \text{e.g., } (0\ 0\ 0)$$

Gradient of some particle's position

Function:

$$g(P) = P_j = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \in \mathbb{R}^3$$

$j \in \{1, 2, \ldots, N\}$ dummy index variable

Gradient w.r.t. P_i:

$$\partial_{P_i} g = \frac{\partial P_j}{\partial P_i} = \frac{\partial \begin{bmatrix} \\ \\ \\ \end{bmatrix}}{\partial \square\square\square} = \begin{bmatrix} \frac{\partial x_j}{\partial x_i} & \frac{\partial x_j}{\partial y_i} & \frac{\partial x_j}{\partial z_i} \\ \frac{\partial y_j}{\partial x_i} & \frac{\partial y_j}{\partial y_i} & \frac{\partial y_j}{\partial z_i} \\ \frac{\partial z_j}{\partial x_i} & \frac{\partial z_j}{\partial y_i} & \frac{\partial z_j}{\partial z_i} \end{bmatrix} = \delta_{ij} I_3 = \delta_{ij} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

"WHY?"

where $\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$ Krönecker delta.

Gradient of position difference

Function:

$$g(\mathbf{p}) = \mathbf{p}_k - \mathbf{p}_\ell \quad \text{for some } k \neq \ell.$$

Gradient:

$$\frac{\partial g}{\partial \mathbf{p}_i} = \partial_i (\mathbf{p}_k - \mathbf{p}_\ell) = \partial_i \mathbf{p}_k - \partial_i \mathbf{p}_\ell$$

$$= (\delta_{ik} - \delta_{i\ell}) \mathbf{I}_3$$

Gradient of distance squared (e.g., springs)

Function:

$$g(\mathbf{p}) = \|\mathbf{p}_k - \mathbf{p}_\ell\|^2 = (\mathbf{p}_k - \mathbf{p}_\ell)^T(\mathbf{p}_k - \mathbf{p}_\ell)$$

Gradient:

$$\partial_i g = \left(\partial_i(\mathbf{p}_k - \mathbf{p}_\ell)\right)^T(\mathbf{p}_k - \mathbf{p}_\ell) + (\mathbf{p}_k - \mathbf{p}_\ell)^T\left(\partial_i(\mathbf{p}_k - \mathbf{p}_\ell)\right)$$

$$= 2\left(\partial_i(\mathbf{p}_k - \mathbf{p}_\ell)\right)^T(\mathbf{p}_k - \mathbf{p}_\ell)$$

$$= 2\left[(\delta_{ik} - \delta_{i\ell})\mathbb{I}_3\right](\mathbf{p}_k - \mathbf{p}_\ell)$$

$$= 2\,(\delta_{ik} - \delta_{i\ell})(\mathbf{p}_k - \mathbf{p}_\ell)$$

Gradient of distance (e.g., springs)

Function: $g(\mathbf{p}) = \|\mathbf{p}_k - \mathbf{p}_l\| = \left(\|\mathbf{p}_k - \mathbf{p}_l\|^2\right)^{1/2}$

Gradient:

