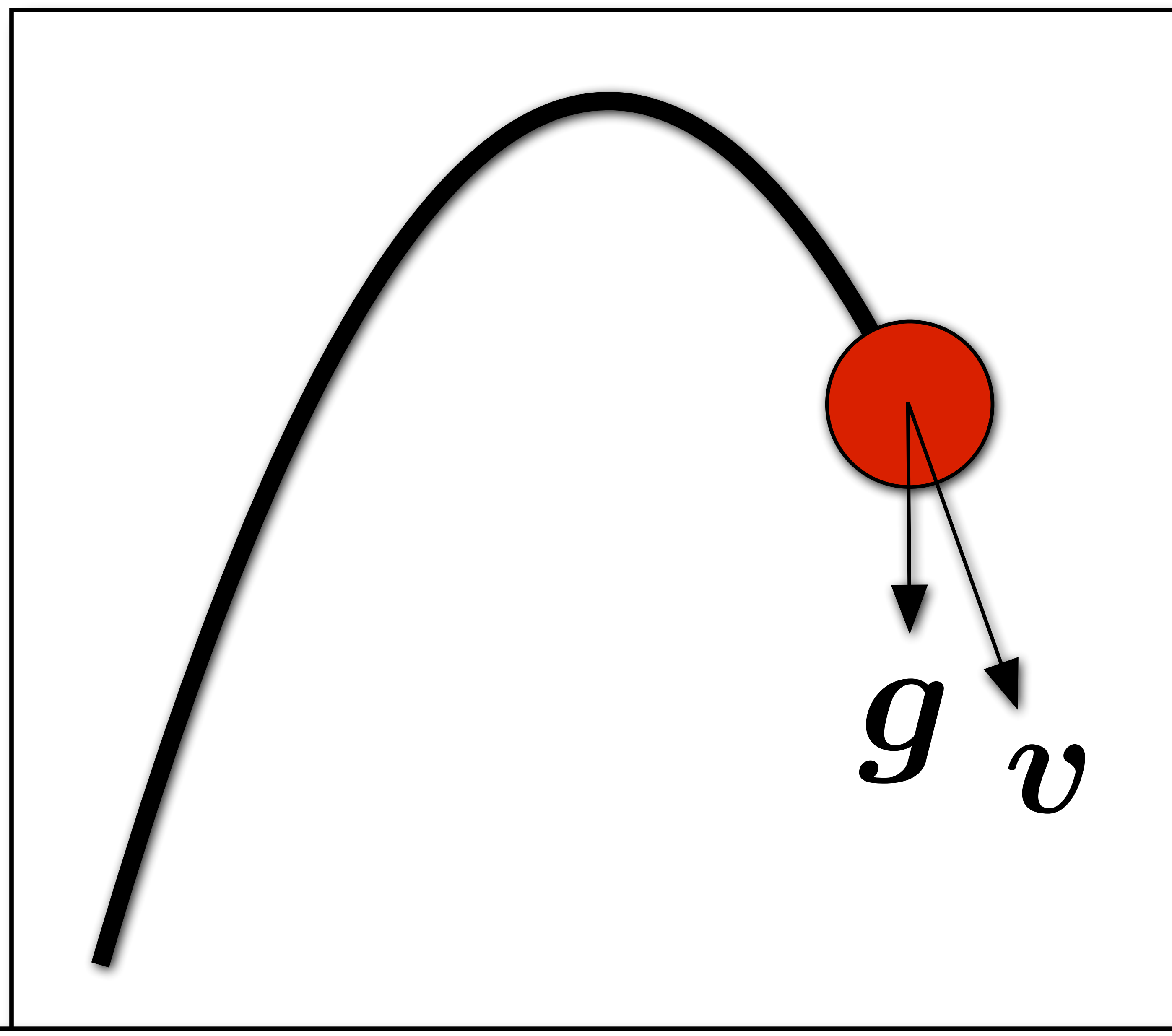
Lecture 19:

Introduction to Physical Simulation

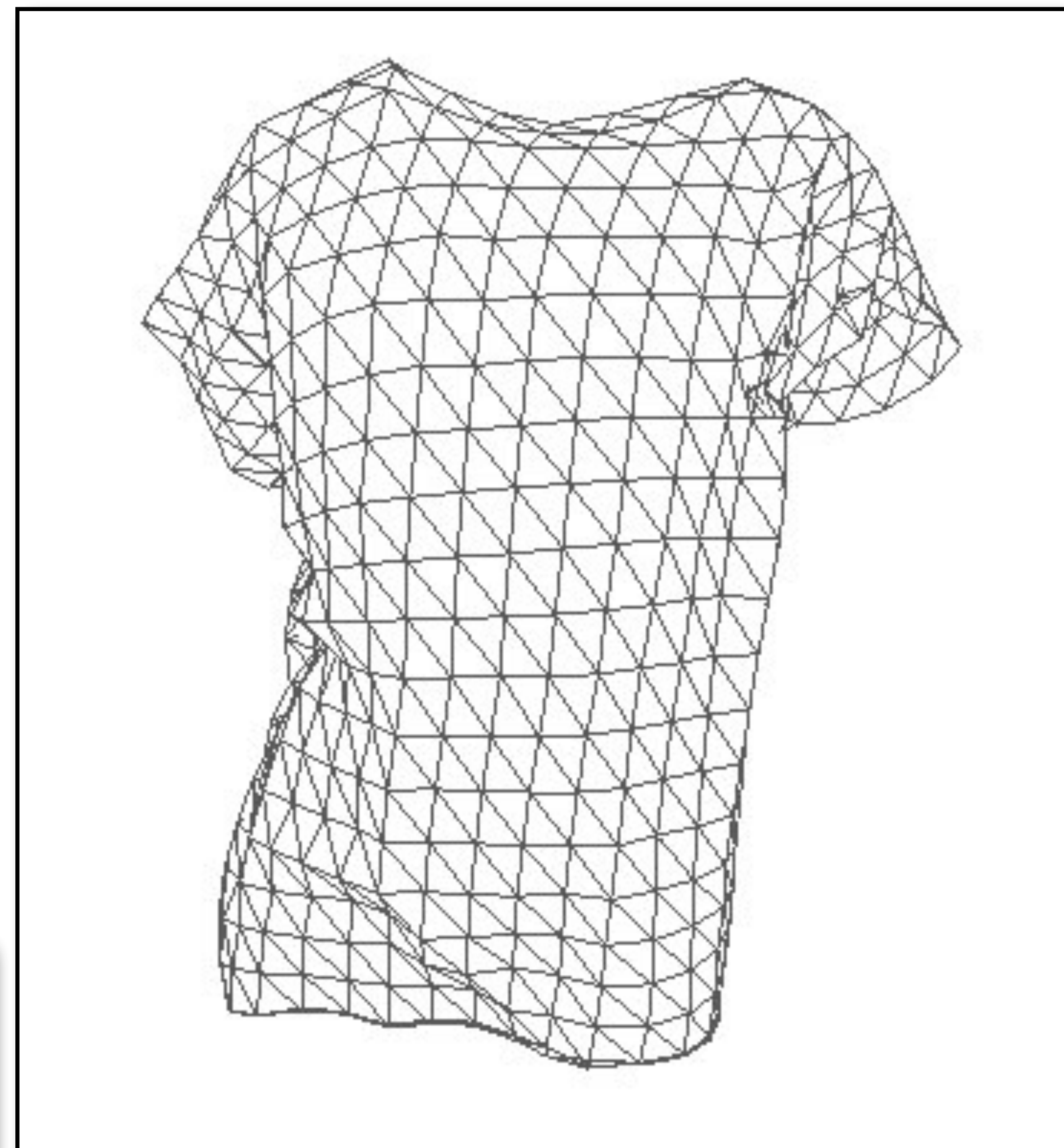
Computer Graphics and Imaging
UC Berkeley CS184/284A

Physically Based Animation

Generate motion of objects using numerical simulation



$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t \boldsymbol{v}^t + \frac{1}{2}(\Delta t)^2 \boldsymbol{a}^t$$



Example: Cloth Simulation

Rahul Narain, Armin Samii, James F. O'Brien
SIGGRAPH Asia 2012

Example: Fluids

Macklin and Müller, Position Based Fluids TOG 2013

Example: Fracture

Tobias Pfaff, Rahul Narain, Juan Miguel de Joya, James F. O'Brien
SIGGRAPH 2014

Particle Systems

Single particles are very simple

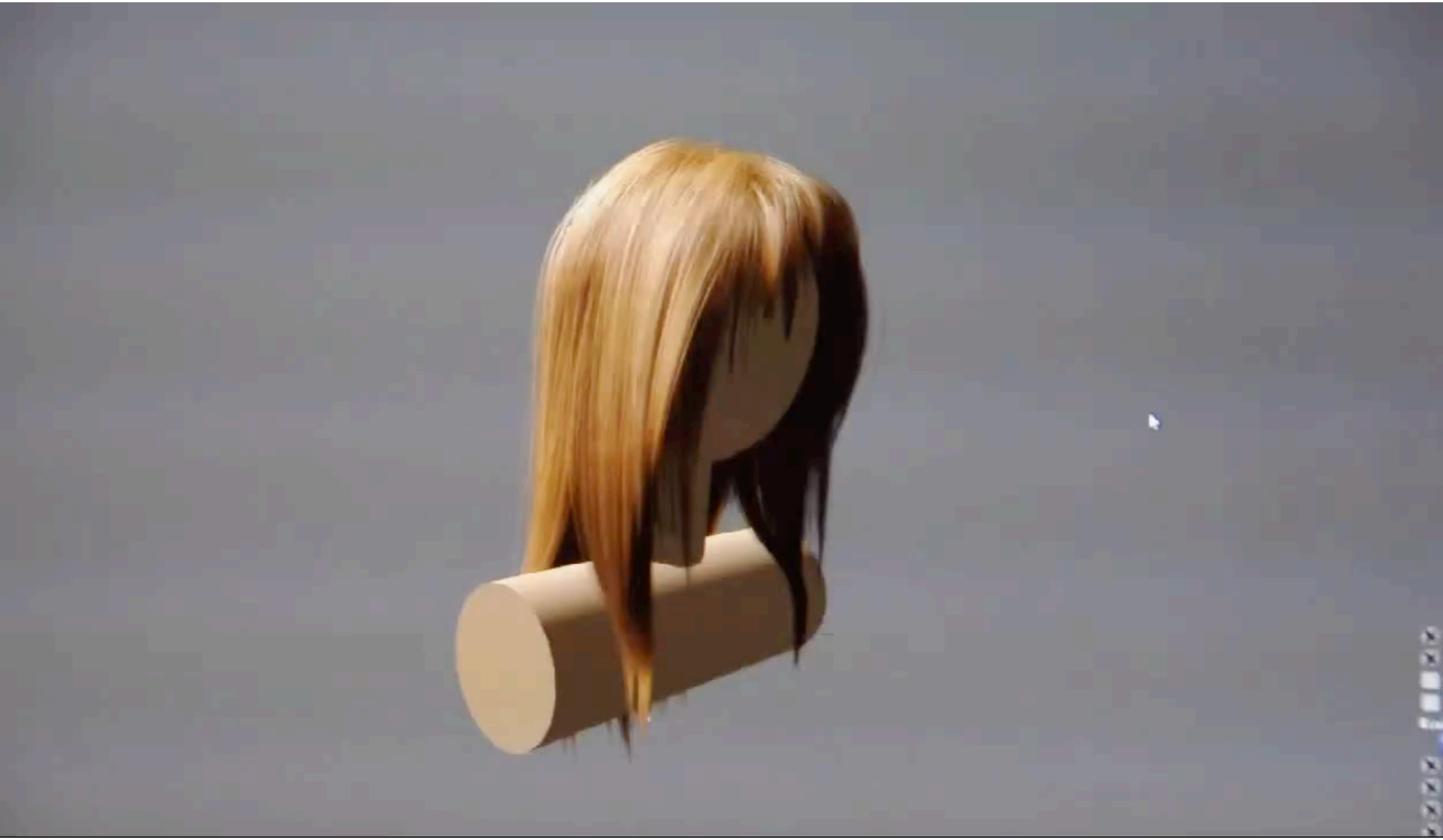
Large groups can produce interesting effects

Supplement basic ballistic rules

- Gravity
- Friction, drag
- Collisions
- Force fields
- Springs
- Interactions

Karl Sims, SIGGRAPH 1990

Example: Hair



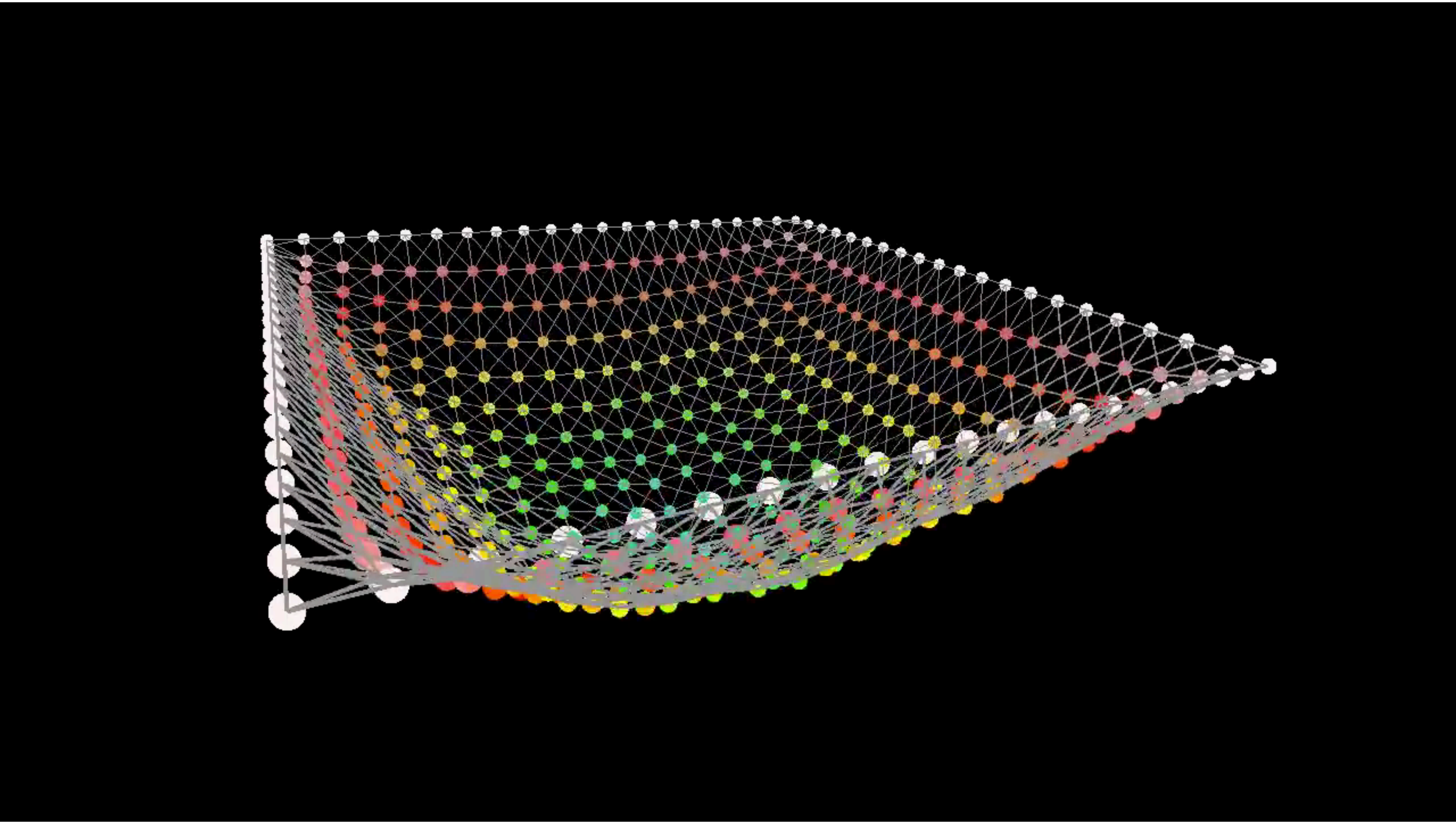
Example: Adaptive Simulation

Mass + Spring Systems

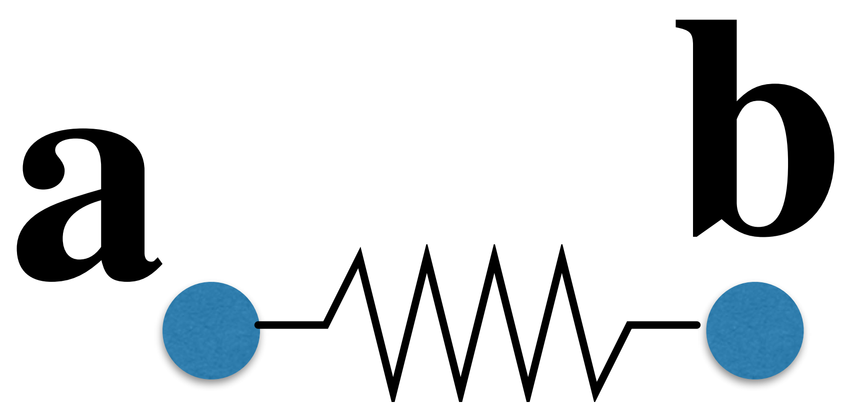
Example: Mass Spring Rope

Credit: Elizabeth Labelle, https://youtu.be/Co8enp8CH34

Example: Mass Spring Mesh



A Simple Spring



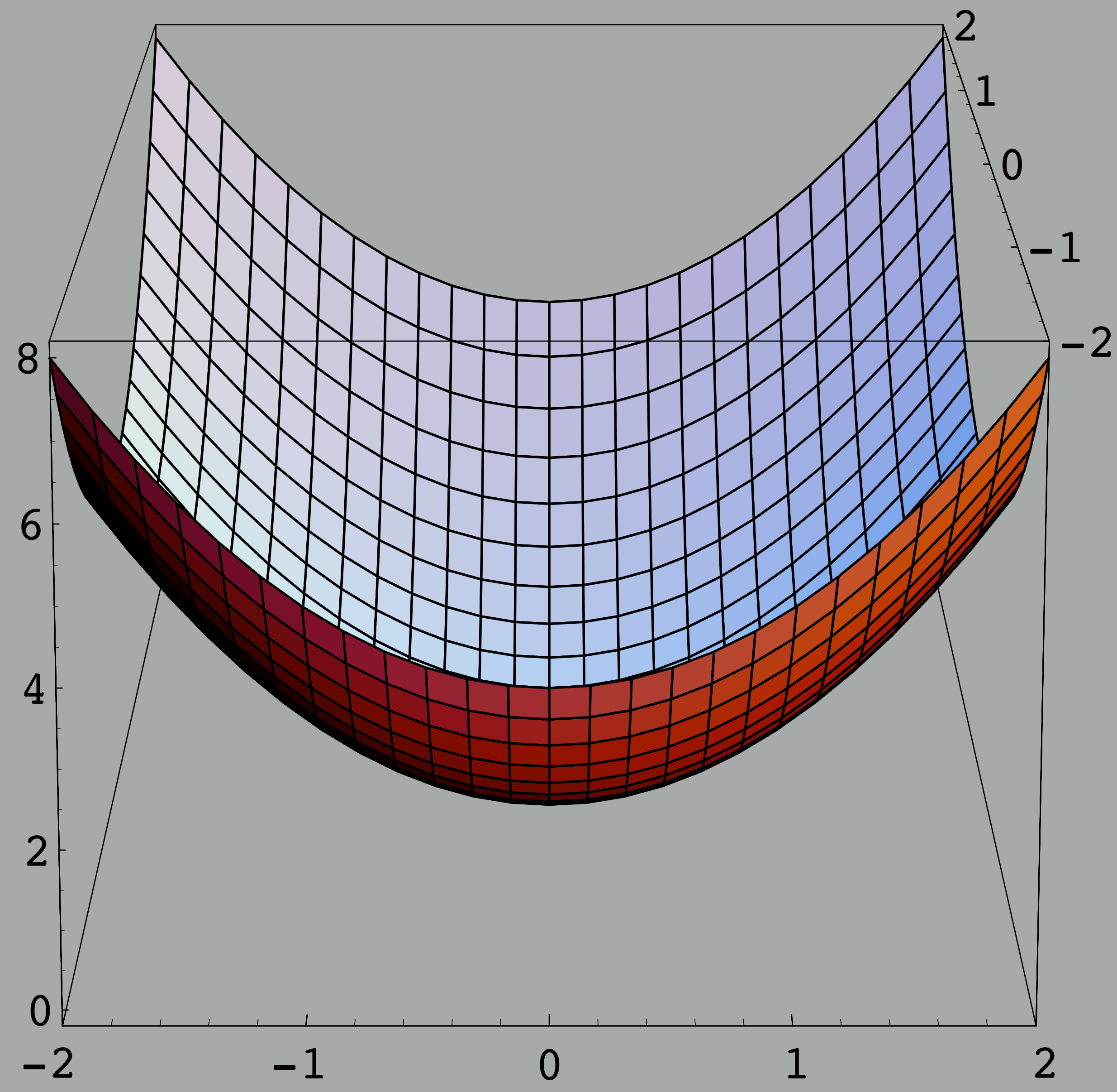
$$\boldsymbol{f}_{a \to b} = k_s(\boldsymbol{b} - \boldsymbol{a})$$

$$\boldsymbol{f}_{b \to a} = -\boldsymbol{f}_{a \to b}$$

Problem: this spring wants to have zero length

A Simple Spring

Energy potential

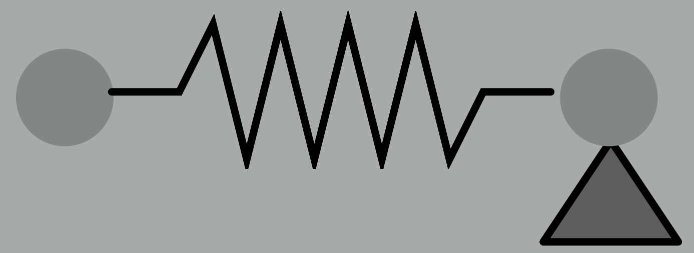


$$E = 1/2\, k_s(\boldsymbol{b} - \boldsymbol{a}) \cdot (\boldsymbol{b} - \boldsymbol{a})$$

$$\boldsymbol{f}_{a \to b} = k_s(\boldsymbol{b} - \boldsymbol{a})$$

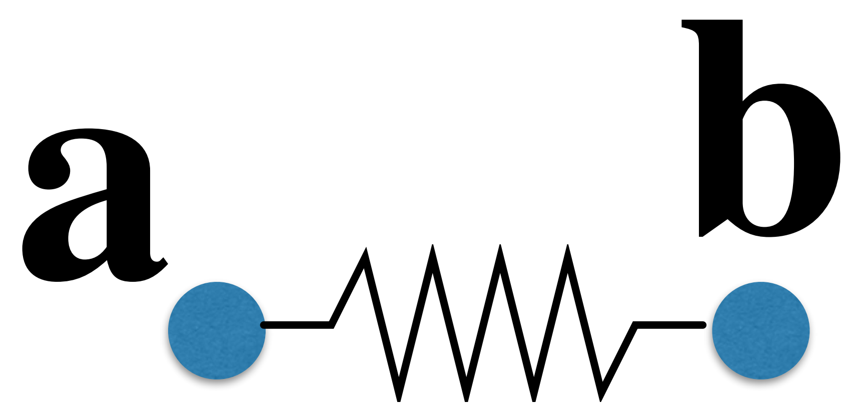
$$\boldsymbol{f}_{b \to a} = -\boldsymbol{f}_{a \to b}$$

$$\boldsymbol{f}_a = -\nabla_a E = -\left[\frac{\partial E}{\partial a_x}, \frac{\partial E}{\partial a_y}, \frac{\partial E}{\partial a_z}\right]$$



Ren Ng, James O'Brien

Non-Zero Length Spring



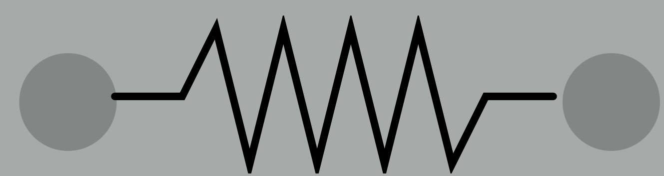
$$\boldsymbol{f}_{a\rightarrow b} = k_s \frac{\boldsymbol{b}-\boldsymbol{a}}{||\boldsymbol{b}-\boldsymbol{a}||}(||\boldsymbol{b}-\boldsymbol{a}|| - l)$$