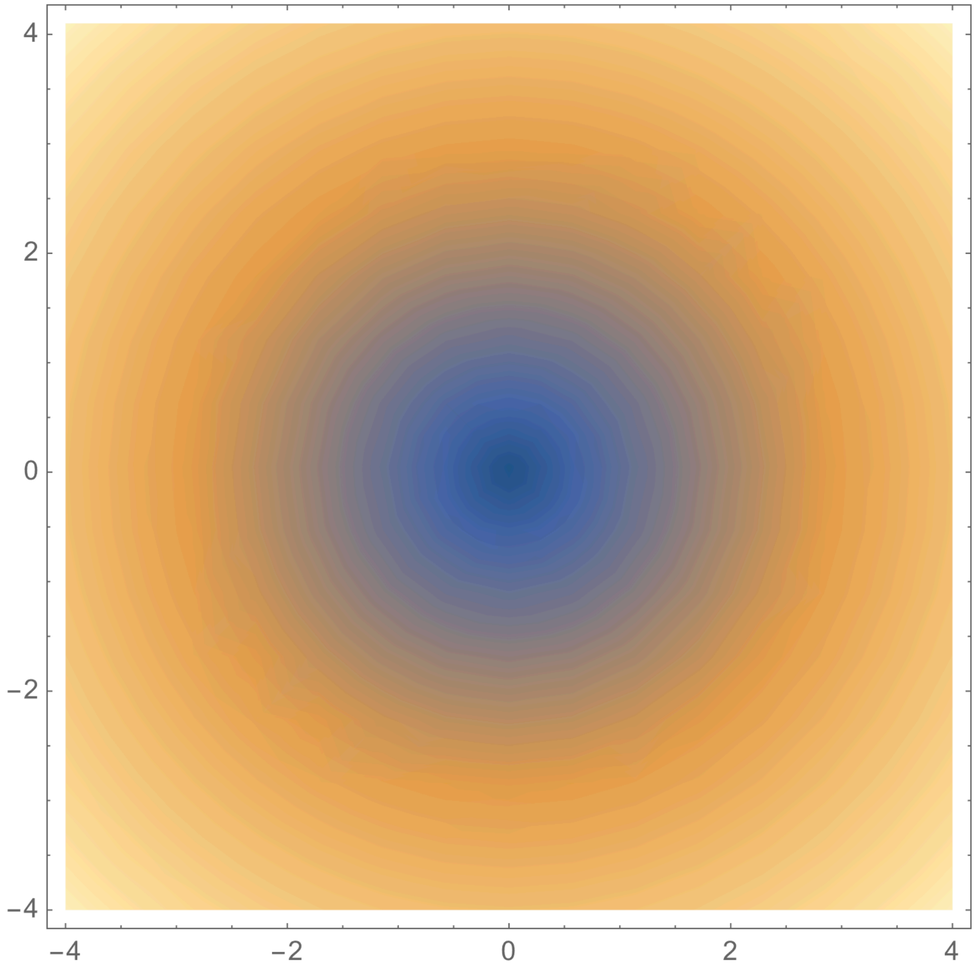
$$\partial_i g(\mathbf{p}) = \partial_i \left(\|\mathbf{p}_k - \mathbf{p}_l\|^2\right)^{1/2} = \frac{1}{2}\left(\|\mathbf{p}_k - \mathbf{p}_l\|^2\right)^{-1/2} \left(\partial_i \|\mathbf{p}_k - \mathbf{p}_l\|^2\right)$$

($1/R_{kl}$; from earlier)

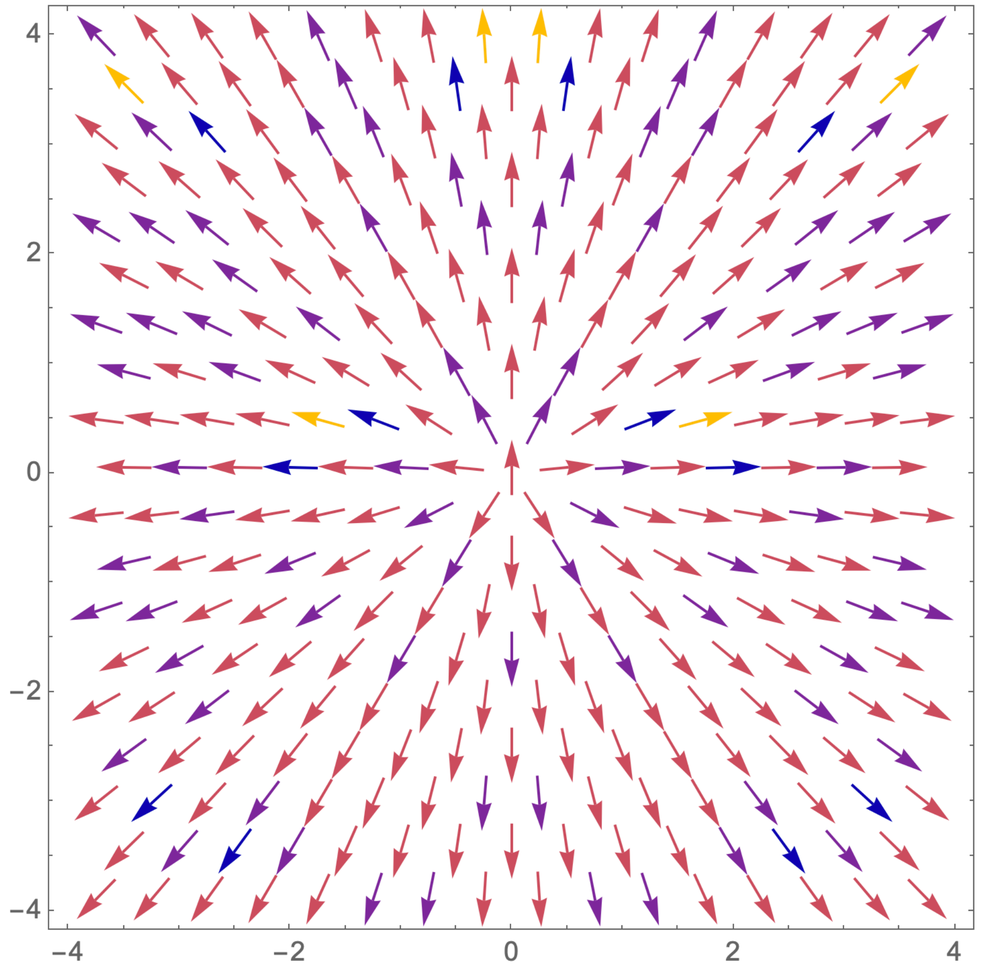
$$= \frac{1}{2} \frac{1}{\|\mathbf{p}_k - \mathbf{p}_l\|} \left(2\,(\delta_{ik} - \delta_{il})(\mathbf{p}_k - \mathbf{p}_l)\right)$$

$$= (\delta_{ik} - \delta_{il}) \underbrace{\frac{(\mathbf{p}_k - \mathbf{p}_l)}{\|\mathbf{p}_k - \mathbf{p}_l\|}}_{\text{unit vector, } \hat{R}_{kl}} = (\delta_{ik} - \delta_{il})\,\hat{R}_{kl}$$

Gradient of distance (e.g., springs)



R



$\hat{R}$

Gradient of distance to power α (e.g., gravity, bending)

Function: $g(p) = R^\alpha$ for $\alpha \neq 0$ & $R = \|p_k - p_\ell\|$.

from before

$\vec{R}$

Gradient:

$$\partial_i g = \partial_i R^\alpha = \alpha R^{\alpha-1} (\partial_i R)$$

$$= \alpha R^{\alpha-1} (\delta_{ik} - \delta_{i\ell}) \hat{R}$$

$$= \alpha (\delta_{ik} - \delta_{i\ell}) R^{\alpha-2} \vec{R}$$

$$= \alpha \|p_k - p_\ell\|^{\alpha-2} (p_k - p_\ell)(\delta_{ik} - \delta_{i\ell})$$

Gradient of unit distance vector (e.g., bending)

Function: $g(p) = \hat{R} = \frac{\vec{R}}{R} = \frac{(p_k - p_l)}{\|p_k - p_l\|}$

Gradient:

$$\partial_i g = \partial_i \hat{R} = \partial_i(\vec{R} \cdot R^{-1}) = (\partial_i \vec{R})\, R^{-1} + \vec{R}\,(\partial_i R^{-1})$$

$$= \left[(\delta_{ik} - \delta_{il})\, I_3\right] R^{-1} + \vec{R}\left[-\frac{(\delta_{ik} - \delta_{il})}{R^2}\, \hat{R}^T\right]$$

$$= \frac{(\delta_{ik} - \delta_{il})}{R}\left[I_3 - \hat{R}\,\hat{R}^T\right]$$

(a rank-2 matrix in 3D)

Some deformation forces

Zero rest-length spring between two particles

Energy: $E(p) = \frac{1}{2} k \| p_k - p_\ell \|^2$

Force on particle i:

from before

$$f_i = -\partial_i E = -\frac{1}{2} k \left(\partial_i \| p_k - p_\ell \|^2 \right)$$

$$= -\frac{1}{2} k \left[2 \left(\delta_{ik} - \delta_{i\ell} \right) \left(p_k - p_\ell \right) \right]$$

$$= -k \left(\delta_{ik} - \delta_{i\ell} \right) \left(p_k - p_\ell \right)$$

$$\Longrightarrow f_k = -f_\ell \quad \ddot\smile$$

Linear spring with finite rest-length, L

Energy: $E(p) = \frac{1}{2} k \left(\underbrace{\| p_k - p_\ell \|}_{R} - L \right)^2$