Rest length

Problem: oscillates forever

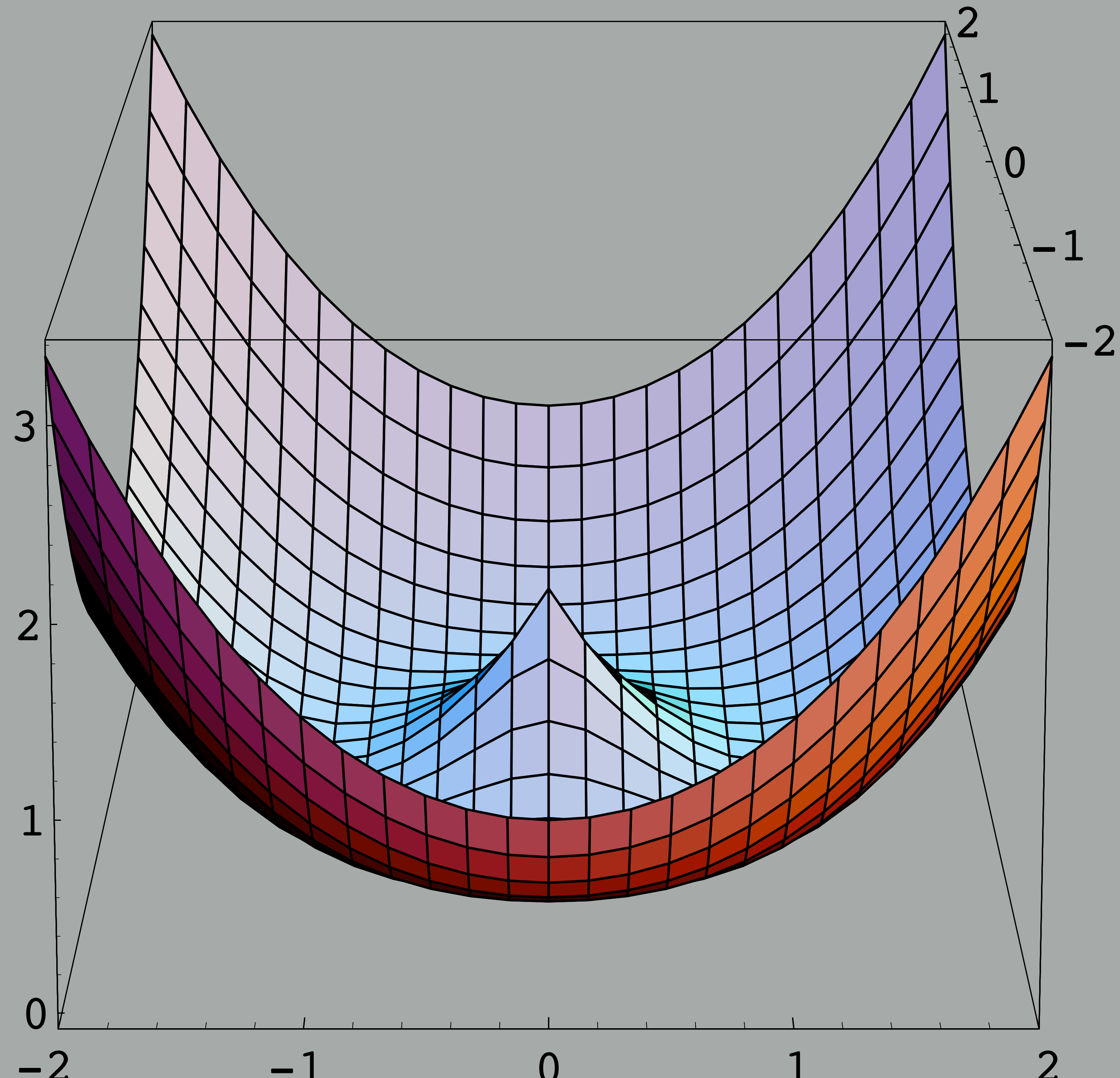
Extra material for 284A

Non-Zero Length Springs



$$\boldsymbol{f}_{a\rightarrow b} = k_S \frac{\boldsymbol{b}-\boldsymbol{a}}{||\boldsymbol{b}-\boldsymbol{a}||}\left(||\boldsymbol{b}-\boldsymbol{a}|| - l\right)$$

Rest length



$$E = k_S\left(||\boldsymbol{b}-\boldsymbol{a}|| - l\right)^2$$

Comments on Springs

Springs with zero rest length are linear

Springs with non-zero rest length are nonlinear

- Force *magnitude* linear w/ displacement (from rest length)
- **Force direction is non-linear**
- **Singularity at** $||\boldsymbol{b} - \boldsymbol{a}|| = 0$

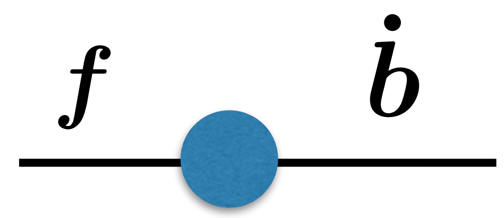
Dot Notation for Derivatives

A dot above a variable indicated taking a derivative w.r.t. time. For example:

- $\mathbf{x}$ might be a variable indicating the position of something
- $\dot{\mathbf{x}}$ would be $\dfrac{d\mathbf{x}}{dt}$, *i.e.*, velocity
- $\ddot{\mathbf{x}}$ would be $\dfrac{d^2\mathbf{x}}{dt^2}$, *i.e.*, acceleration

Simple Motion Damping

Simple motion damping



$$\boldsymbol{f} = -k_d \dot{\boldsymbol{b}}$$

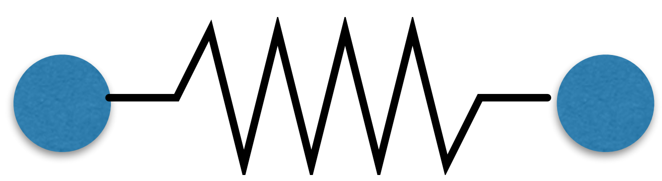
- Behaves like viscous drag on motion
- Slows down motion in the direction of motion
- k_d is a damping coefficient
- "Mass-proportional" damping

Problem: slows down *all* motion

- Want a rusty spring's oscillations to slow down, but should it also fall to the ground more slowly?

Internal Damping for Spring

Damp only the internal, spring-driven motion



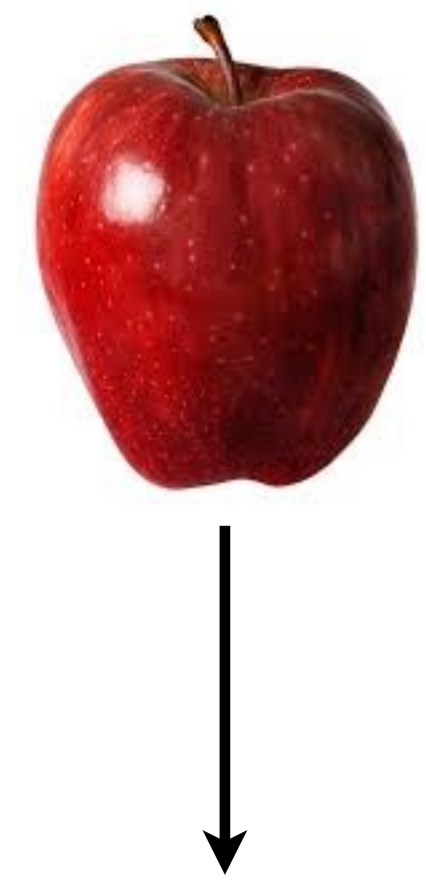
$$\boldsymbol{f}_a = -k_d \frac{\boldsymbol{b}-\boldsymbol{a}}{||\boldsymbol{b}-\boldsymbol{a}||}(\dot{\boldsymbol{b}}-\dot{\boldsymbol{a}}) \cdot \frac{\boldsymbol{b}-\boldsymbol{a}}{||\boldsymbol{b}-\boldsymbol{a}||}$$

- Viscous drag only on change in spring length
  - Won't slow group motion for the spring system (e.g. global translation or rotation of the group)
- "Stiffness proportional" damping

Gravity

Gravity at earth's surface due to earth

- F = –mg
- m is mass of object
- g is gravitational acceleration, g = –9.8m/s²



$$F_g = -mg$$

$$g = (0, 0, -9.8)\,\mathrm{m/s^2}$$

Standard Form

Nonlinear

$$\mathscr{K}(\mathbf{x}) + \mathscr{D}(\mathbf{x}, \dot{\mathbf{x}}) + \mathscr{M}(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}) = \mathbf{f}$$

Internal Elasticity **Damping** **Momentum** **External Forces**

Linearized

$$\mathbf{K}\,\mathbf{x} + \mathbf{D}\,\dot{\mathbf{x}} + \mathbf{M}\,\ddot{\mathbf{x}} = \mathbf{f}$$

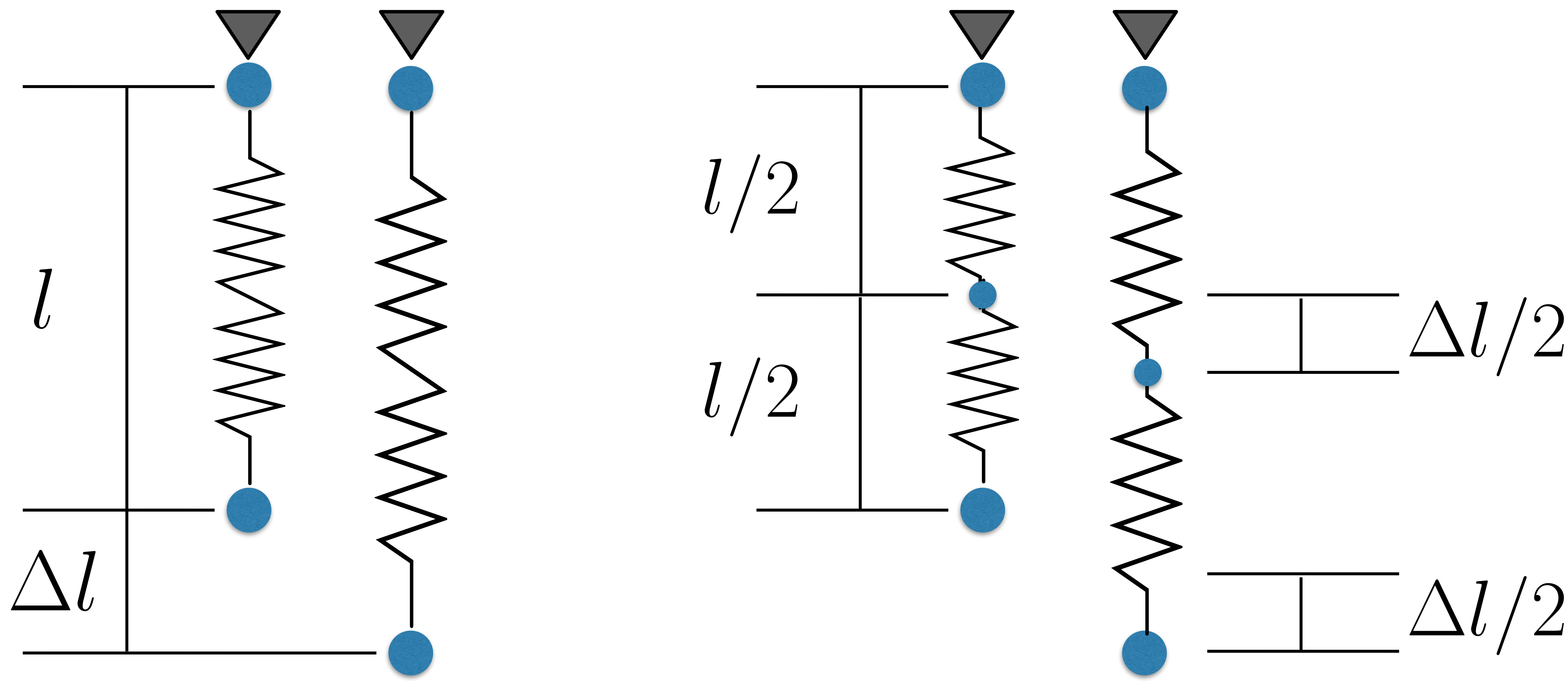
Zero-length springs result in constant $\mathbf{K}$ and $\mathbf{D}$

Typically $\mathbf{M}$ is constant

We can keep $\mathbf{M}$ diagonal by "lumping" called a *"Lumped* Mass Matrix"

Spring Constants

Consider two "resolutions" to model a single spring



Problem: constant k_s produces different force on bottom spring for these two different discretizations

Spring Constants

Problem: constant k_s gives inconsistent results with different discretizations of our spring/mass structures

- E.g. 10x10 vs 20x20 mesh for cloth simulation would give different results, and we want them to be the same, just higher level of detail

Solution:

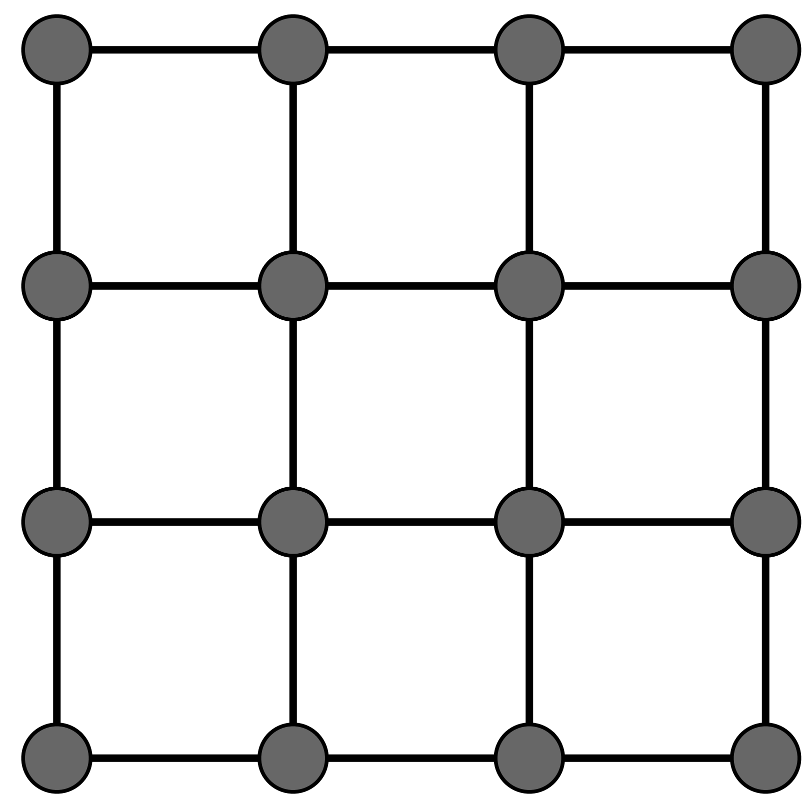
- Change in length is not what we want to measure
- We want to consider the strain = change in length as fraction of original length

$$\epsilon = \frac{\Delta l}{l_0}$$

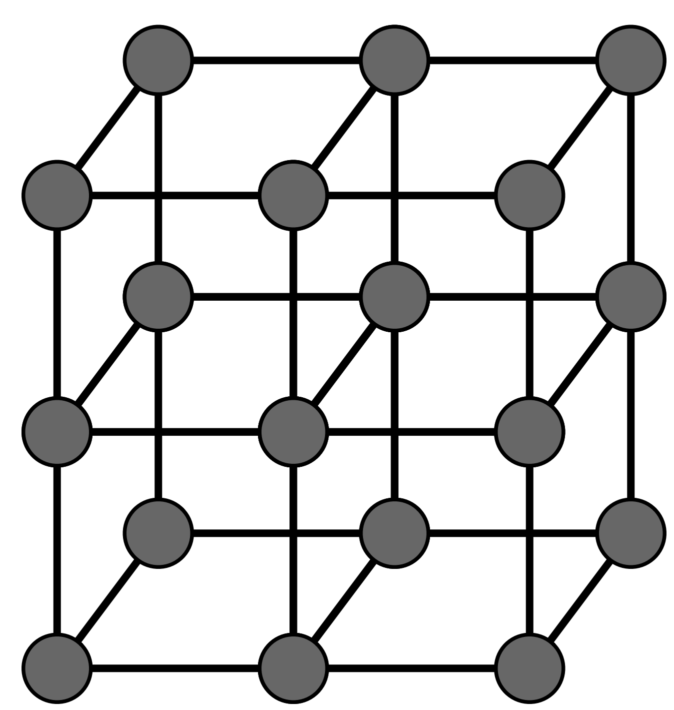
- Implementation 1: divide spring force by spring length
- Implementation 2: normalize k_s by spring length

Structures from Springs

Sheets



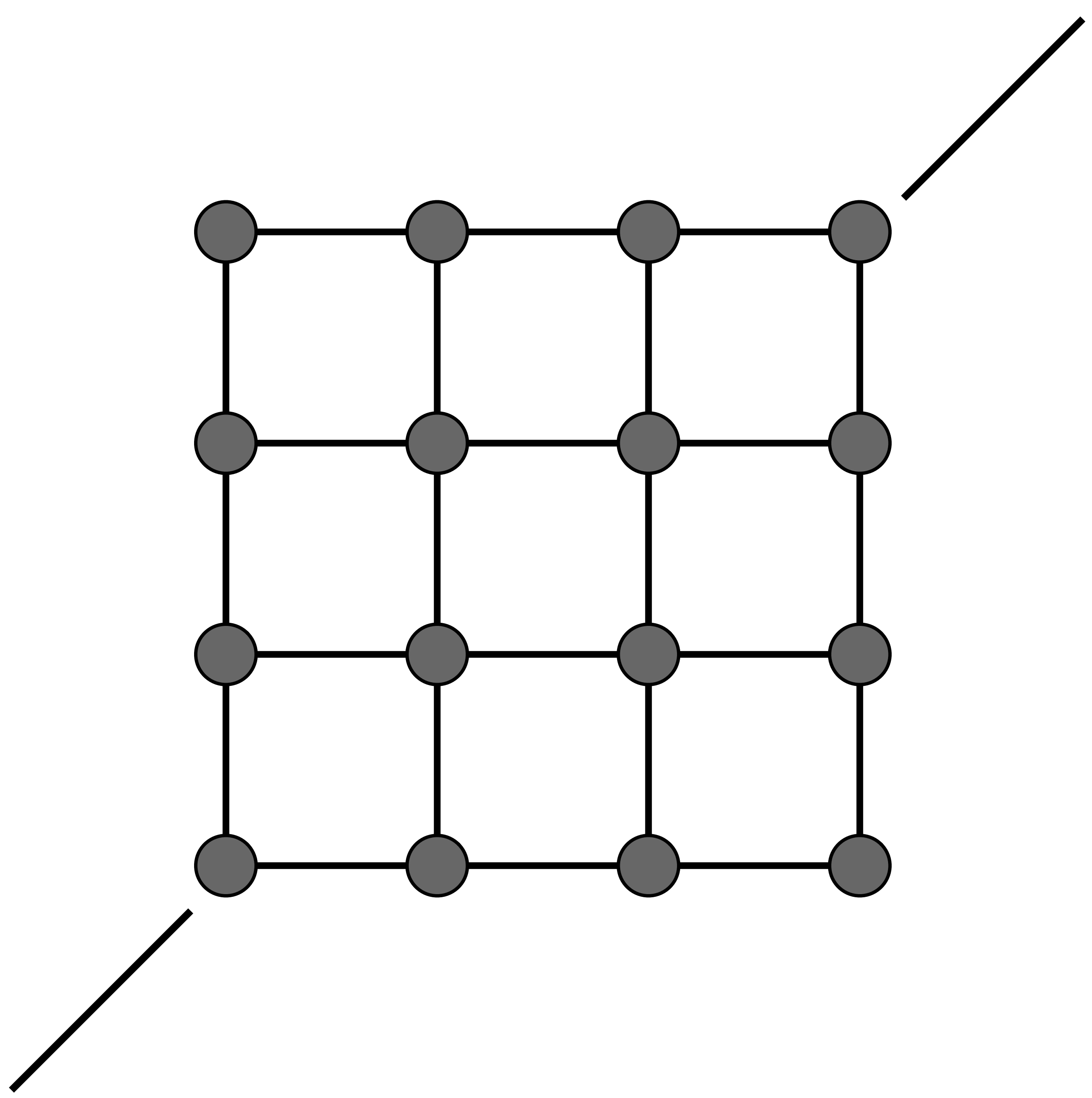
Blocks



Others

Structures from Springs

Behavior is determined by structure linkages

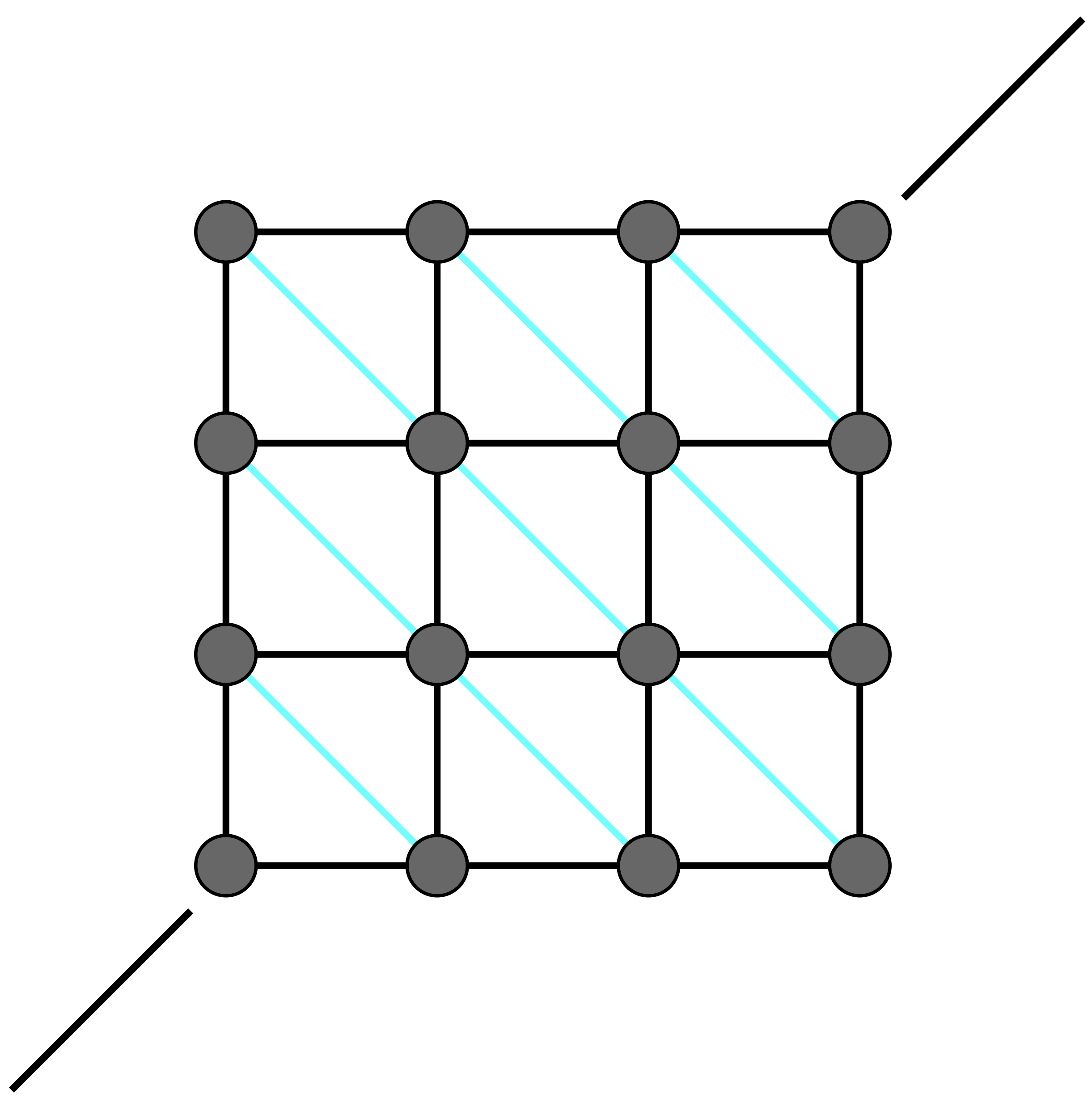


This structure will not resist shearing

This structure will not resist out-of-plane bending…

Structures from Springs

Behavior is determined by structure linkages

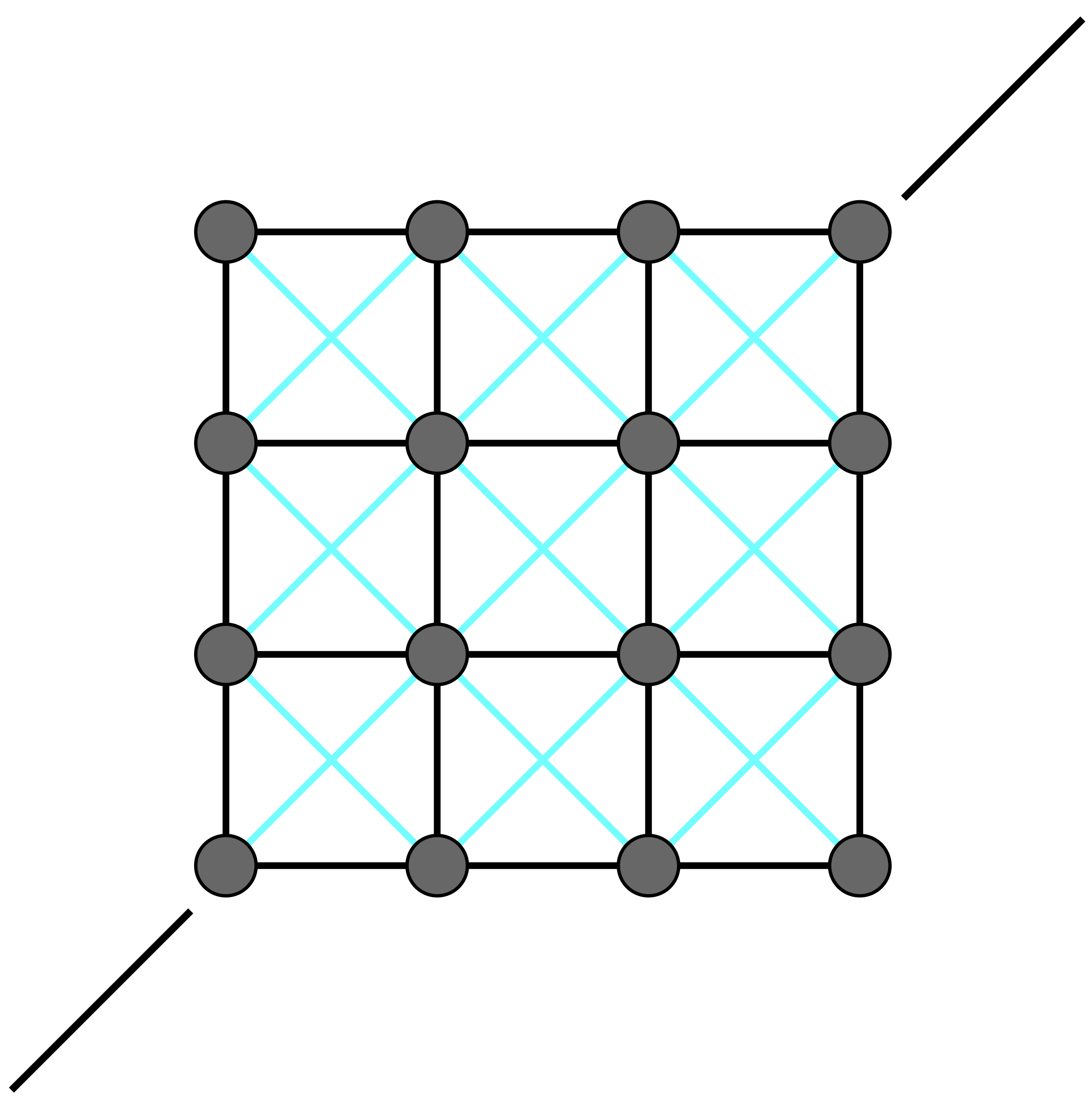


This structure will resist shearing but has anisotropic bias

This structure will not resist out-of-plane bending either...