Force on particle i:

$$f_i = -\partial_i E = -\frac{1}{2} k \left(\partial_i (R-L)^2 \right) = -\frac{1}{2} k \left(2 (R-L) \overbrace{\partial_i R}^{\text{from before}} \right)$$

$$= -k (\delta_{ik} - \delta_{i\ell})(R-L)\hat{R}$$

$$= -k (\delta_{ik} - \delta_{i\ell})(\| p_k - p_\ell \| - L) \frac{(p_k - p_\ell)}{\| p_k - p_\ell \|}$$

$$\Longrightarrow f_k = -f_\ell$$

Damped linear spring with finite rest-length, L

Recall generalized damped spring force:

$$f = -J^T(k_s C + k_d \dot{C})$$

By inspection, we have

$$f_i = -(\delta_{ik} - \delta_{il})\,\hat{R}_{kl}\left\{ k_s \overbrace{(R_{kl} - L)}^{C} + k_c \overbrace{\dot{R}_{kl}}^{\dot{C}} \right\}$$

where

$$\dot{R}_{kl} = \frac{d}{dt} R_{kl}(p) = \sum_{i=k,l} \partial_i R_{kl}(p) \cdot \frac{dp_i}{dt}$$

($\frac{dp_i}{dt} = v_i$)

$$= \sum_{i=k,l} (\delta_{ik} - \delta_{il})\,\hat{R}_{kl} \cdot v_i = (v_k - v_l) \cdot \hat{R}_{kl} = \left(\begin{array}{c}\text{relative velocity along} \\ \text{spring}\end{array}\right)$$

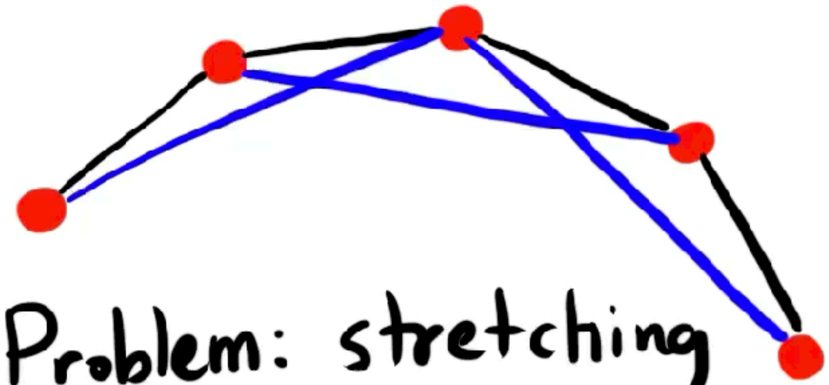
Hair bending forces

"Sulley" [Monsters, Inc., Pixar]

Hair bending force (flat rest shape)

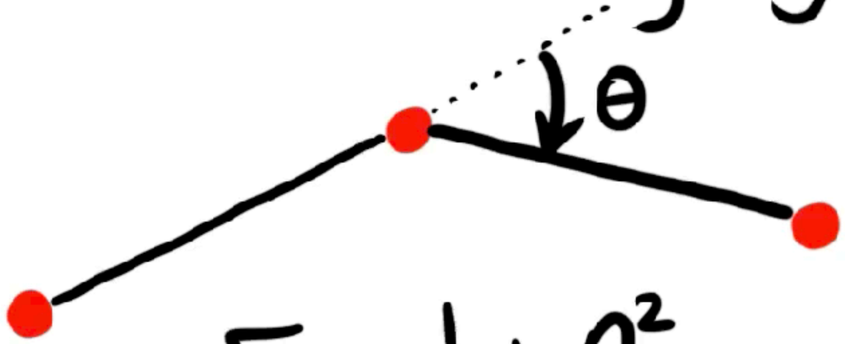
Bending models.