Structures from Springs

Behavior is determined by structure linkages

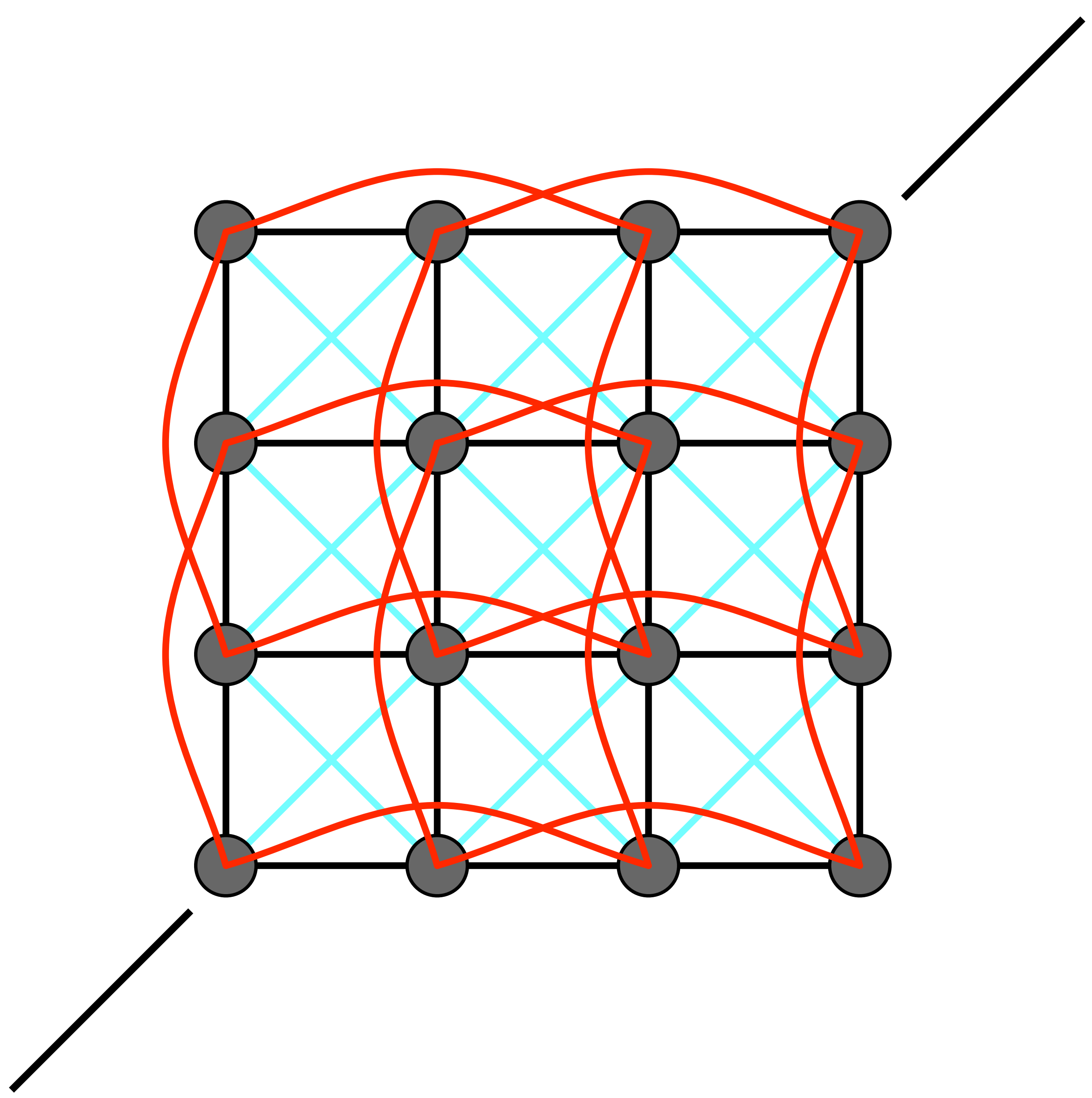


This structure will resist shearing.
Less directional bias.

This structure will not resist out-of-plane bending either...

Structures from Springs

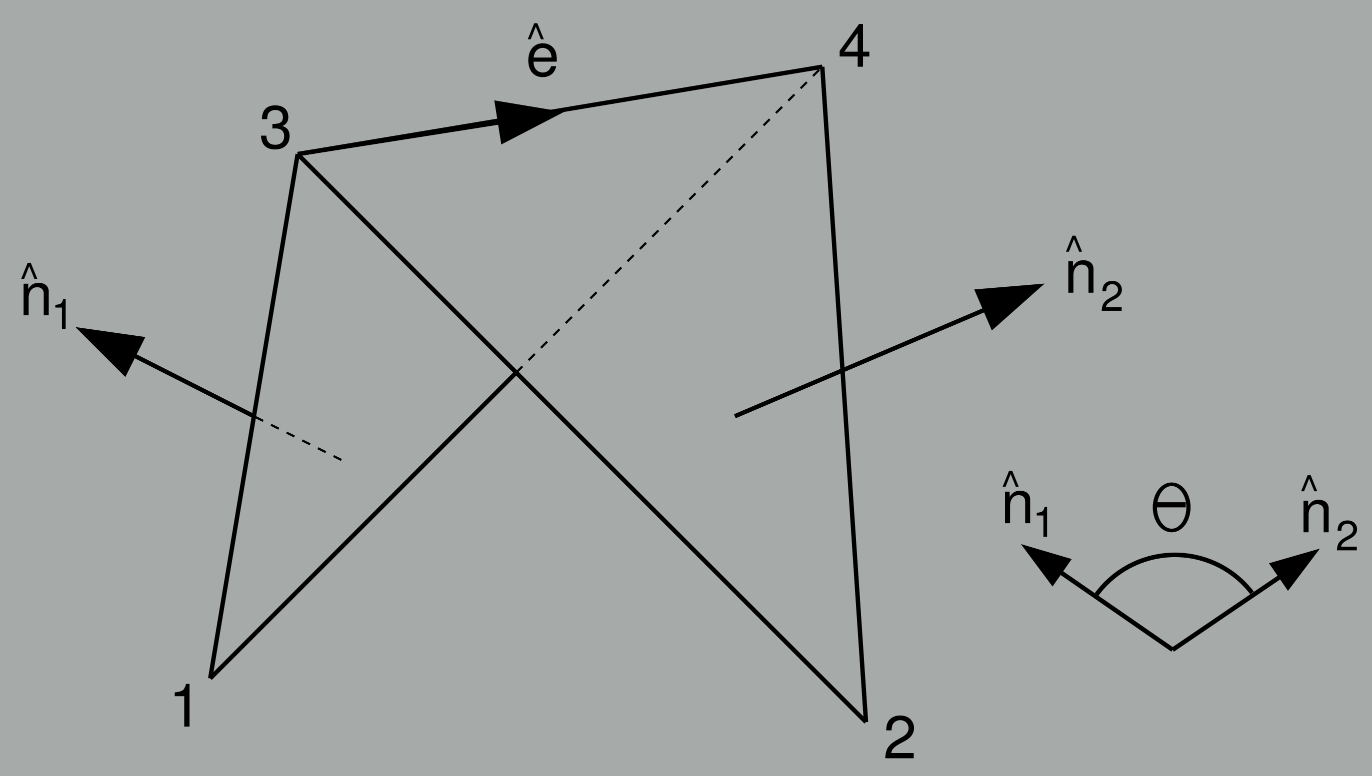
Spring structures will behave like what they are (obviously?)



This structure will resist shearing.
Less directional bias.

This structure will resist out-of-plane bending
Red springs should be much weaker

Edge Springs (bending)



$$u_1 = |E| \frac{N_1}{|N_1|^2} \qquad u_2 = |E| \frac{N_2}{|N_2|^2}$$

$$u_3 = \frac{(x_1 - x_4) \cdot E}{|E|} \frac{N_1}{|N_1|^2} + \frac{(x_2 - x_4) \cdot E}{|E|} \frac{N_2}{|N_2|^2}$$

$$u_4 = -\frac{(x_1 - x_3) \cdot E}{|E|} \frac{N_1}{|N_1|^2} - \frac{(x_2 - x_3) \cdot E}{|E|} \frac{N_2}{|N_2|^2}$$

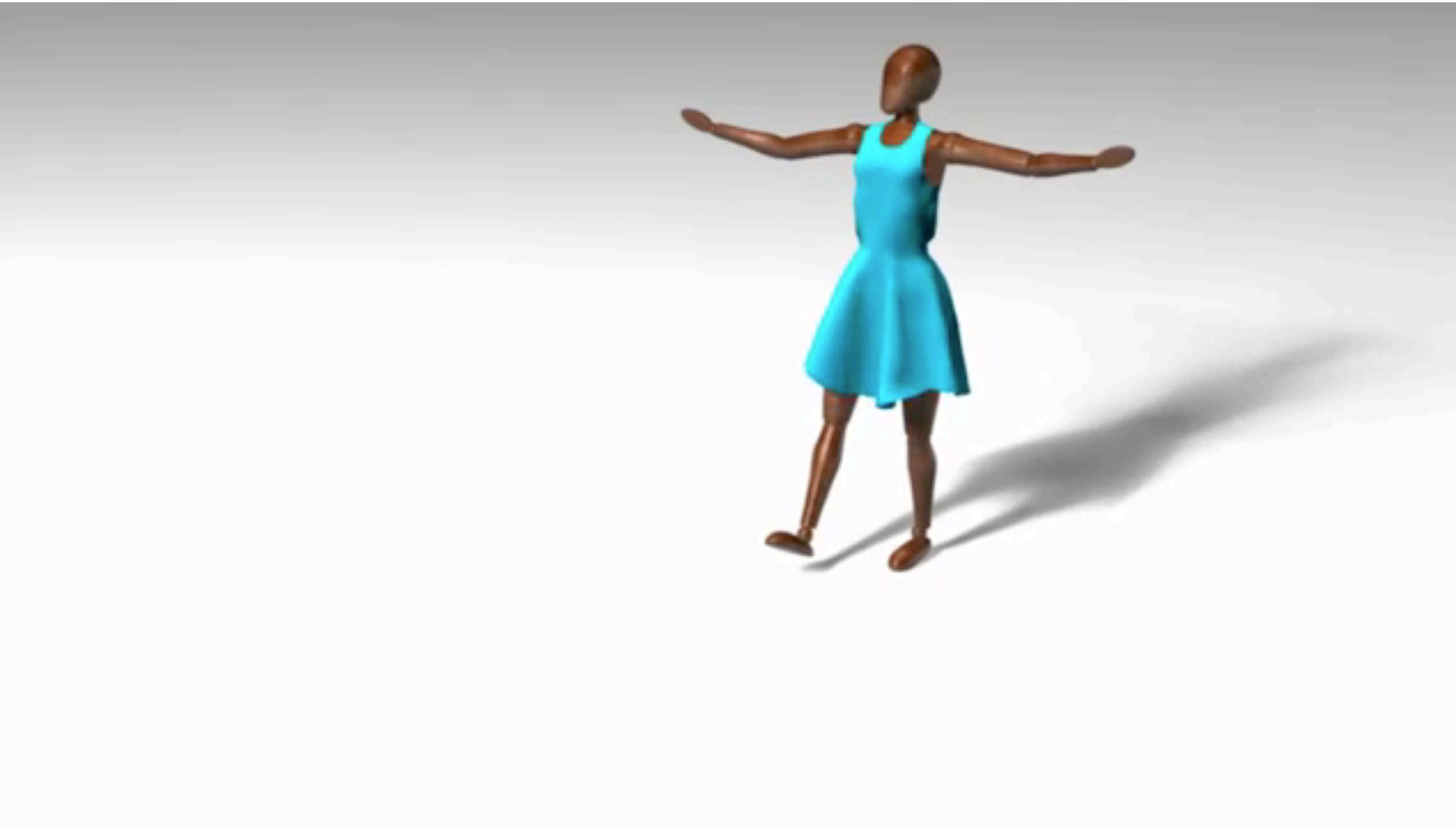
$$F_i^e = k^e \frac{|E|^2}{|N_1| + |N_2|} \sin(\theta/2)\, u_i$$

From Bridson *et al.*, 2003

Extra material for 284A

Bending Springs and Sharp Creases

Example: Mass Spring Dress + Character



FEM (Finite Element Method) Instead of Springs

FEM: Variety of Materials

More Accurate Materials

Linear force-displacement (stress-strain) relationship limiting

- Bi-phasic materials, *e.g.:* cloth, biological tissues, *etc.*
- Other nonlinear material behaviors

One-dimensional strain doesn't capture everything

- Anisotropic materials
- Volume-preserving
- Interaction between directional behaviors
  - Spring coupling is *ad hoc* and undesirable

Solution: non-linear FEM

- Not much harder to implement than springs!

Basic FEM

Extra material for 284A

FEM Problem Setup

Lagrangian Formulation

- Where in space did this material mode to?
- Commonly used for solid materials

Eulerian Formulation

- What material is at this location in space?
- Commonly used for fluids

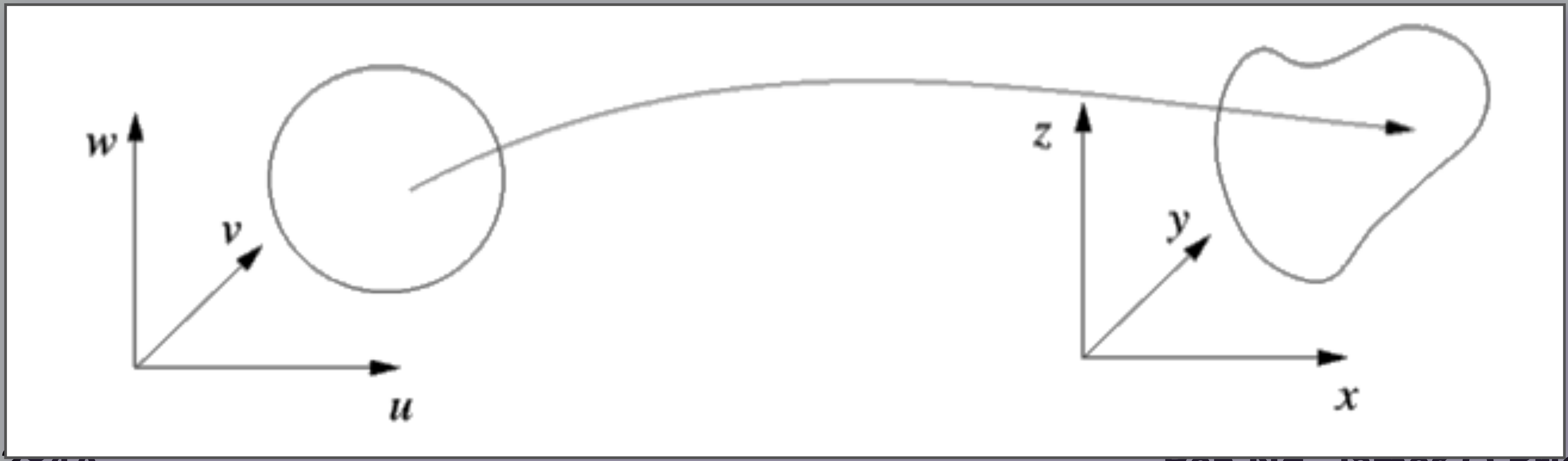
Extra material for 284A

Problem Setup

Lagrangian Formulation

- Where in space did this material mode to?
- Commonly used for solid materials

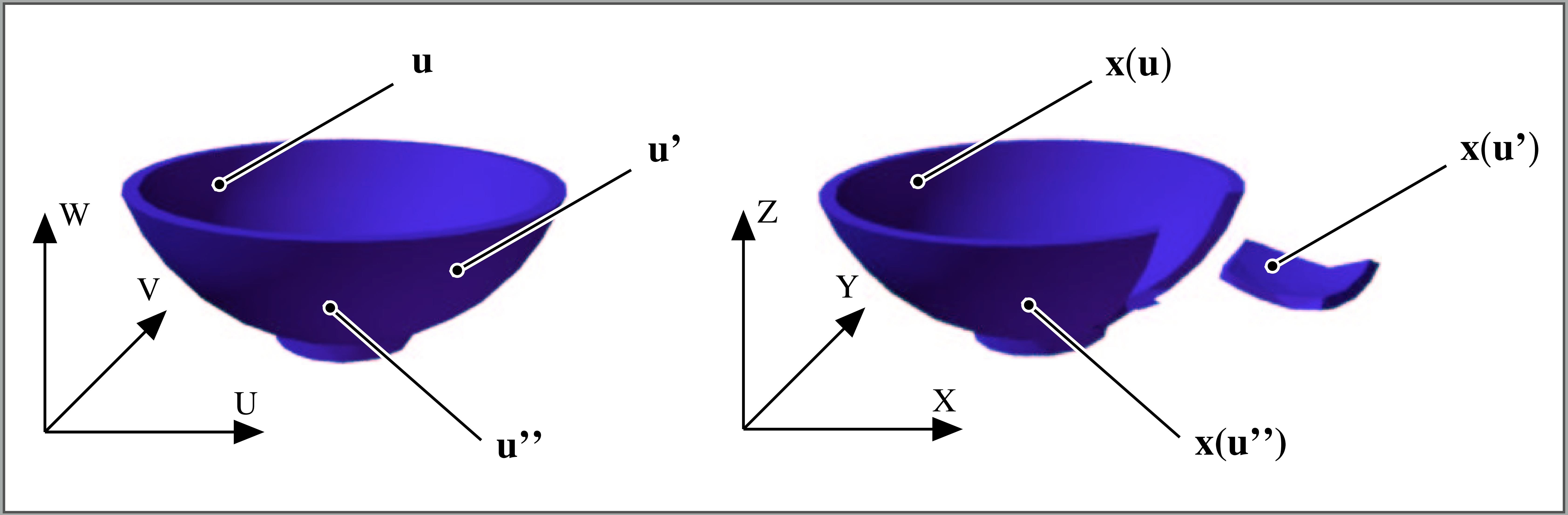
$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{u})$$



Extra material for 284A

Lagrangian Formulation

Deformation described by mapping from material (local) to word coordinates



Example

Extra material for 284A

Another Example

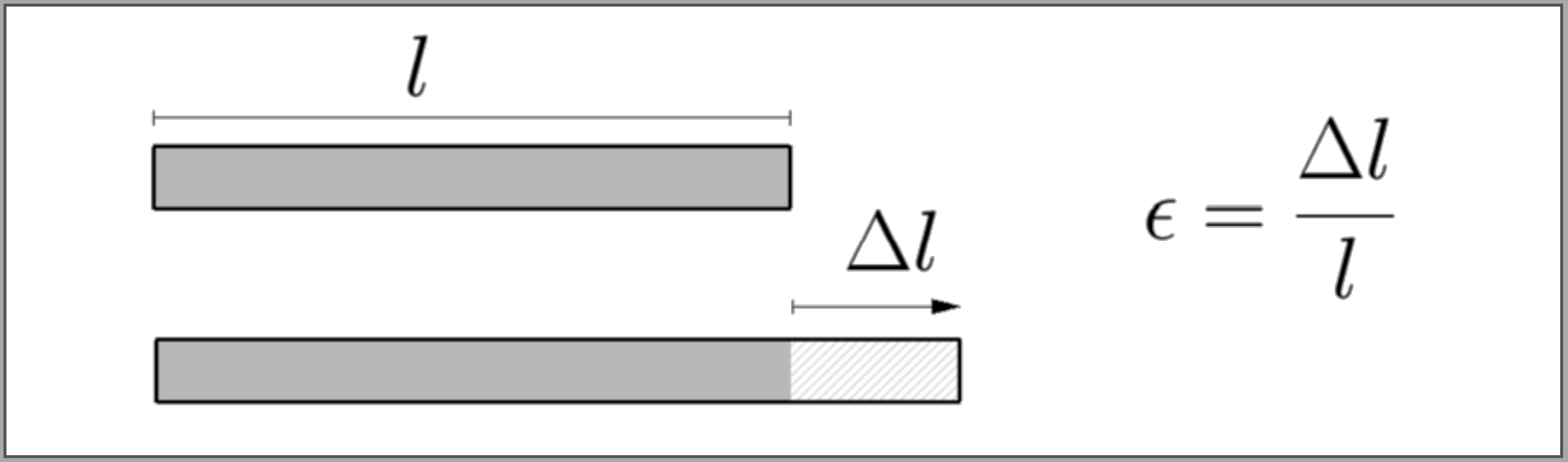
Video footage © LucasArts, used with permission.

Strain

Strain measures deformation

Purely geometric

Example: simple strain in a bar



Strain

Green's strain tensor

$$\epsilon_{ij} = \left(\frac{\partial \boldsymbol{x}}{\partial u_i} \cdot \frac{\partial \boldsymbol{x}}{\partial u_j} \right) - \delta_{ij}$$

Vanishes when not deformed

Only measures deformation

Does not depend on the coordinate system

Strain

Green's strain tensor

$$\epsilon_{ij} = \left(\frac{\partial \boldsymbol{x}}{\partial u_i} \cdot \frac{\partial \boldsymbol{x}}{\partial u_j} \right) - \delta_{ij}$$

$$l_x^2 - l_u^2 = \boldsymbol{d} \cdot \boldsymbol{\epsilon} \cdot \boldsymbol{d}$$

Strain

Cauchy's strain tensor

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial x_i}{\partial u_j} + \frac{\partial x_j}{\partial u_i}\right) - \delta_{ij}$$

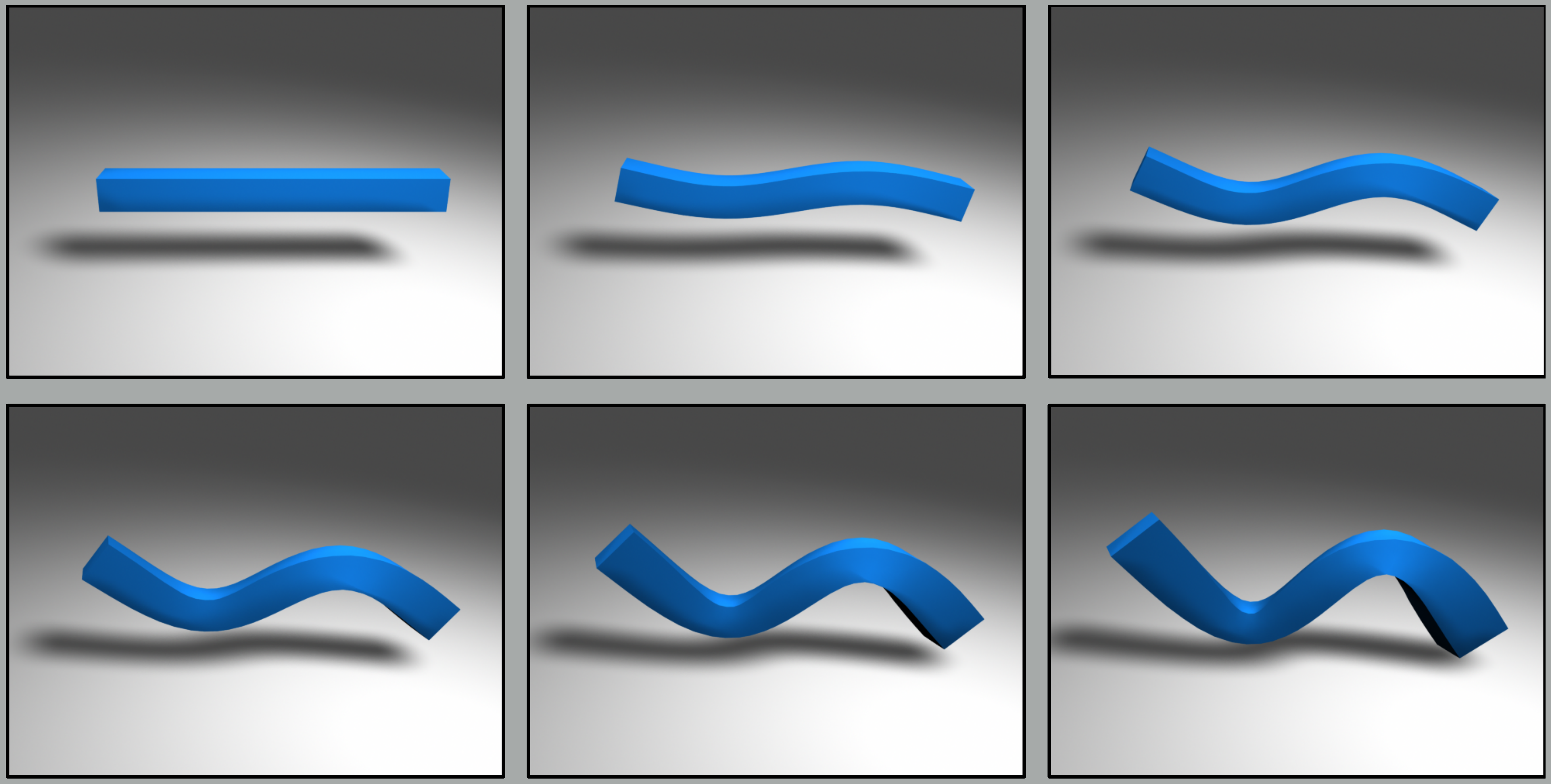
Linearization of Green's strain tensor

Vanishes when not deformed

Not invariant w.r.t rotations

$$l_x - l_u \approx \boldsymbol{d} \cdot \boldsymbol{\epsilon} \cdot \boldsymbol{d}$$

Linearization Errors



We'll fix this problem later...

Extra material for 284A

Strain Rate

Time derivative of Green's strain tensor

Measures rate of deformation

Used for internal damping

$$\dot{\epsilon}_{ij} = \left(\frac{\partial \boldsymbol{x}}{\partial u_i} \cdot \frac{\partial \dot{\boldsymbol{x}}}{\partial u_j} \right) + \left(\frac{\partial \dot{\boldsymbol{x}}}{\partial u_i} \cdot \frac{\partial \boldsymbol{x}}{\partial u_j} \right)$$

Strain Rate

Time derivative of Cauchy's strain tensor

Measures rate of deformation

Used for internal damping

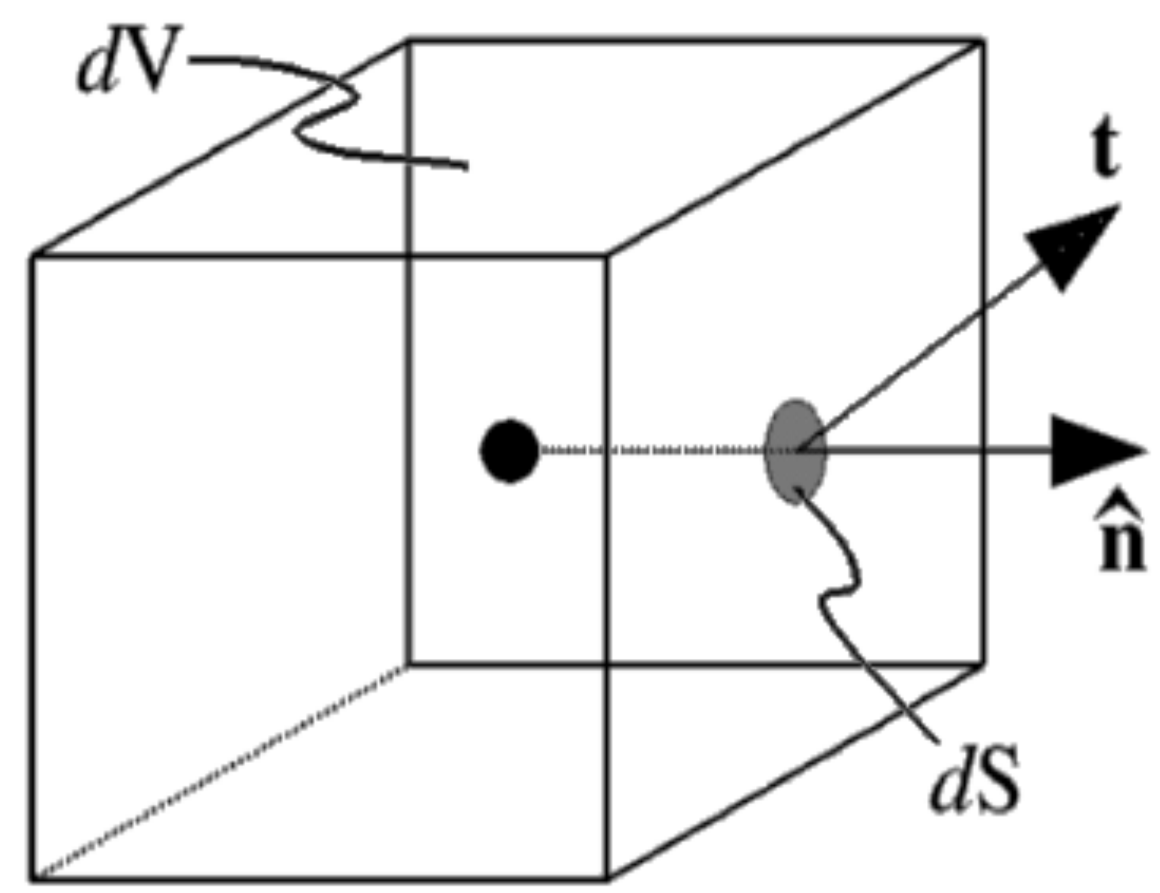
$$\dot{\epsilon}_{ij} = \frac{1}{2}\left(\frac{\partial \dot{x}_i}{\partial u_j} + \frac{\partial \dot{x}_j}{\partial u_i}\right)$$

Stress

Stress determines internal forces

Measures how much material "wants" to return to original shape

$$t = \sigma \cdot \hat{n}$$



Stress due to Strain (linear)

$$\sigma_{ij}^{(\epsilon)} = \underline{C_{ijkl}}\epsilon_{kl}$$

Constitutive parameters

Generalization of

$$f = kd$$

Stress due to Strain (linear, isotropic)

$$\sigma_{ij}^{(\epsilon)} = \sum_{k=1}^{3} \underline{\lambda \epsilon_{kk} \delta_{ij}} + \underline{2\mu \epsilon_{ij}}$$

Elastic (Lamé) Constants

(in)compressibility

rigidity

Generalization of

$$f = kd$$

Ren Ng, James O'Brien

Extra material for 284A

Stress due to Rate

$$\sigma_{ij}^{(\nu)} = \sum_{k=1}^{3} \underline{\psi \dot{\epsilon}_{kk} \delta_{ij}} + \underline{2\psi \dot{\epsilon}_{ij}}$$

Damping Constants

ϕ

ψ

Generalization of

$$f = cv$$

Energy Potentials

Elastic Energy Density

$$\eta = \frac{1}{2} \sum_{i=1}^{3} \sum_{j=1}^{3} \sigma_{ij}^{(\epsilon)} \epsilon_{ij}$$

Generalization of

$$E = \frac{1}{2} k d^2$$

Kinetic Energy Density

$$\kappa = \frac{1}{2} \sum_{i=1}^{3} \sum_{j=1}^{3} \sigma_{ij}^{(\nu)} \dot{\epsilon}_{ij}$$

Generalization of

$$E = \frac{1}{2} m v^2$$

Discretization

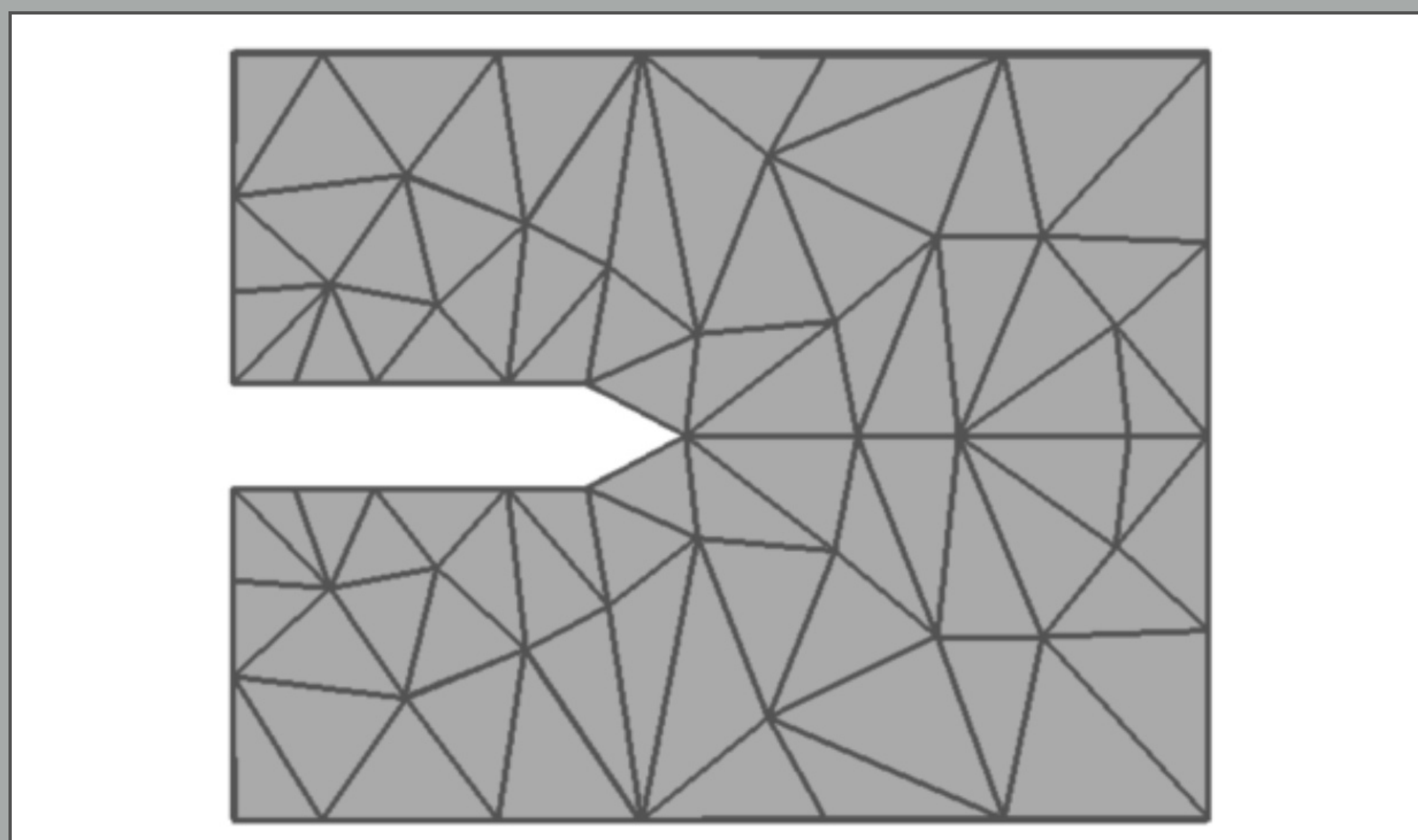
Transition from continuous model to something we can compute with…

Finite Element Method

Disjoint elements tile material domain

Derivatives from shape functions

Nodes shared by adjacent elements

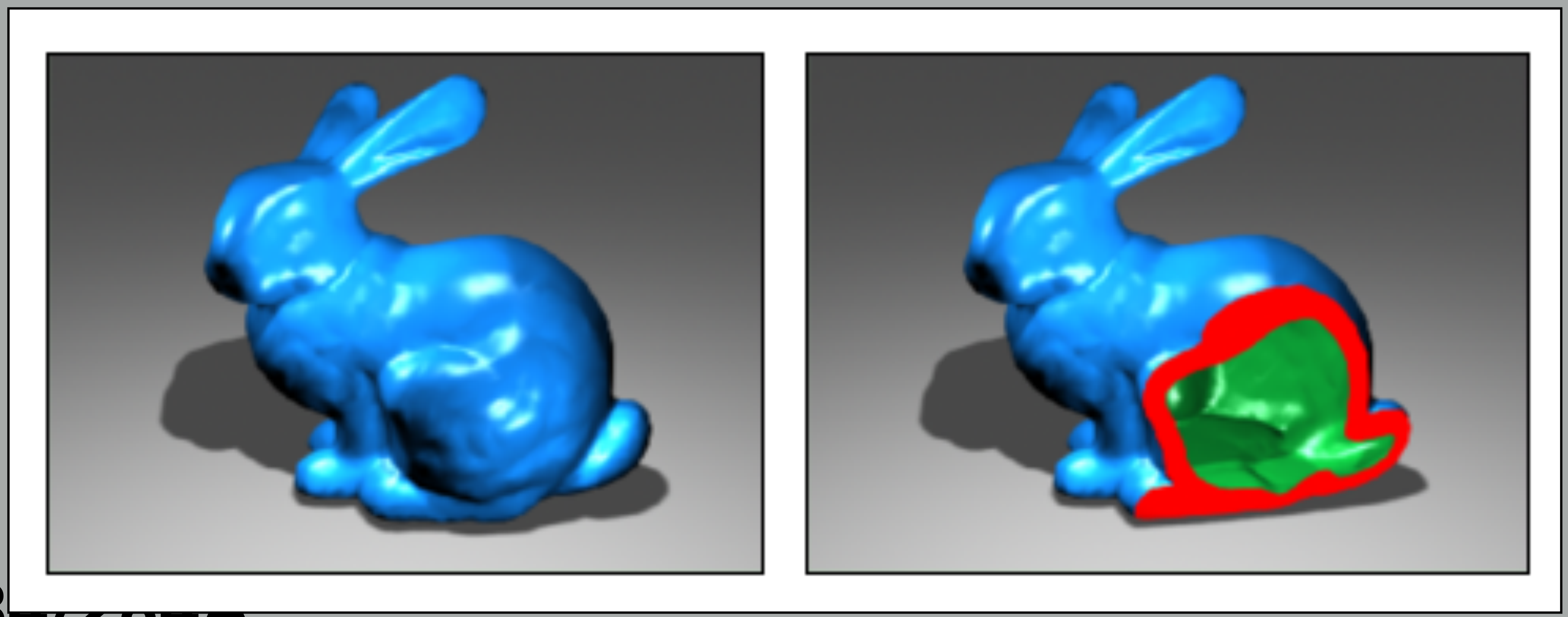


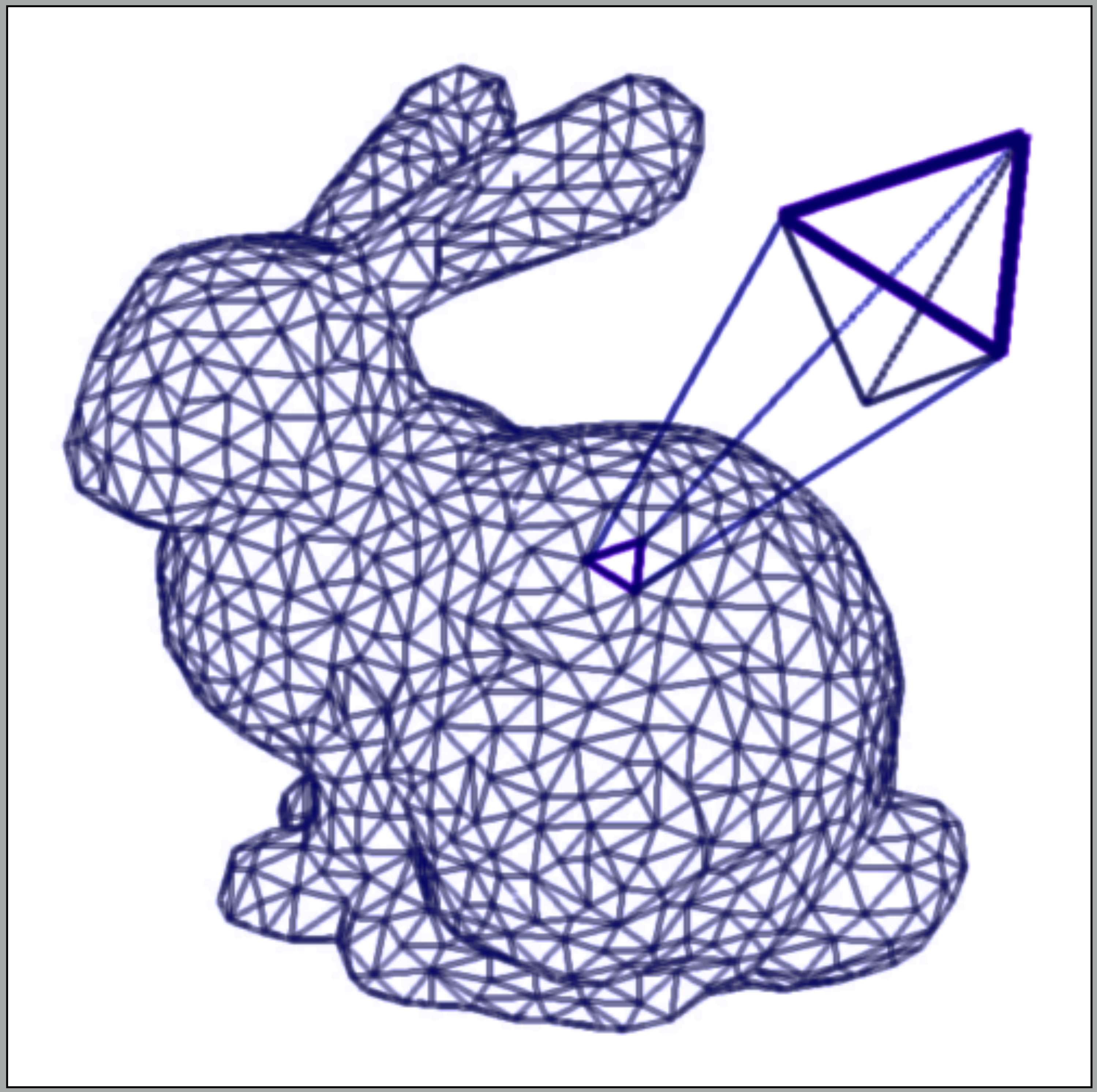
Finite Element Method

Disjoint elements tile material domain

Derivatives from shape functions

Nodes shared by adjacent elements





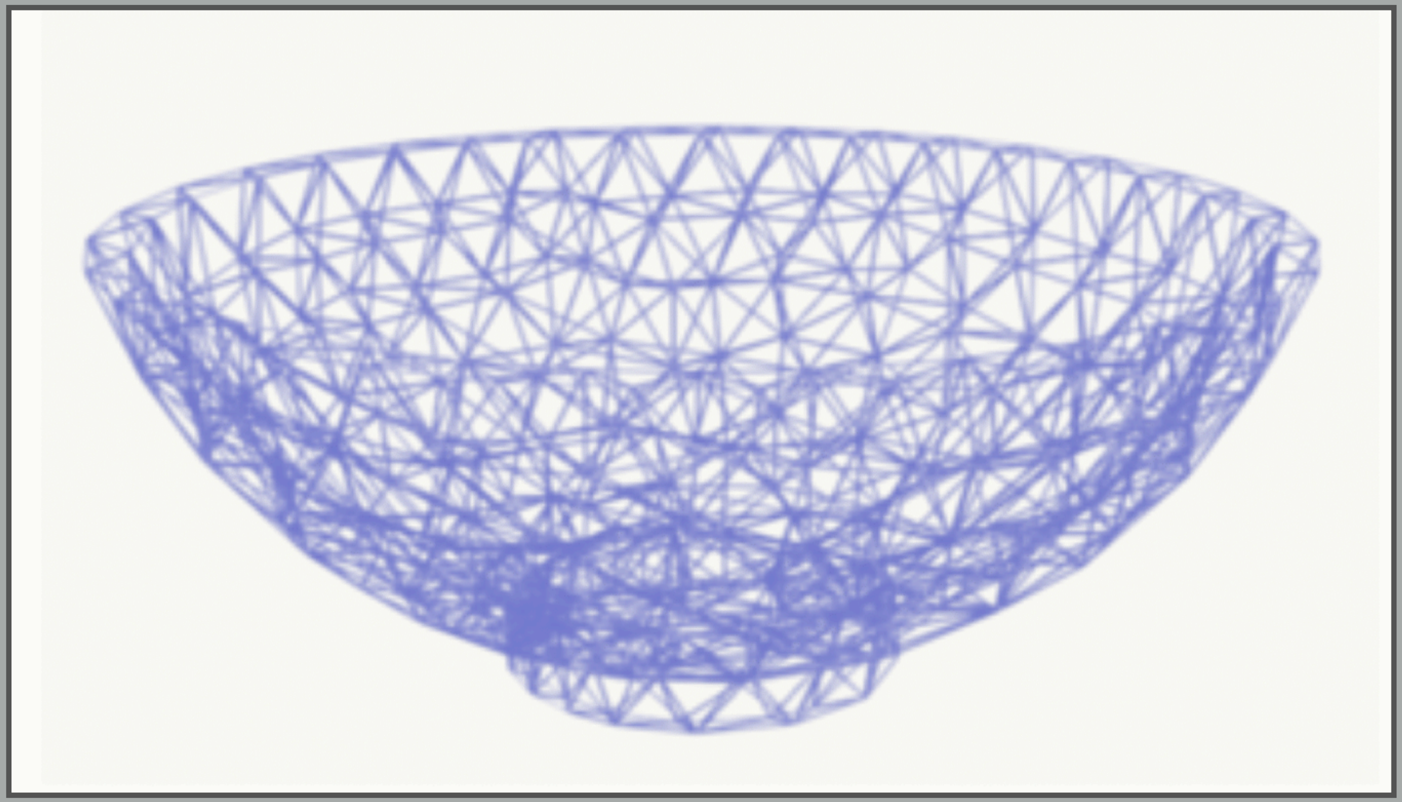
Extra material for 284A

FEM Discretization

Solid volumes

Tetrahedral elements

Linear shape functions



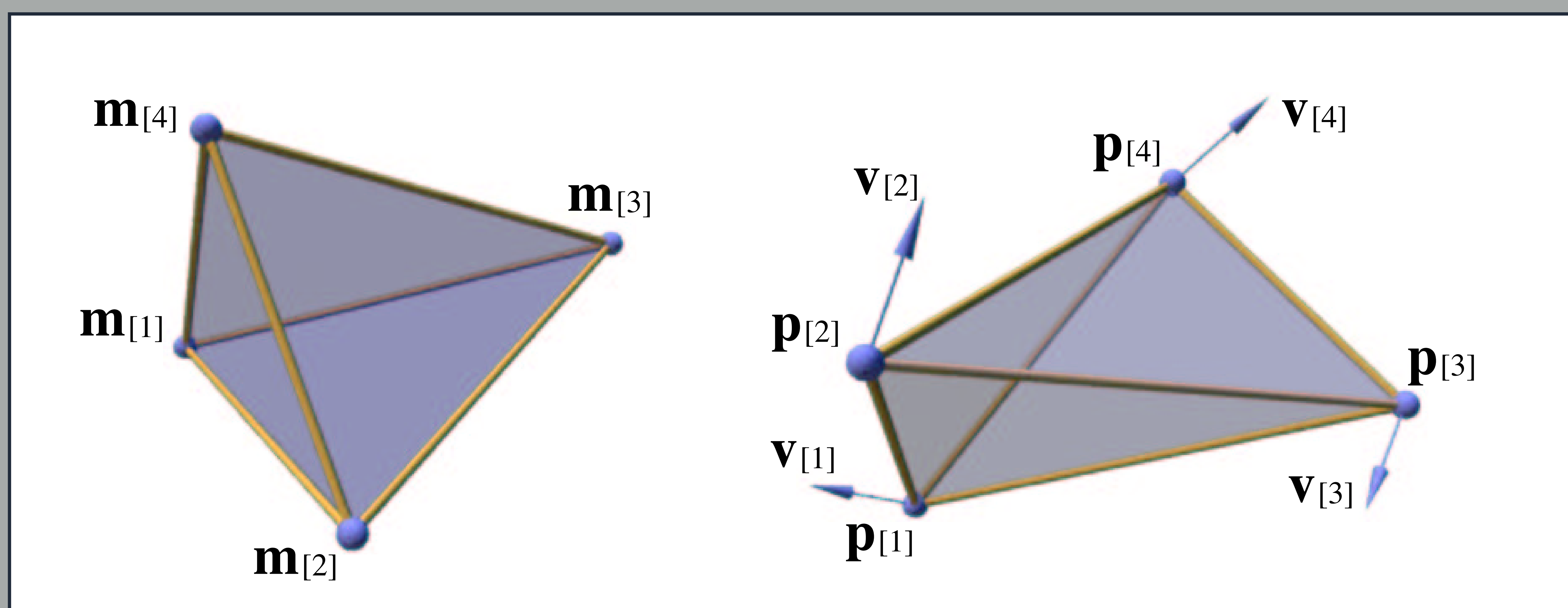
Extra material for 284A

FEM Discretization

• Each element defined by four nodes
• m - location in material (local) coordinates
• p - position in world coordinates
• v - velocity in world coordinates