Simple but hacky:
add linear springs.



Problem: stretching and bending forces fight.

Better: Penalize bending angles.



$$E = \frac{1}{2} k \theta^2 \quad \text{per joint}$$

Easier to differentiate

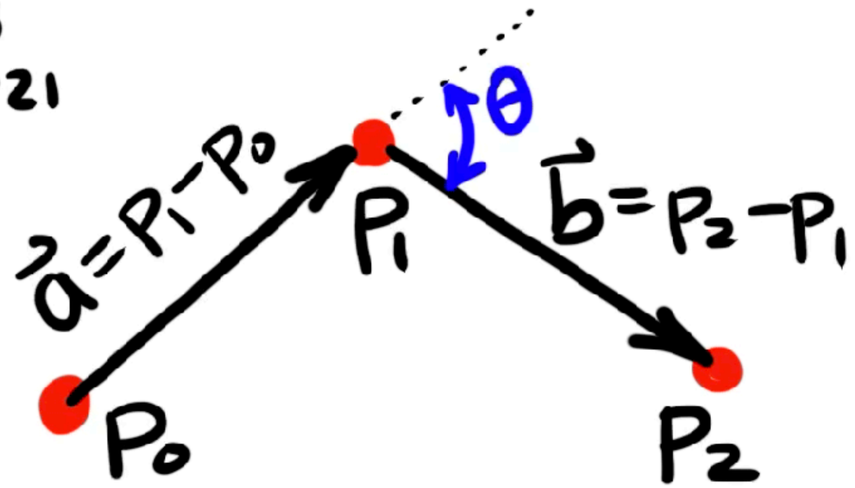
$$k \sin^2 \frac{\theta}{2} = \frac{k}{2}(1 - \cos\theta)$$

or just $$E = -\frac{k}{2} \cos\theta$$

Hair bending force (flat rest shape)

$$E = -\frac{k}{2}\,\hat{a}\cdot\hat{b} = -\frac{k}{2}\cos\theta = -\frac{k}{2}\,\hat{R}_{10}\cdot\hat{R}_{21}$$

$$f_i = -\partial_i E$$



Hair bending force (flat rest shape)

- **Symbolic differentiation (Mathematica). Easy & correct, but cumbersome expressions.**

```
In[22]:= Rhat[x_, y_] := {x, y} / Sqrt[x^2 + y^2];

In[23]:= Energy = -(k/2) Dot[Rhat[x1 - x0, y1 - y0], Rhat[x2 - x1, y2 - y1]]
```

$$\text{Out[23]=} \quad -\frac{1}{2}\,k\left(\frac{(-x0+x1)\,(-x1+x2)}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\sqrt{(-x1+x2)^2+(-y1+y2)^2}}+\frac{(-y0+y1)\,(-y1+y2)}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\sqrt{(-x1+x2)^2+(-y1+y2)^2}}\right)$$

```
In[24]:= f0 = D[Energy, {{x0, y0}}]
```

$$\text{Out[24]=} \quad \left\{-\frac{1}{2}\,k\left(\frac{(-x0+x1)^2\,(-x1+x2)}{\left((-x0+x1)^2+(-y0+y1)^2\right)^{3/2}\sqrt{(-x1+x2)^2+(-y1+y2)^2}}-\frac{-x1+x2}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\sqrt{(-x1+x2)^2+(-y1+y2)^2}}+\frac{(-x0+x1)\,(-y0+y1)\,(-y1+y2)}{\left((-x0+x1)^2+(-y0+y1)^2\right)^{3/2}\sqrt{(-x1+x2)^2+(-y1+y2)^2}}\right),\right.$$
$$\left.-\frac{1}{2}\,k\left(\frac{(-x0+x1)\,(-x1+x2)\,(-y0+y1)}{\left((-x0+x1)^2+(-y0+y1)^2\right)^{3/2}\sqrt{(-x1+x2)^2+(-y1+y2)^2}}+\frac{(-y0+y1)^2\,(-y1+y2)}{\left((-x0+x1)^2+(-y0+y1)^2\right)^{3/2}\sqrt{(-x1+x2)^2+(-y1+y2)^2}}-\frac{-y1+y2}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\sqrt{(-x1+x2)^2+(-y1+y2)^2}}\right)\right\}$$

```
In[25]:= f1 = D[Energy, {{x1, y1}}]
```