Each element defined by four nodes

- m - location in material (local) coordinates
- p - position in world coordinates
- v - velocity in world coordinates



Element Shape Functions

Barycentric coordinates

$$\begin{bmatrix} \boldsymbol{u} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{m}_{[1]} & \boldsymbol{m}_{[2]} & \boldsymbol{m}_{[3]} & \boldsymbol{m}_{[4]} \\ 1 & 1 & 1 & 1 \end{bmatrix} \boldsymbol{b}$$

Invert to obtain basis matrix

$$\boldsymbol{b} = \boldsymbol{\beta} \begin{bmatrix} \boldsymbol{u} \\ 1 \end{bmatrix}$$

where

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{m}_{[1]} & \boldsymbol{m}_{[2]} & \boldsymbol{m}_{[3]} & \boldsymbol{m}_{[4]} \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1}$$

Extra material for 284A

Material Derivatives

World pos. as function of material coordinates

$$\boldsymbol{x}(\boldsymbol{u}) = \boldsymbol{P}\,\boldsymbol{\beta}\begin{bmatrix}\boldsymbol{u}\\1\end{bmatrix}$$

where

$$\boldsymbol{P} = \begin{bmatrix}\boldsymbol{p}_{[1]} & \boldsymbol{p}_{[2]} & \boldsymbol{p}_{[3]} & \boldsymbol{p}_{[4]}\end{bmatrix}$$

Derivative w.r.t. material coordinates

$$\frac{\partial \boldsymbol{x}}{\partial u_i} = \boldsymbol{P}\,\boldsymbol{\beta}_{\mathrm{col}_i}$$

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial x_i}{\partial u_j} + \frac{\partial x_j}{\partial u_i}\right) - \delta_{ij}$$

Recall

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial x_i}{\partial u_j} + \frac{\partial x_j}{\partial u_i}\right) - \delta_{ij}$$

$$\sigma_{ij}^{(\epsilon)} = \sum_{k=1}^{3} \lambda \epsilon_{kk} \delta_{ij} + 2\mu\epsilon_{ij}$$

$$\eta = \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{3} \sigma_{ij}^{(\epsilon)} \epsilon_{ij}$$

Node Forces

Combine derivative formula w/ equations for elastic energy

Integrate over volume of element

Take derivative w.r.t. node positions

$$\boldsymbol{f}_{[i]}^{(\epsilon)} = -\frac{\mathrm{vol}}{2} \sum_{j=1}^{4} \boldsymbol{p}_{[j]} \sum_{k=1}^{3} \sum_{l=1}^{3} \beta_{jl} \beta_{ik} \sigma_{kl}^{(\epsilon)}$$

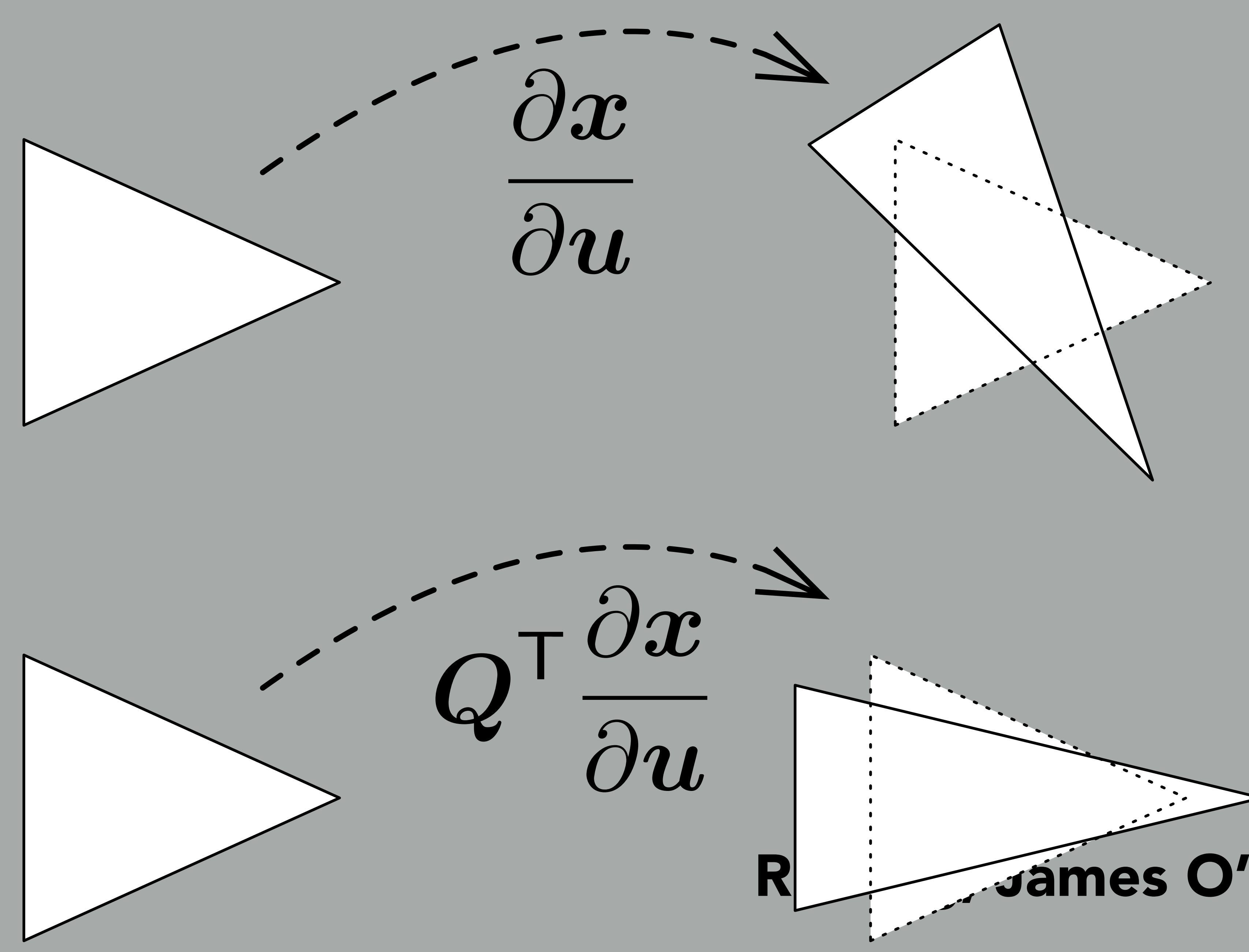
Ren Ng, James O'Brien

Corotational Method

Factor out rotation using polar decomposition

- Cauchy strain without errors due to rotations

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{u}} \rightarrow \boldsymbol{Q}\boldsymbol{F}$$



See paper by
Müller & Gross, 2004

Node Forces and Jacobian

Combine derivative formula w/ equations for elastic energy

Integrate over volume of element

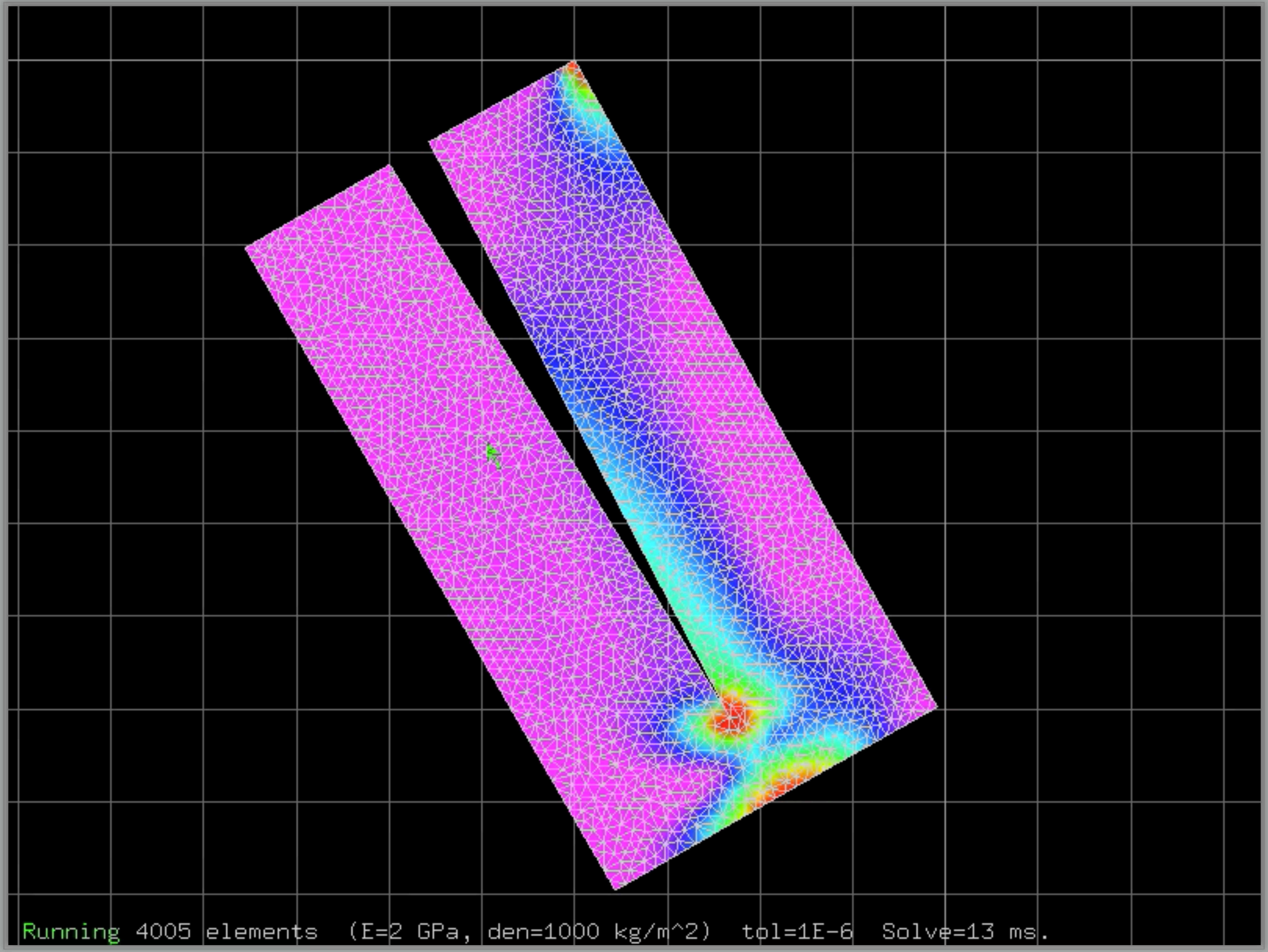
Take derivative w.r.t. node positions

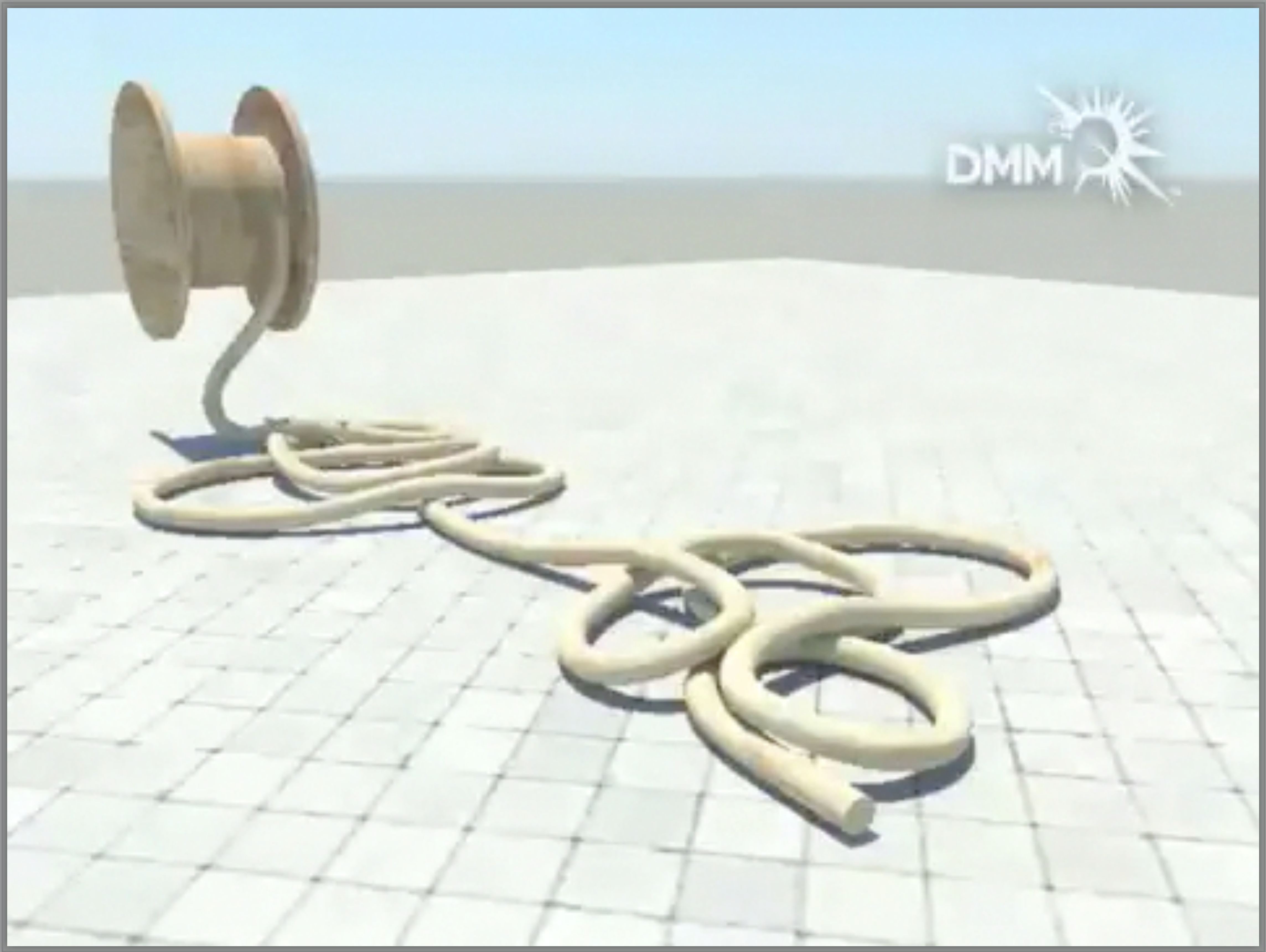
Jacobian core is constant

- 12 x 12 made from little 3 x 3 $J_{[i][j]}$ blocks

$$\boldsymbol{f}_{[i]} = \boldsymbol{Q}\,\boldsymbol{\sigma}\,\boldsymbol{n}_{[i]}$$

$$\boldsymbol{J}_{[i][j]} = -\boldsymbol{Q}(\lambda \boldsymbol{n}_{[i]}\boldsymbol{n}_{[j]}^{\mathsf{T}} + \mu(\boldsymbol{n}_{[i]} \cdot \boldsymbol{n}_{[j]})\boldsymbol{I} + \mu \boldsymbol{n}_{[j]}\boldsymbol{n}_{[i]}^{\mathsf{T}})\boldsymbol{Q}^{\mathsf{T}}$$


Running 4005 elements (E=2 GPa, den=1000 kg/m^2) tol=1E-6 Solve=13 ms.


DMM

Numerical Integration

Euler's Method

Euler's Method (a.k.a. Forward Euler, Explicit)

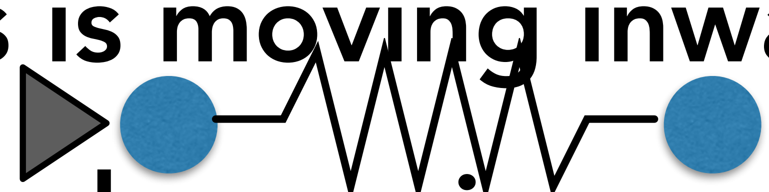
- Simple iterative method
- Commonly used
- Only first order accurate
- *Most often goes unstable (bad)*

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^{t} + \Delta t\, \dot{\boldsymbol{x}}^{t}$$

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^{t} + \Delta t\, \ddot{\boldsymbol{x}}^{t}$$

Euler's Method and Instability

When mass is moving inward:



$$\boldsymbol{f}_{a \to b} = k_S(\boldsymbol{b} - \boldsymbol{a})$$

- Force is decreasing
- Each time-step overestimates the velocity change (increases energy)

When mass gets to origin

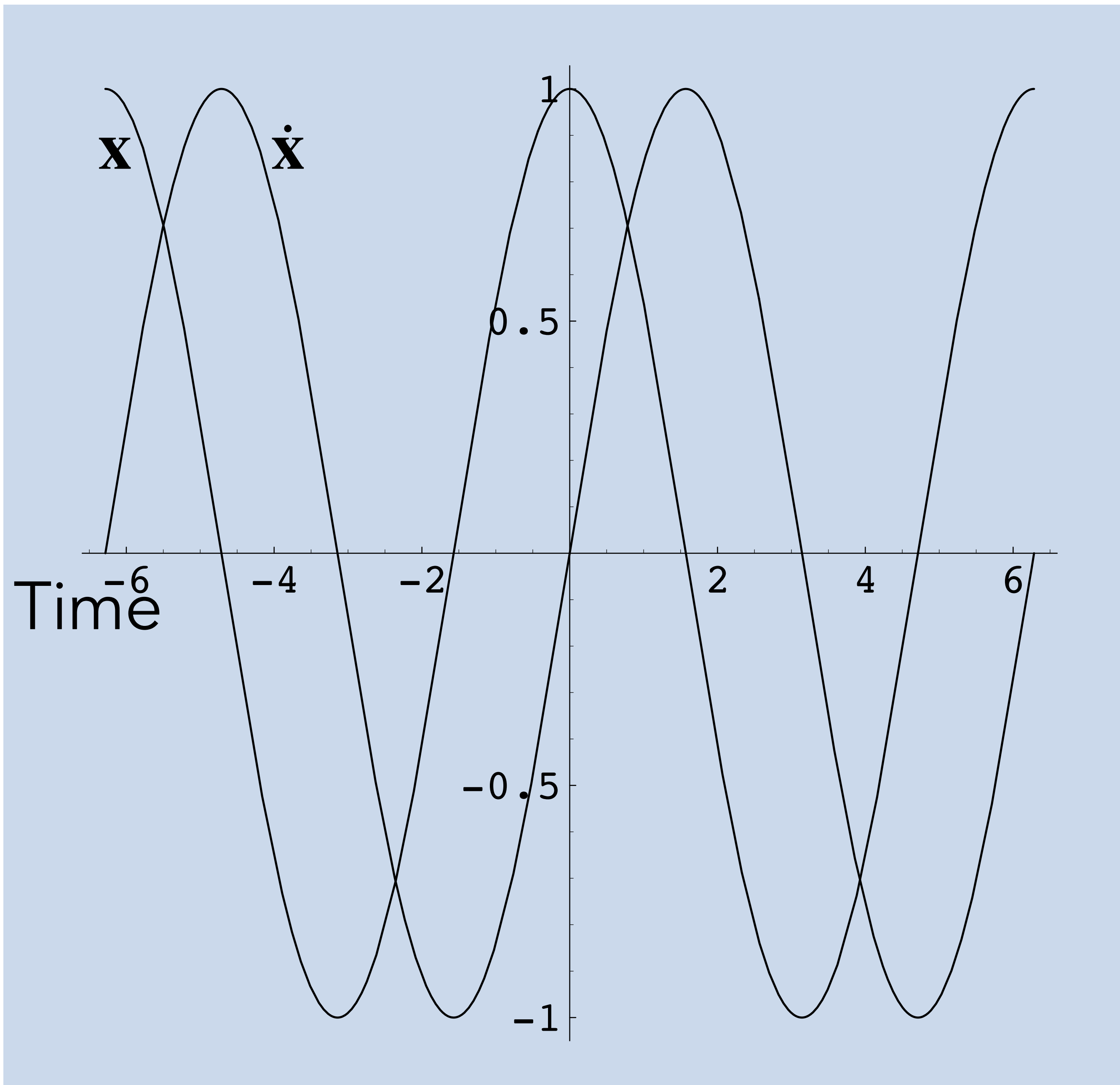
- Has velocity that is too high, now traveling outward

When mass is moving outward

- Force is increasing
- Each time-step underestimates the velocity change (increases energy)

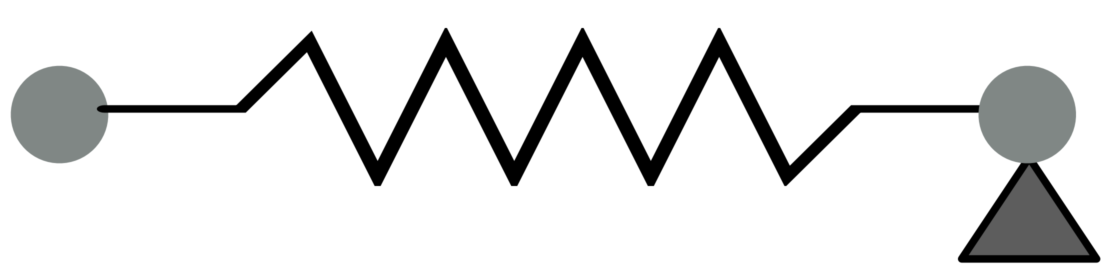
At each motion cycle, mass gains energy exponentially

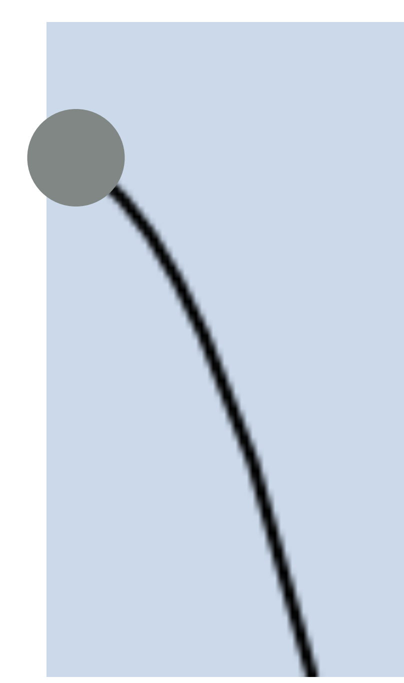
Euler's Method and Instability



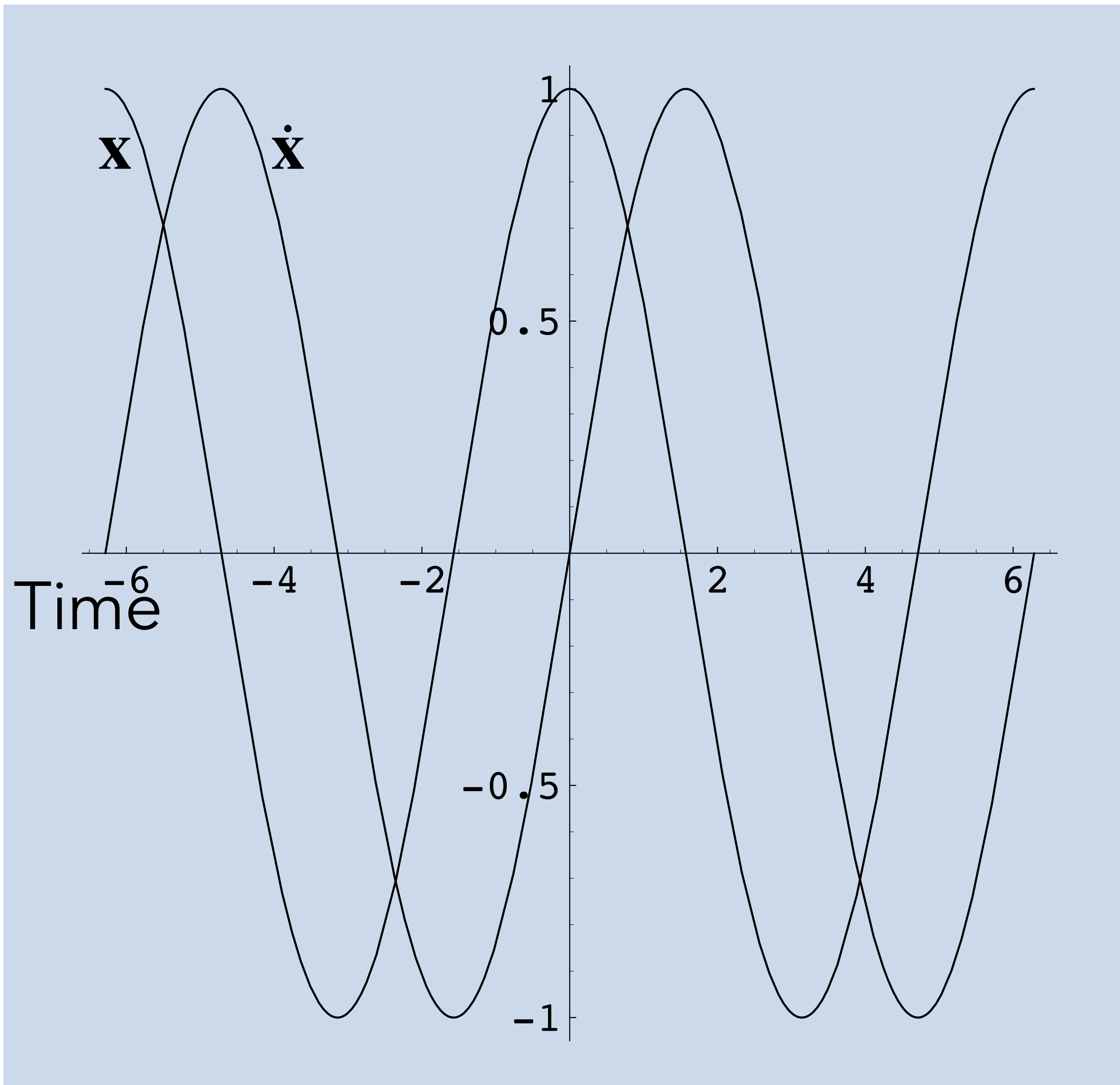
Ideal system:

- Preserves energy $E_T \equiv$ const
- Cycle between kinetic and elastic
  - $E_T = E_K + E_E = \dfrac{m\,\dot{x}^2}{2} + \dfrac{k\,x^2}{2}$



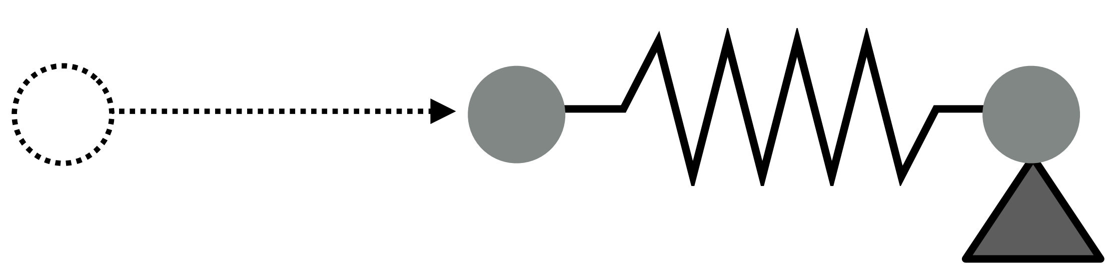


Euler's Method and Instability

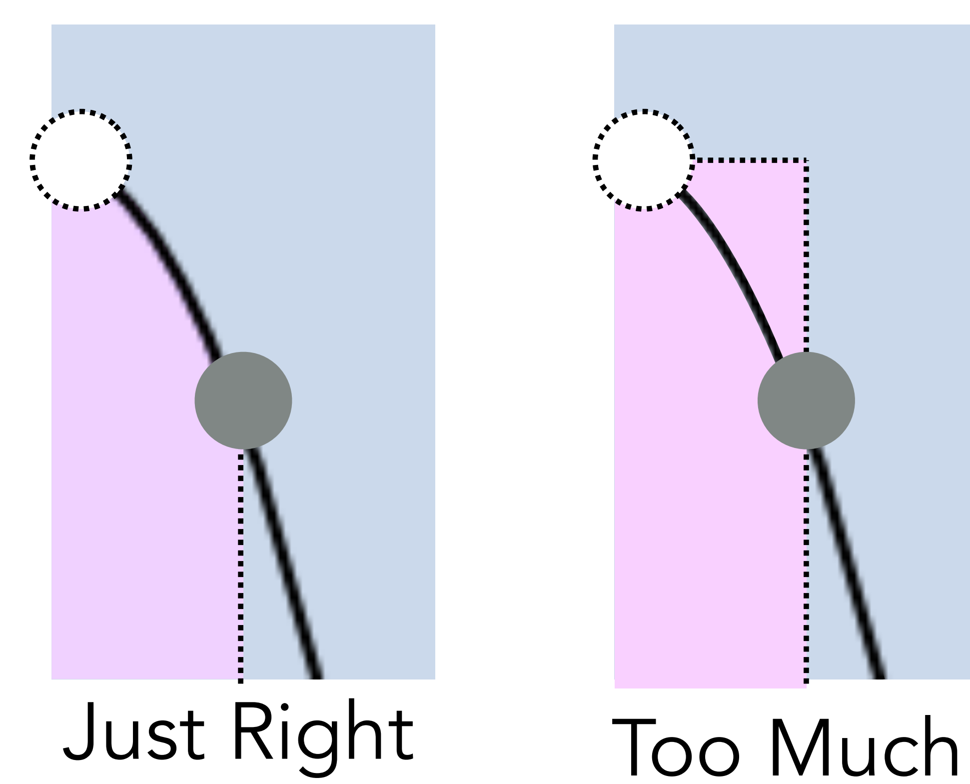


Ideal system:

- **Preserves energy $E_T \equiv$ const**
- **Cycle between kinetic and elastic**
  - $E_T = E_K + E_E = \dfrac{m\,\dot{x}^2}{2} + \dfrac{k\,x^2}{2}$

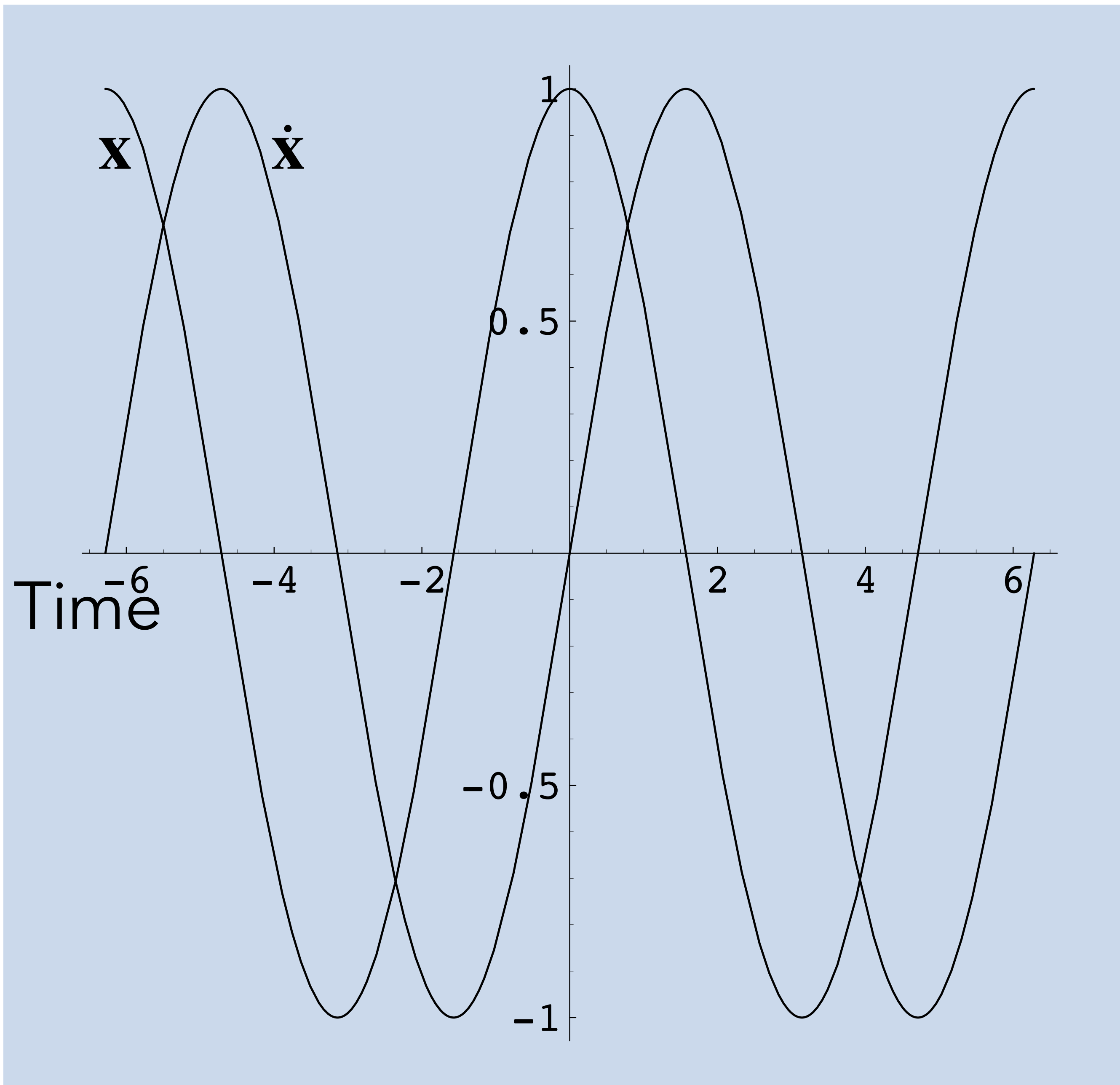


Base fixed at zero



Total *acceleration* is integral under curve.

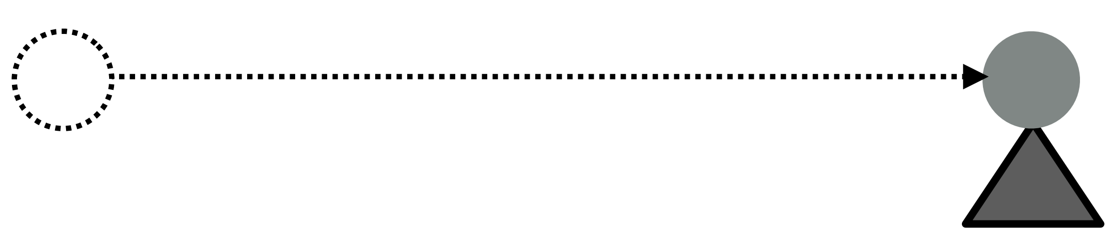
Euler's Method and Instability



Ideal system:

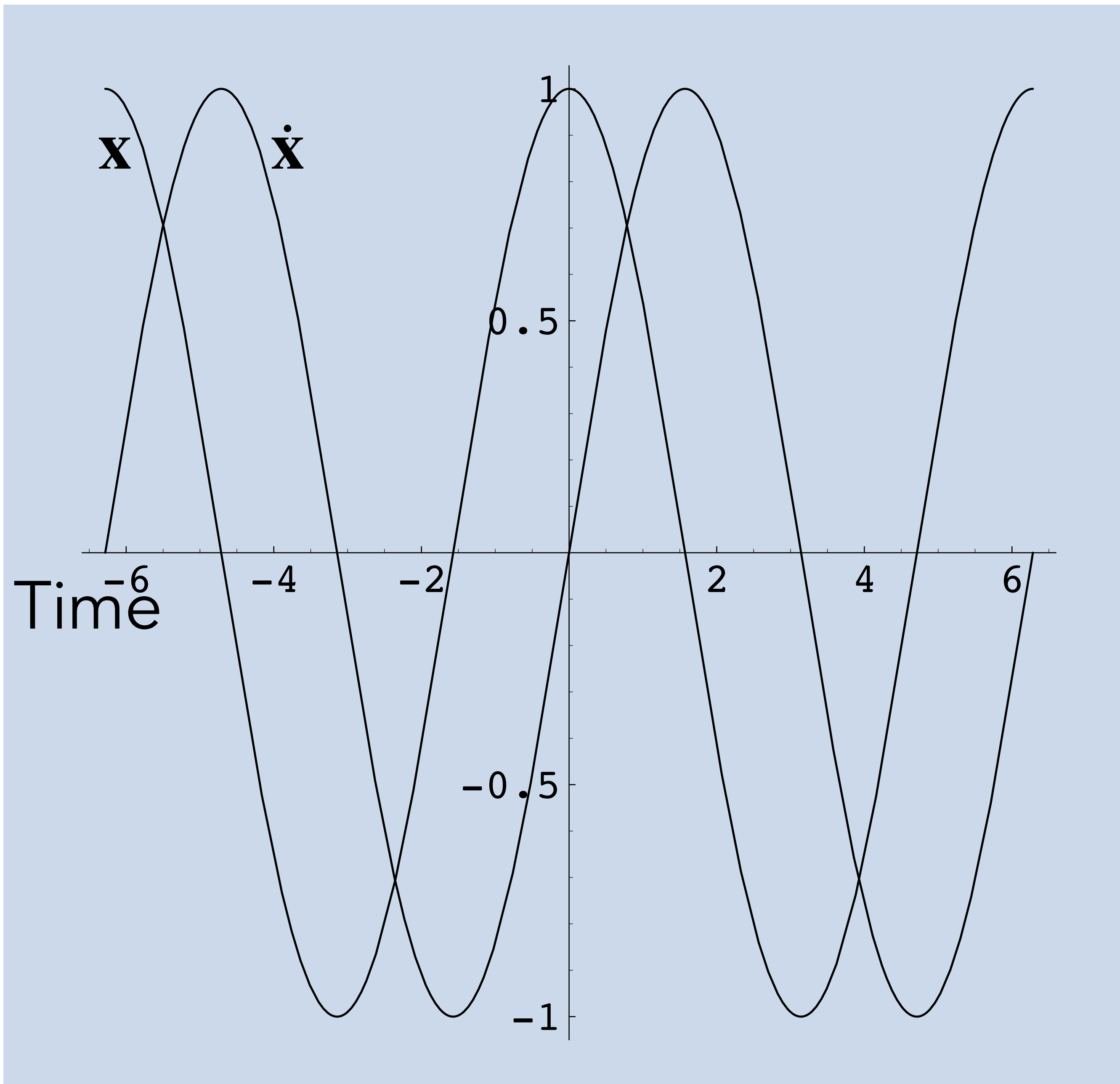
- **Preserves energy** $E_T \equiv$ **const**
- **Cycle between kinetic and elastic**
  - $E_T = E_K + E_E = \frac{m\,\dot{x}^2}{2} + \frac{k\,x^2}{2}$

When zero displacement reached, going too fast!



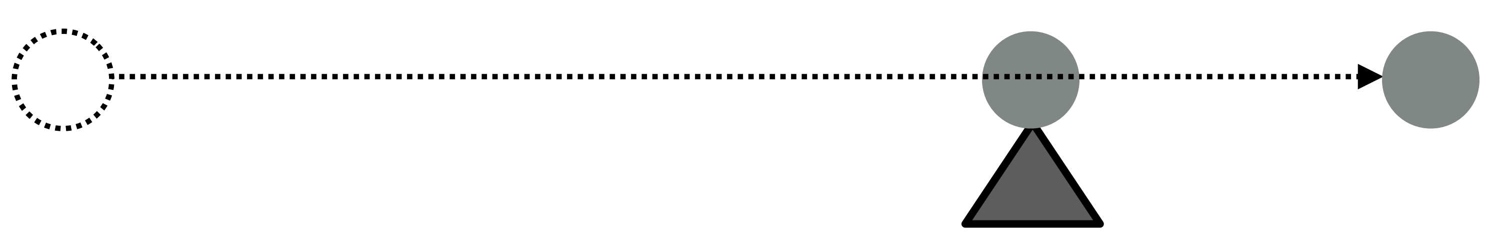
Base fixed at zero

Euler's Method and Instability

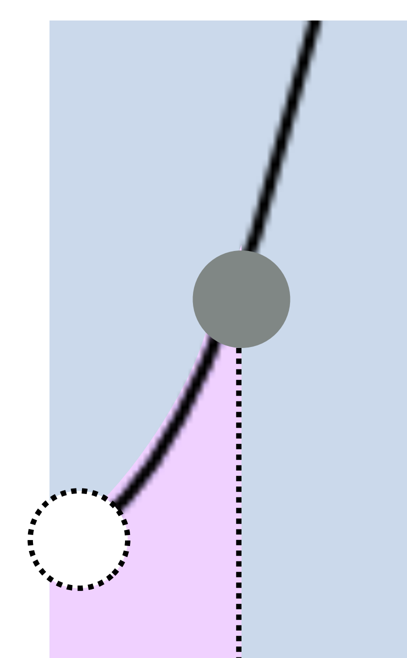


Ideal system:

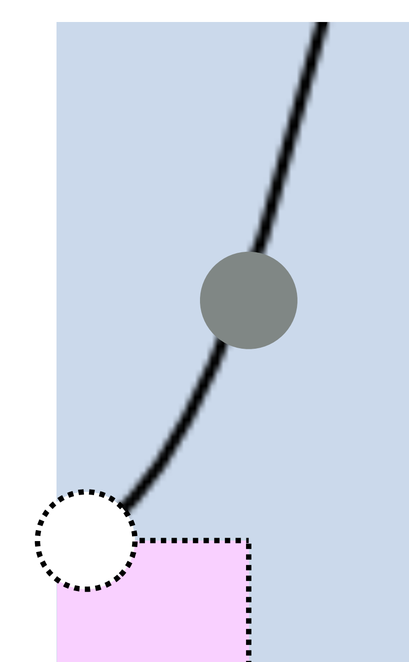
- **Preserves energy $E_T \equiv$ const**
- **Cycle between kinetic and elastic**
  - $E_T = E_K + E_E = \dfrac{m\,\dot{x}^2}{2} + \dfrac{k\,x^2}{2}$



Base fixed at zero



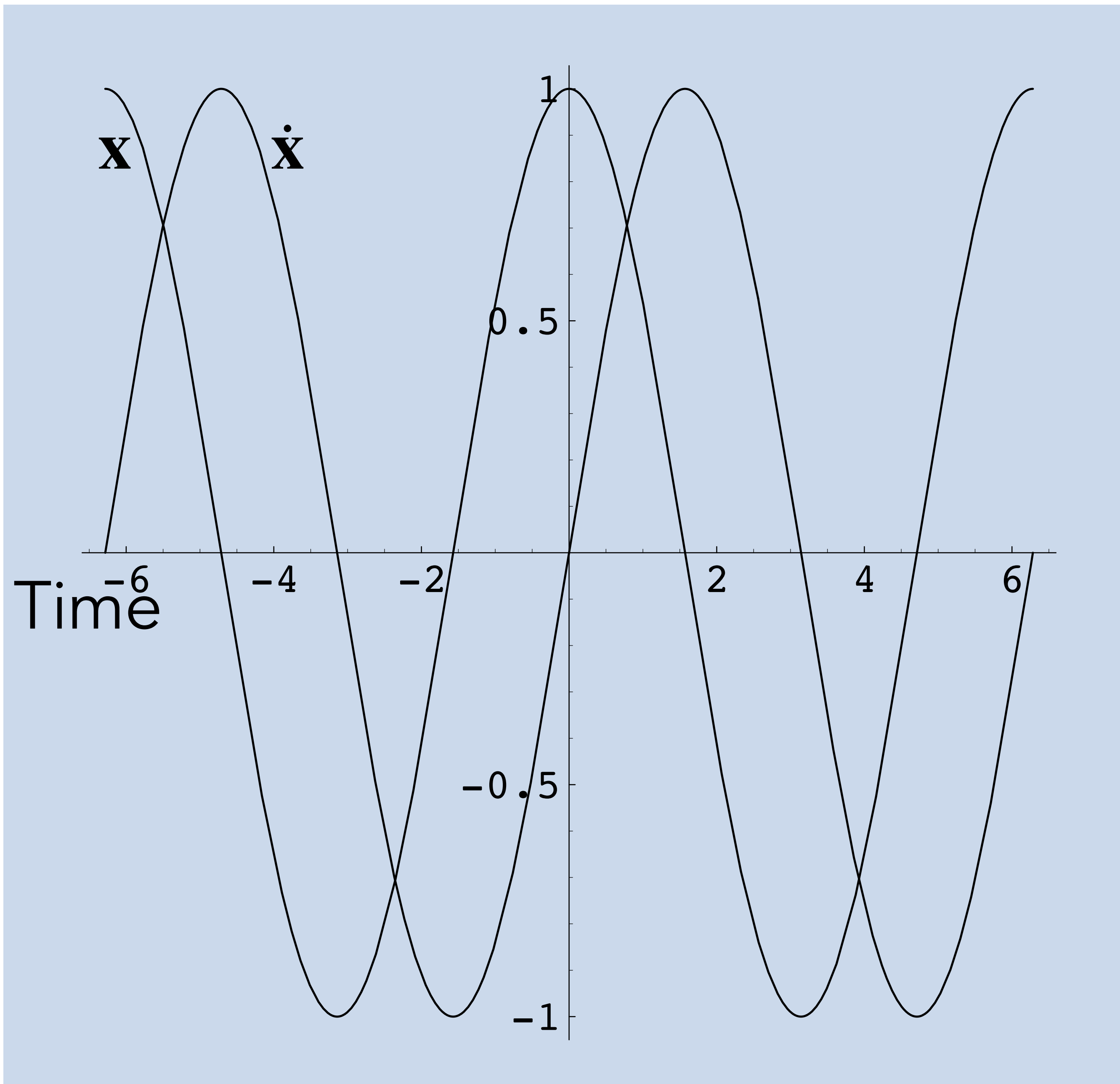
Just Right



Too Little!

Total *deceleration* is integral under curve.