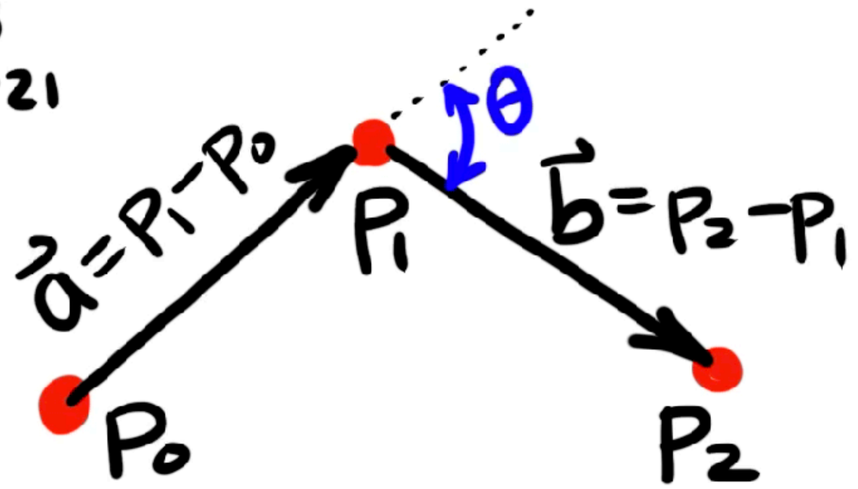
$$\text{Out[25]=} \quad \left\{-\frac{1}{2}\,k\left(\frac{(-x0+x1)\,(-x1+x2)^2}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\left((-x1+x2)^2+(-y1+y2)^2\right)^{3/2}}+\frac{(-x1+x2)\,(-y0+y1)\,(-y1+y2)}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\left((-x1+x2)^2+(-y1+y2)^2\right)^{3/2}}-\frac{(-x0+x1)^2\,(-x1+x2)}{\left((-x0+x1)^2+(-y0+y1)^2\right)^{3/2}\sqrt{(-x1+x2)^2+(-y1+y2)^2}}-\right.$$
$$\left.\frac{-x0+x1}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\sqrt{(-x1+x2)^2+(-y1+y2)^2}}+\frac{-x1+x2}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\sqrt{(-x1+x2)^2+(-y1+y2)^2}}-\frac{(-x0+x1)\,(-y0+y1)\,(-y1+y2)}{\left((-x0+x1)^2+(-y0+y1)^2\right)^{3/2}\sqrt{(-x1+x2)^2+(-y1+y2)^2}}\right),$$
$$-\frac{1}{2}\,k\left(\frac{(-x0+x1)\,(-x1+x2)\,(-y1+y2)}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\left((-x1+x2)^2+(-y1+y2)^2\right)^{3/2}}+\frac{(-y0+y1)\,(-y1+y2)^2}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\left((-x1+x2)^2+(-y1+y2)^2\right)^{3/2}}-\frac{(-x0+x1)\,(-x1+x2)\,(-y0+y1)}{\left((-x0+x1)^2+(-y0+y1)^2\right)^{3/2}\sqrt{(-x1+x2)^2+(-y1+y2)^2}}-\right.$$
$$\left.\frac{-y0+y1}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\sqrt{(-x1+x2)^2+(-y1+y2)^2}}-\frac{(-y0+y1)^2\,(-y1+y2)}{\left((-x0+x1)^2+(-y0+y1)^2\right)^{3/2}\sqrt{(-x1+x2)^2+(-y1+y2)^2}}+\frac{-y1+y2}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\sqrt{(-x1+x2)^2+(-y1+y2)^2}}\right)\right\}$$

```
In[26]:= f2 = D[Energy, {{x2, y2}}]
```