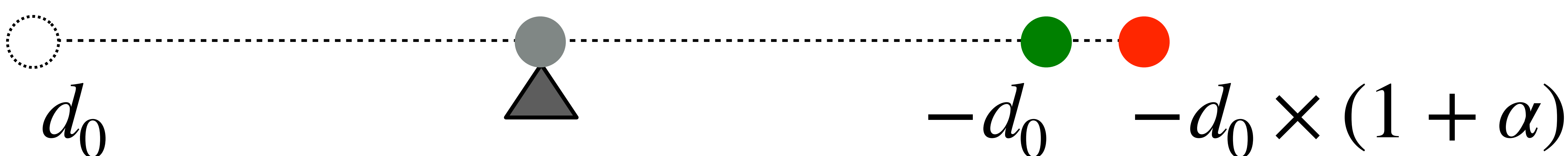
Euler's Method and Instability



Ideal system:

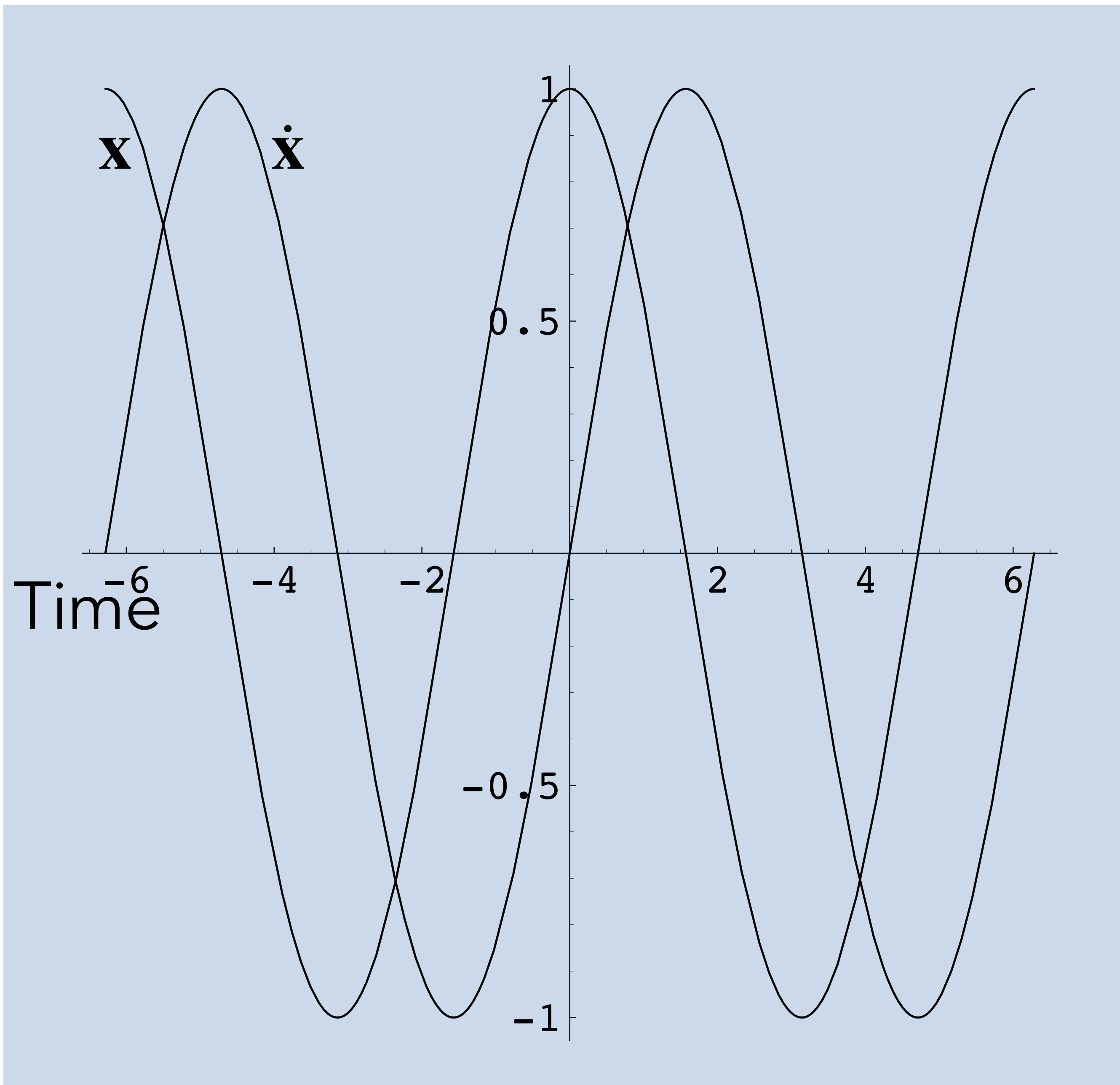
- **Preserves energy** $E_T \equiv$ **const**
- **Cycle between kinetic and elastic**
  - $E_T = E_K + E_E = \frac{m\,\dot{x}^2}{2} + \frac{k\,x^2}{2}$

Overshoots symmetric location



Base fixed at zero

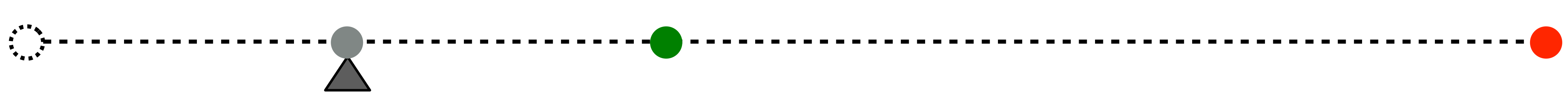
Euler's Method and Instability



Ideal system:

- **Preserves energy** $E_T \equiv$ **const**
- **Cycle between kinetic and elastic**
  - $E_T = E_K + E_E = \frac{m\,\dot{x}^2}{2} + \frac{k\,x^2}{2}$

Overshoots symmetric location



$$-d_0 \times (1+\alpha)^{\#\text{cycles}}$$

Exponential divergence!

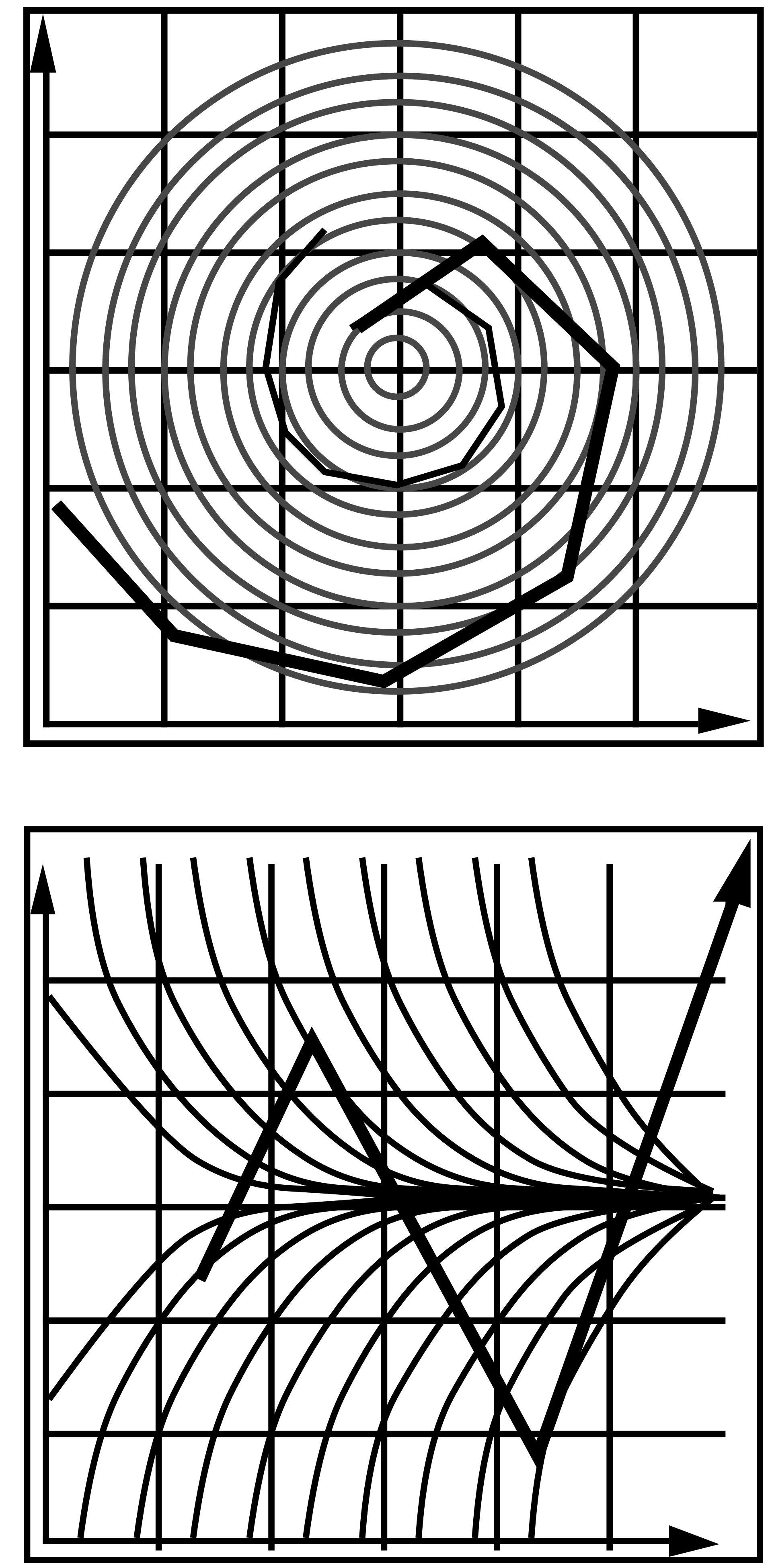
Euler's Method and Instability

Forward Euler (explicit)

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t\, \boldsymbol{v}(\boldsymbol{x}, t)$$

Two key problems:

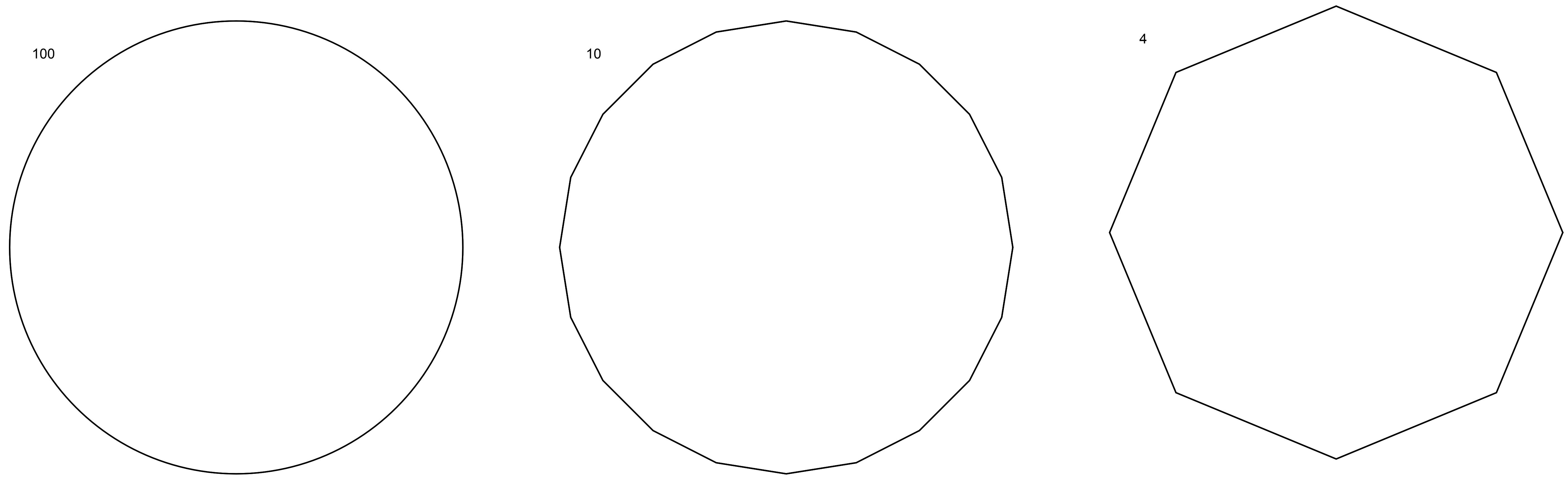
- Inaccuracies increase as time step Δt increases
- Instability is a common, serious problem that can cause simulation to diverge



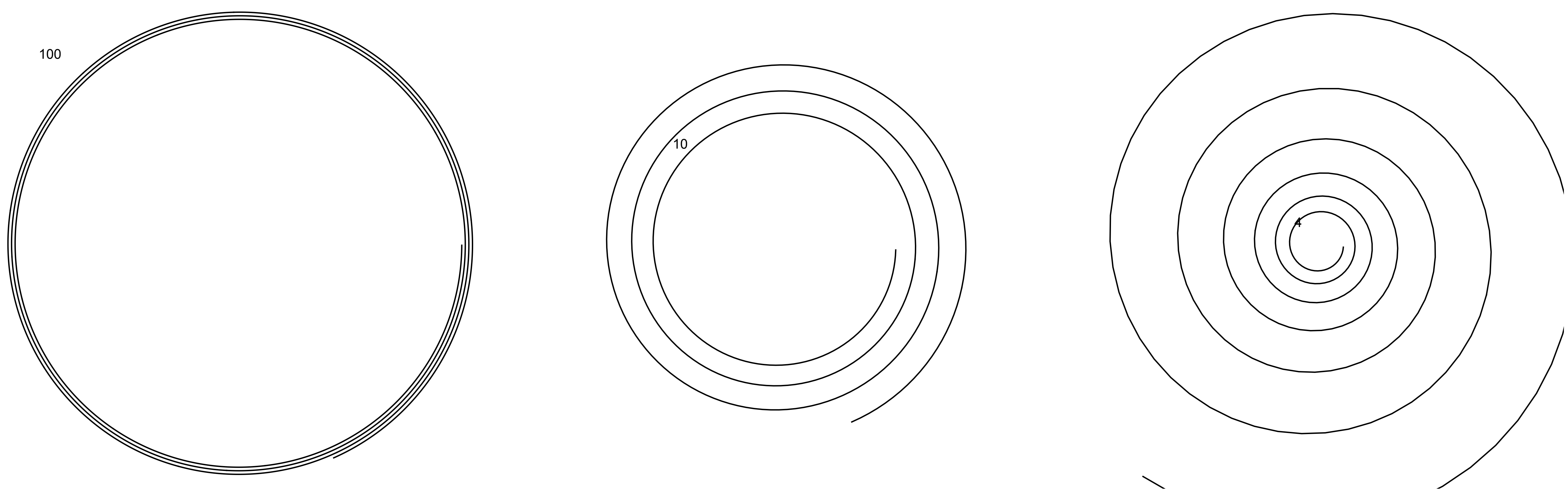
Witkin and Baraff

Integration Errors Accumulate

Evaluating known function (a circle)



Integrating first derivative



Errors and Instability

Solving by numerical integration with finite differences leads to two problems

Errors

- Errors at each time step accumulate. Accuracy decreases as simulation proceeds
- Accuracy may not be critical in graphics applications

Instability

- Errors can compound, causing the simulation to diverge even when the underlying system does not
- Lack of stability is a fundamental problem in simulation, and cannot be ignored

Combating Instability

Some Methods to Combat Instability

Modified Euler

- Average velocities at start and endpoint

Adaptive step size

- Compare one step and two half-steps, recursively, until error is acceptable

Implicit methods

- Use the velocity at the next time step (hard)

Position-based / Verlet integration

- Constrain positions and velocities of particles after time step

Modified Euler

Modified Euler

- Average velocity at start and end of step
- OK if system is not very stiff (*e.g.: k_s* is small)
- But, still unstable

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^t + \Delta t \, \ddot{\boldsymbol{x}}^t$$

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \frac{\Delta t}{2} \left(\dot{\boldsymbol{x}}^t + \dot{\boldsymbol{x}}^{t+\Delta t} \right)$$

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t \, \dot{\boldsymbol{x}}^t + \frac{(\Delta t)^2}{2} \, \ddot{\boldsymbol{x}}^t$$

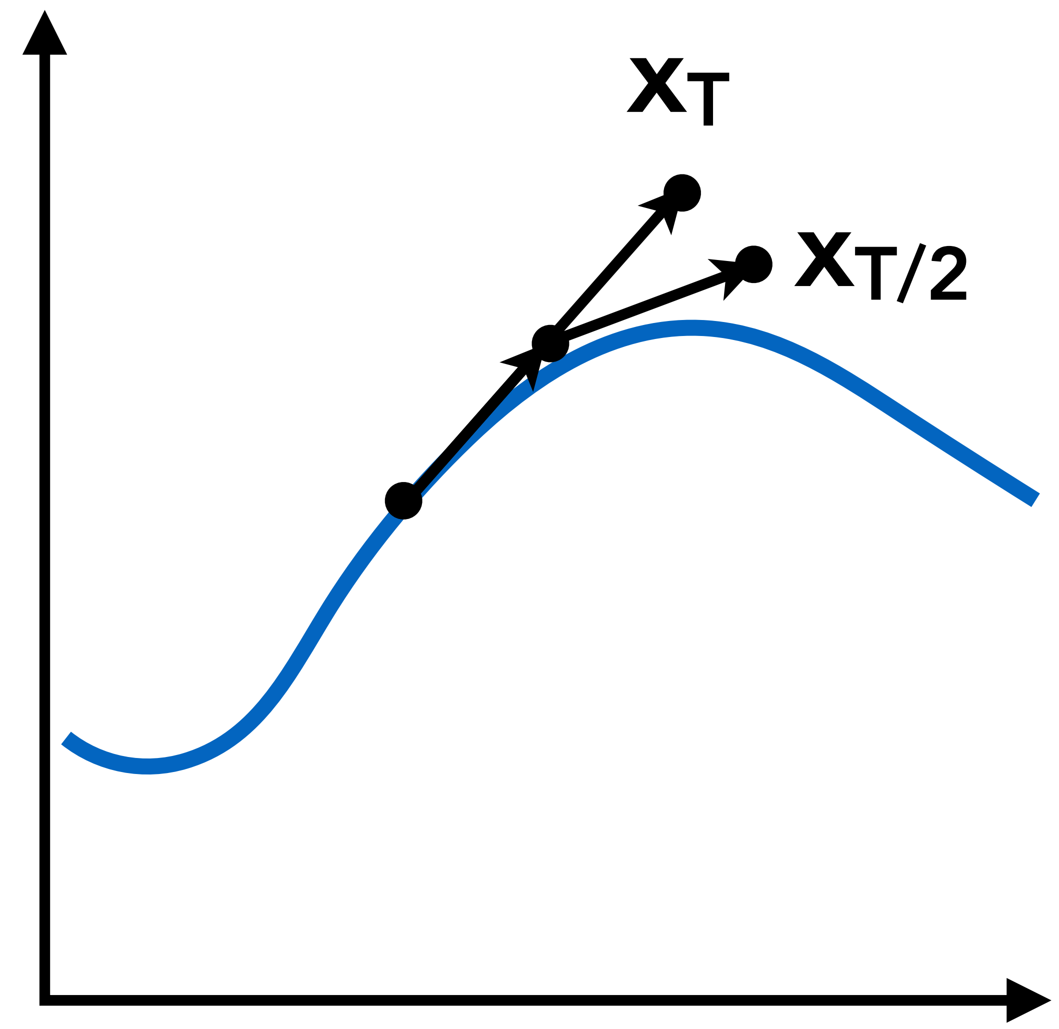
Adaptive Step Size

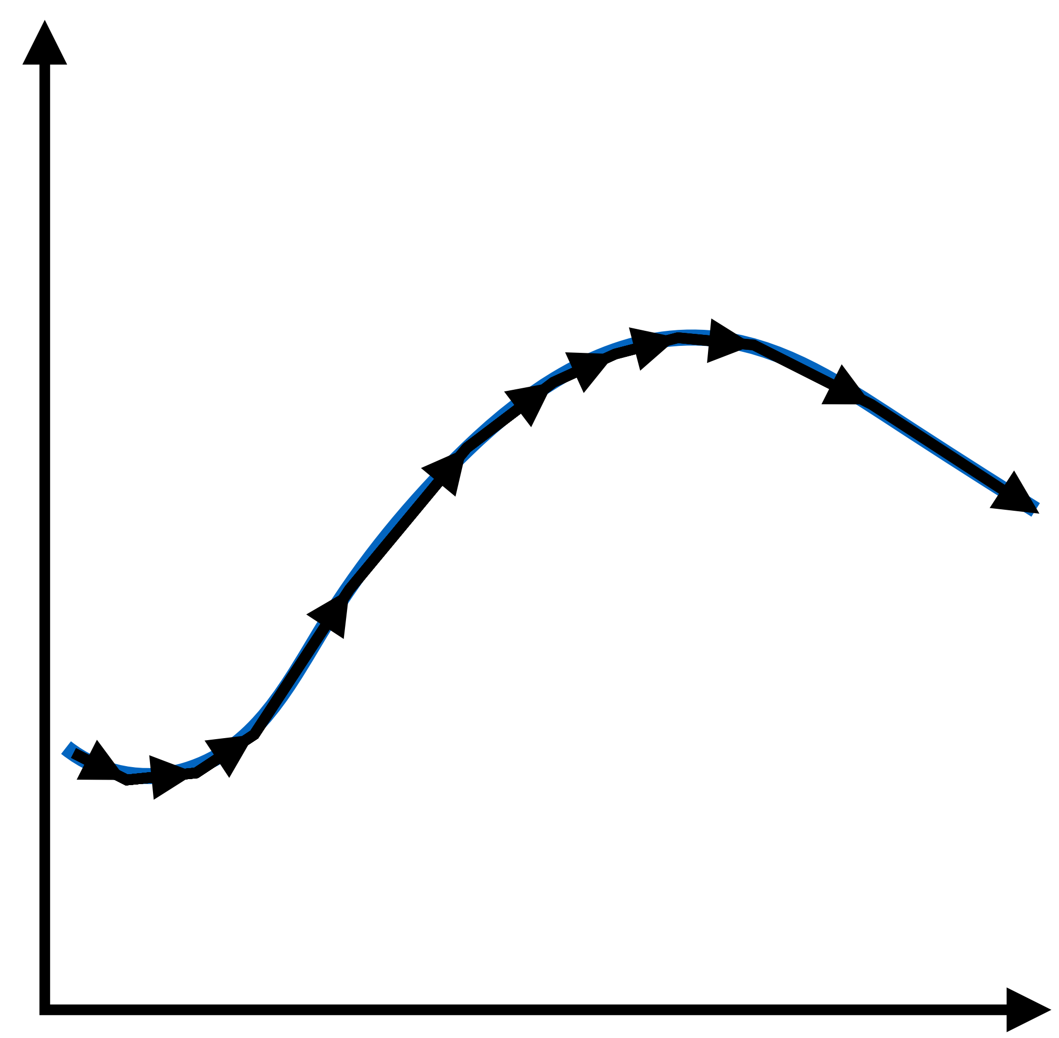
Adaptive step size

- Technique for choosing step size based on error estimate
- Highly recommended technique
- But may need very small steps!

Repeat until error is below threshold:

- Compute x_T an Euler step, size T
- Compute $x_{T/2}$ two Euler steps, size T/2
- Compute error $\| x_T - x_{T/2} \|$
- If (error > threshold) reduce step size and try again





Slide credit: Funkhouser

Implicit Euler Method

Implicit methods

- Informally called backward methods
- Use derivatives in the future, for the current step

Forward/Explicit Euler

$$\mathscr{K}(\mathbf{x}^t) + \mathscr{D}(\mathbf{x}^t, \dot{\mathbf{x}}^t) + \mathbf{M}\,\ddot{\mathbf{x}} = \mathbf{f}^t$$

Backward/Implicit Euler

$$\mathscr{K}(\mathbf{x}^{t+\Delta t}) + \mathscr{D}(\mathbf{x}^{t+\Delta t}, \dot{\mathbf{x}}^{t+\Delta t}) + \mathbf{M}\,\ddot{\mathbf{x}} = \mathbf{f}^t$$

Red variables are unknown.

Implicit Euler Method

Implicit methods

- Informally called backward methods
- Use derivatives in the future, for the current step

Forward/Explicit Euler

$$\mathscr{K}(\mathbf{x}^t) + \mathscr{D}(\mathbf{x}^t, \dot{\mathbf{x}}^t) + \mathbf{M}\,\ddot{\mathbf{x}} = \mathbf{f}^t$$

Backward/Implicit Euler

$$\mathscr{K}(\mathbf{x}^t + \Delta\mathbf{x}) + \mathscr{D}(\mathbf{x}^t + \Delta\mathbf{x}, \dot{\mathbf{x}}^t + \Delta\dot{\mathbf{x}}) + \mathbf{M}\,\ddot{\mathbf{x}} = \mathbf{f}^t$$

$$\left.\begin{aligned} \Delta\mathbf{x} &= \Delta t\,\dot{\mathbf{x}}^{t+\Delta t} \\ \Delta\dot{\mathbf{x}} &= \Delta t\,\ddot{\mathbf{x}} \end{aligned}\right\}$$

Substitute and solve for $\ddot{\mathrm{x}}$

Implicit Euler Method

Implicit methods

- Informally called backward methods
- Use derivatives in the future, for the current step

Forward/Explicit Euler

$$\mathscr{K}(\mathbf{x}^t) + \mathscr{D}(\mathbf{x}^t, \dot{\mathbf{x}}^t) + \mathbf{M}\,\ddot{\mathbf{x}} = \mathbf{f}^t$$

Semi-Implicit Euler / Linearized Implicit Euler (also one Newton solve)

$$\mathbf{K} \cdot (\mathbf{x}^t + \Delta\mathbf{x}) + \mathbf{D} \cdot (\dot{\mathbf{x}}^t + \Delta\dot{\mathbf{x}}) + \mathbf{M} \cdot \ddot{\mathbf{x}} = \mathbf{f}^t$$

$$\left.\begin{aligned} \Delta\mathbf{x} &= \Delta t\, \dot{\mathbf{x}}^{t+\Delta t} \\ \Delta\dot{\mathbf{x}} &= \Delta t\, \ddot{\mathbf{x}} \end{aligned}\right\}$$

Substitute and solve for $\ddot{\mathrm{x}}$

Ren Ng, James O'Brien

Implicit Euler Method

Implicit methods

- Informally called backward methods
- Use derivatives in the future, for the current step
- Solve nonlinear problem for $\ddot{\mathbf{x}}$
- Use root-finding algorithm, *e.g.* Newton's method
- Can be made unconditionally stable

- Dump energy and may look over-damped

Position-Based / Verlet Integration

Idea:

- After modified Euler forward-step, constrain positions of particles to prevent divergent, unstable behavior
- Use constrained positions to calculate velocity
- Both of these ideas will dissipate energy, stabilize

Pros / cons

- Fast and simple
- Dissipates energy in constraints
- Highly recommended (assignment)

Position-Based / Verlet Integration

Algorithm 1 Position-based dynamics

for all vertices i **do**
  initialize $\mathbf{x}_i = \mathbf{x}_i^0$, $\mathbf{v}_i = \mathbf{v}_i^0$, $w_i = 1/m_i$
end for
loop
  for all vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{\text{ext}}(\mathbf{x}_i)$
  for all vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
  for all vertices i **do** genCollConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
  loop solverIteration **times**
    projectConstraints($C_1, \ldots, C_{M+M_{\text{Coll}}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
  end loop
  for all vertices i **do**
    $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
    $\mathbf{x}_i \leftarrow \mathbf{p}_i$
  end for
  velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
end loop

Position-Based Simulation Methods in Computer Graphics
Bender, Müller, Macklin, Eurographics 2015

Projective Dynamics

Examples of Projective Dynamics

- Position Based Dynamics
  - “Position Based Dynamics,” VRIPHYS 2006
- Provot’s Method
  - “Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior,” GI 95
- Fast Springs
  - “Fast Simulation of Mass-Spring Systems,” SIGGRAPH Asia 2013
- Shape Matching
  - “Meshless Deformations Based on Shape Matching,” SIGGRAPH 2005
- and many others are examples of

Projective Dynamics

General Approach

- Separate system into stiff and non-stiff aspects
  - Stiff expressed as constraints
    - *e.g.:* $||\mathbf{a} - \mathbf{b}|| - \textit{max\_length} \leq 0$
  - Non-stiff expressed as forces
    - *e.g.:* $\mathbf{f}_a = k_d(\mathbf{b} - \mathbf{a})$
- For each time step
  - Integrate non-stiff stuff normally
  - Enforce stiff constraints
  - Update velocities to satisfy constraints

Simulation as Constraint Optimization

Simulation as Constraint Optimization

Standard view of simulation:

- Start with initial configuration, *e.g.:* $\mathbf{q} = \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix}$
- Integrate forward, *e.g.:* $\mathbf{q}^{t+\Delta t} = \mathbf{q}^{t} + \Delta \mathbf{q}$
- Keep going until end of time

Optimization view

- Start with initial configuration, $\mathbf{q}^{t_0}$, and final configuration, $\mathbf{q}^{t_N}$.
- Interpolate to get initial interior states, $\{\mathbf{q}^{t_1}, \mathbf{q}^{t_2}, \mathbf{q}^{t_3}, \ldots, \mathbf{q}^{t_{N-1}}\}$
- Minimize dynamics error over sequence

Simulation as Constraint Optimization

Dynamics Error:

$$E = \sum_i \left(\left(\mathbf{q}^{t+\Delta t} - \mathbf{q}^t + \Delta t\, \dot{\mathbf{q}}^t\right)^2 + \left(\mathbf{q}^{t+\Delta t} - \mathbf{q}^t + \Delta t\, \dot{\mathbf{q}}^{t+\Delta t}\right) \right)$$

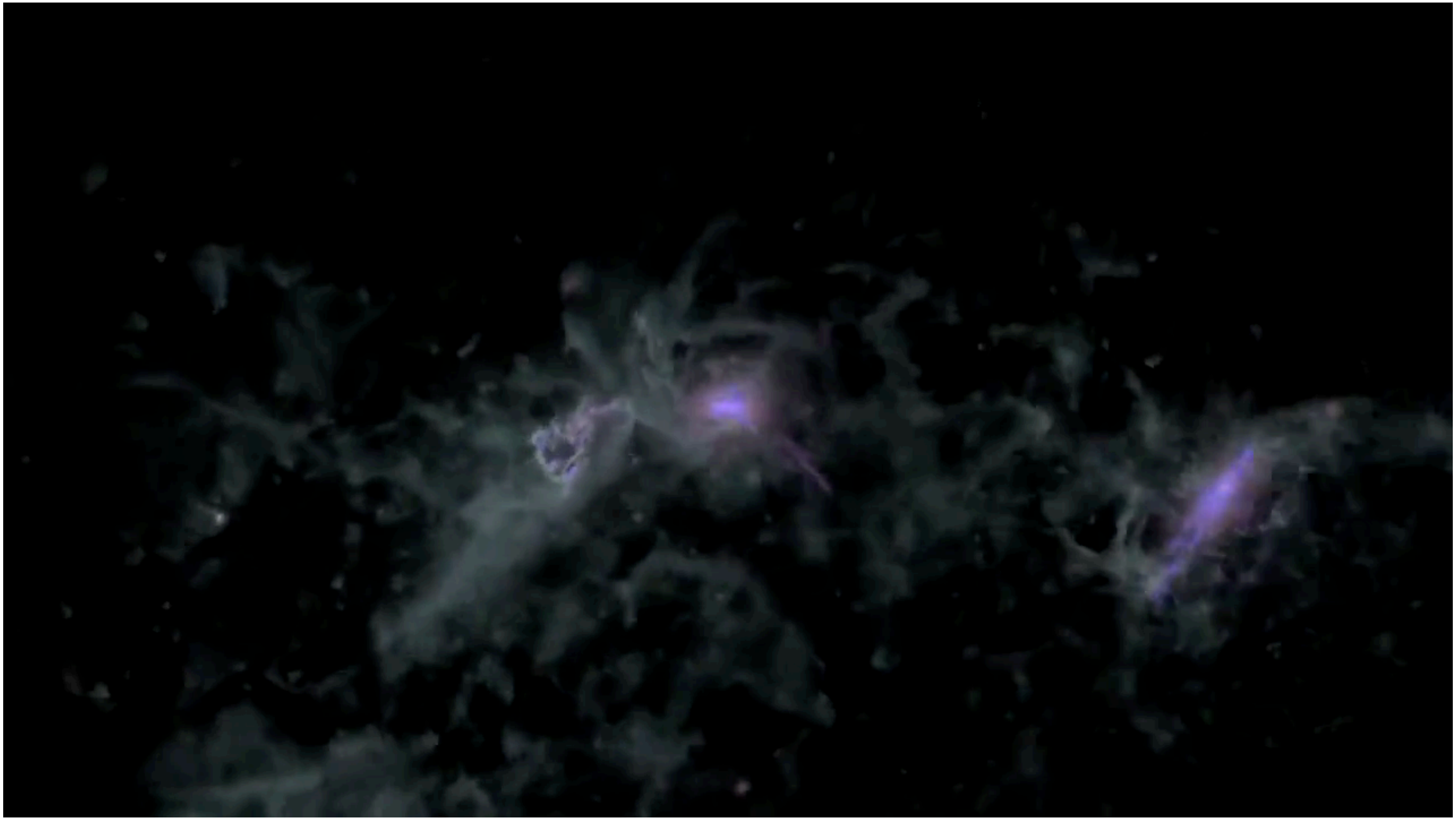
Add more constraints to provide controls.

Add energy terms to control qualities of motion.

Maybe add some control force terms.

Collisions can be annoying because they are discontinuities.

Example: Galaxy Simulation



Disk galaxy simulation, NASA Goddard

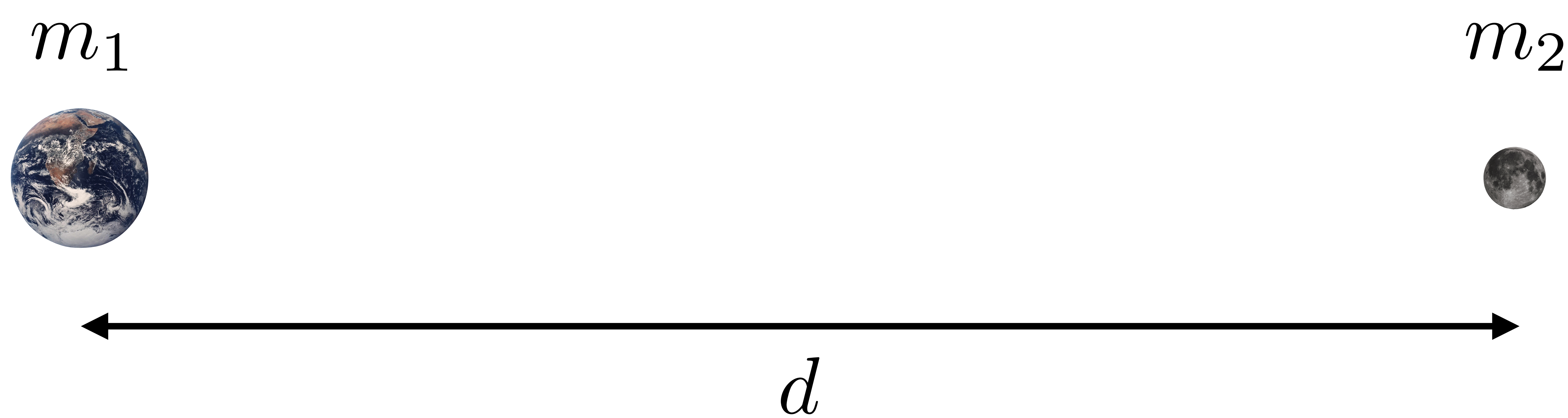
Gravitational Attraction

Newton's universal law of gravitation

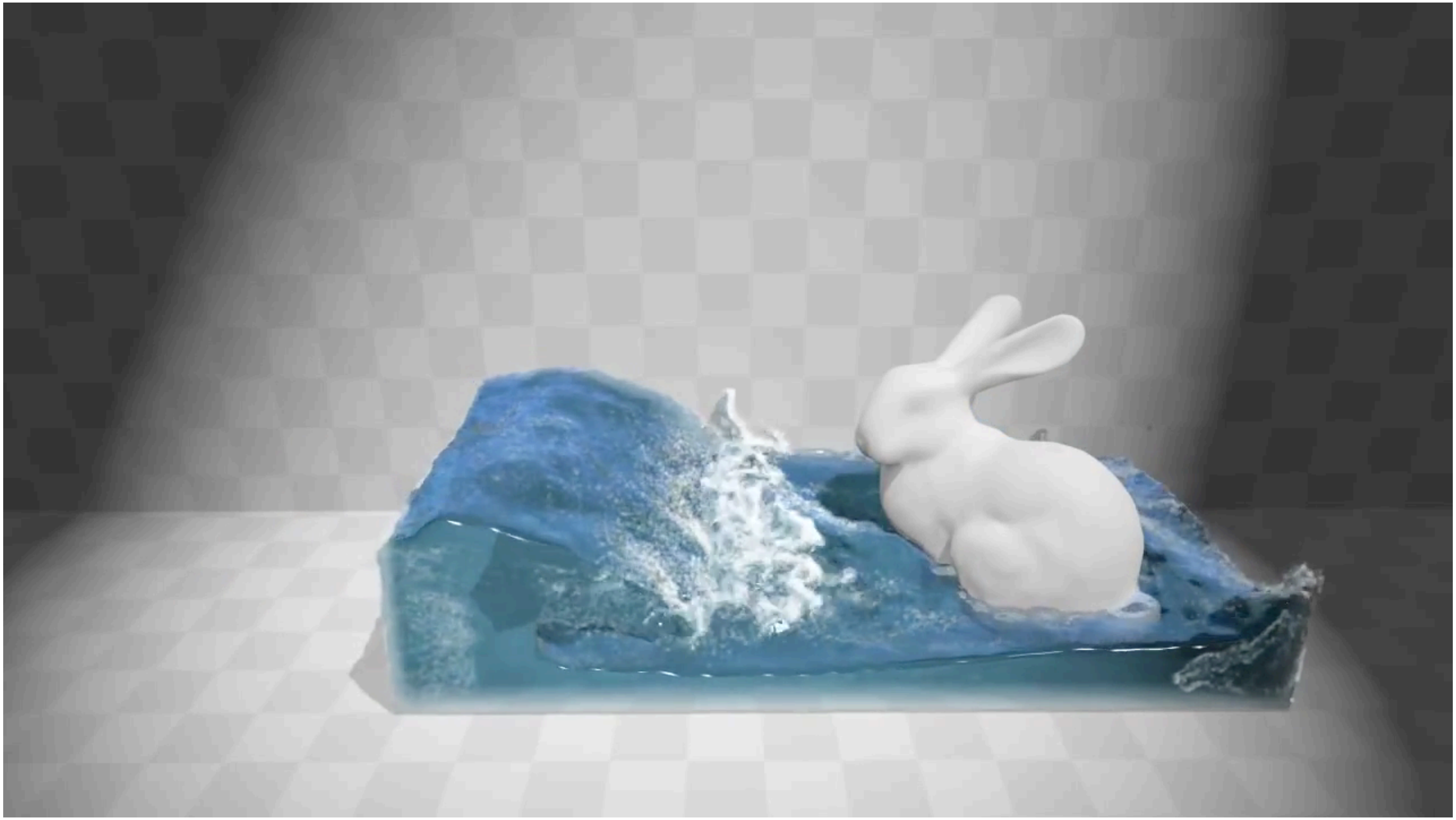
- Gravitational pull between particles

$$F_g = G\frac{m_1 m_2}{d^2}$$

$$G = 6.67428 \times 10^{-11}\,\mathrm{Nm^2kg^{-2}}$$

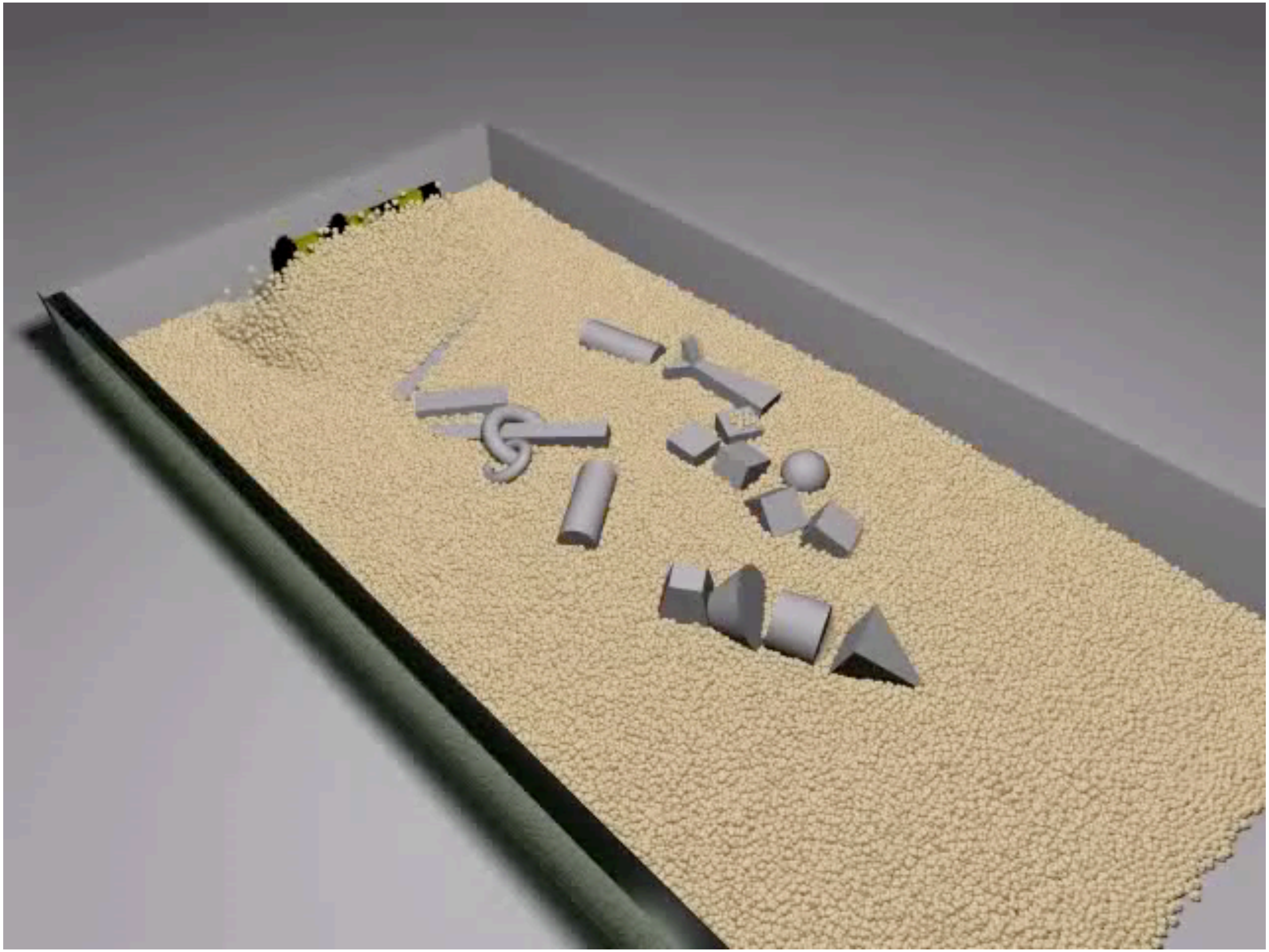


Example: Particle-Based Fluids



Macklin and Müller, Position Based Fluids , TOG 2013

Example: Granular Materials



Bell et al, "Particle-Based Simulation of Granular Materials"

Example: Flocking Birds

Example: Flocking Birds

Credit: Craig Reynolds (see http://www.red3d.com/cwr/boids/)

Simulated Flocking as an ODE

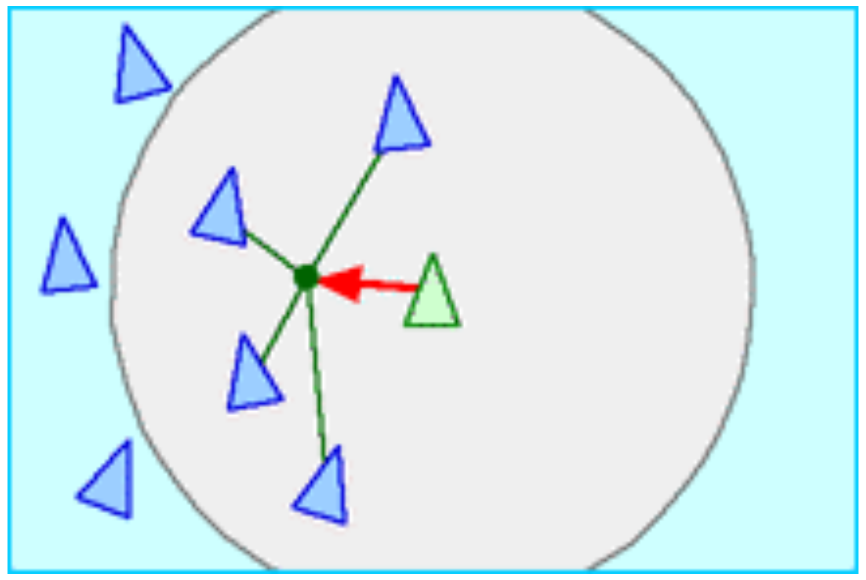
Model each bird as a particle

Subject to very simple forces:

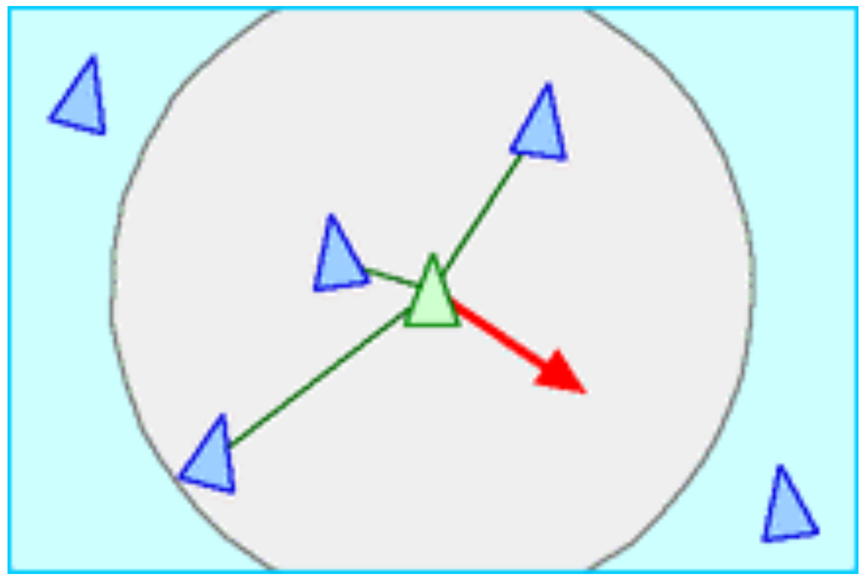
- <u>attraction</u> to center of neighbors
- <u>repulsion</u> from individual neighbors
- <u>alignment</u> toward average trajectory of neighbors

Simulate evolution of large particle system numerically

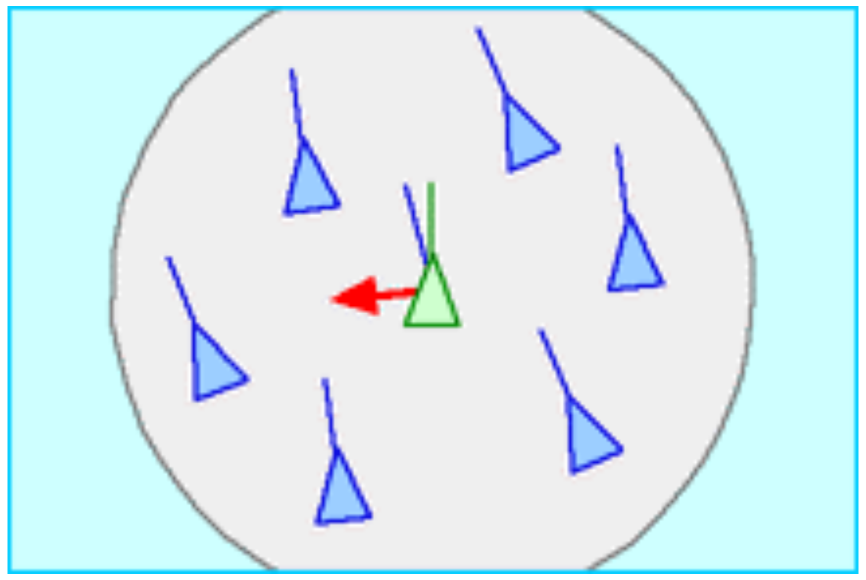
Emergent complex behavior (also seen in fish, bees, ...)



attraction



repulsion



alignment

Credit: Craig Reynolds (see http://www.red3d.com/cwr/boids/) Slide credit: Keenan Crane

Example: Crowds

Where are the bottlenecks in a building plan?

Example: Crowds + "Rock" Dynamics

Acknowledgments

This slide set contain contributions from:

- Kayvon Fatahalian
- David Forsyth
- Pat Hanrahan
- Angjoo Kanazawa
- Steve Marschner
- Ren Ng
- James F. O'Brien
- Mark Pauly