$$\text{Out[26]=} \quad \left\{-\frac{1}{2}\,k\left(-\frac{(-x0+x1)\,(-x1+x2)^2}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\left((-x1+x2)^2+(-y1+y2)^2\right)^{3/2}}-\frac{(-x1+x2)\,(-y0+y1)\,(-y1+y2)}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\left((-x1+x2)^2+(-y1+y2)^2\right)^{3/2}}+\frac{-x0+x1}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\sqrt{(-x1+x2)^2+(-y1+y2)^2}}\right),\right.$$
$$\left.-\frac{1}{2}\,k\left(-\frac{(-x0+x1)\,(-x1+x2)\,(-y1+y2)}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\left((-x1+x2)^2+(-y1+y2)^2\right)^{3/2}}-\frac{(-y0+y1)\,(-y1+y2)^2}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\left((-x1+x2)^2+(-y1+y2)^2\right)^{3/2}}+\frac{-y0+y1}{\sqrt{(-x0+x1)^2+(-y0+y1)^2}\,\sqrt{(-x1+x2)^2+(-y1+y2)^2}}\right)\right\}$$

Hair bending force (flat rest shape)

$$E = -\frac{k}{2}\,\hat{a}\cdot\hat{b} = -\frac{k}{2}\cos\theta = -\frac{k}{2}\,\hat{R}_{10}\cdot\hat{R}_{21}$$



$$f_i = -\partial_i E = +\frac{k}{2}\,\partial_i(\hat{a}\cdot\hat{b})$$

$$= \frac{k}{2}\left[(\partial_i\hat{a})\cdot\hat{b} + \hat{a}\cdot(\partial_i\hat{b})\right]$$

recall $\partial_i \hat{R}_{kl} = \dfrac{(\delta_{ik}-\delta_{il})}{R_{kl}}\left[I - \hat{R}_{kl}\hat{R}_{kl}^T\right]$

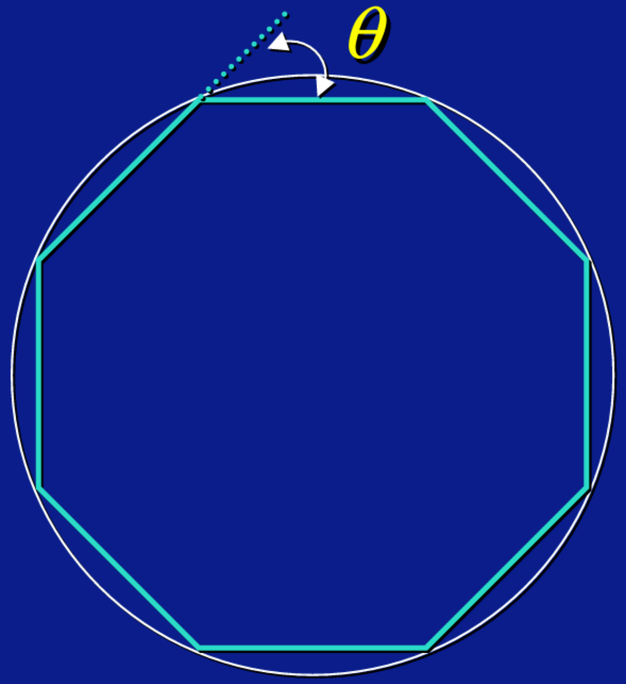
$$= \frac{k}{2}\left[\frac{(\delta_{i1}-\delta_{i0})}{a}\left(\hat{b}-\hat{a}(\hat{a}\cdot\hat{b})\right) + \frac{(\delta_{i2}-\delta_{i1})}{b}\left(\hat{a}-\hat{b}(\hat{b}\cdot\hat{a})\right)\right]$$

$$\Rightarrow f_0 = -\frac{k}{2a}\left(\hat{b}-\hat{a}(\hat{a}\cdot\hat{b})\right),\quad f_2 = \frac{k}{2b}\left(\hat{a}-\hat{b}(\hat{a}\cdot\hat{b})\right),\quad f_1 = -f_0 - f_2$$

Hair bending force (flat rest shape)

- Length-dependent bending stiffness
- Reference: Kass

Discretization



Consider a discretized circle.

$$E = \frac{1}{2} k \sum \theta^2$$

Again, constant energy implies:

$k \propto n$ or $k_i \propto \frac{1}{l_i}$

That's all for now!

"Sulley" [Monsters, Inc., Pixar]