

*Fall 2014*

CSCI 420: **Computer Graphics**

# 14.1 Physically Based Simulation II

## Mass-Spring Systems



Hao Li

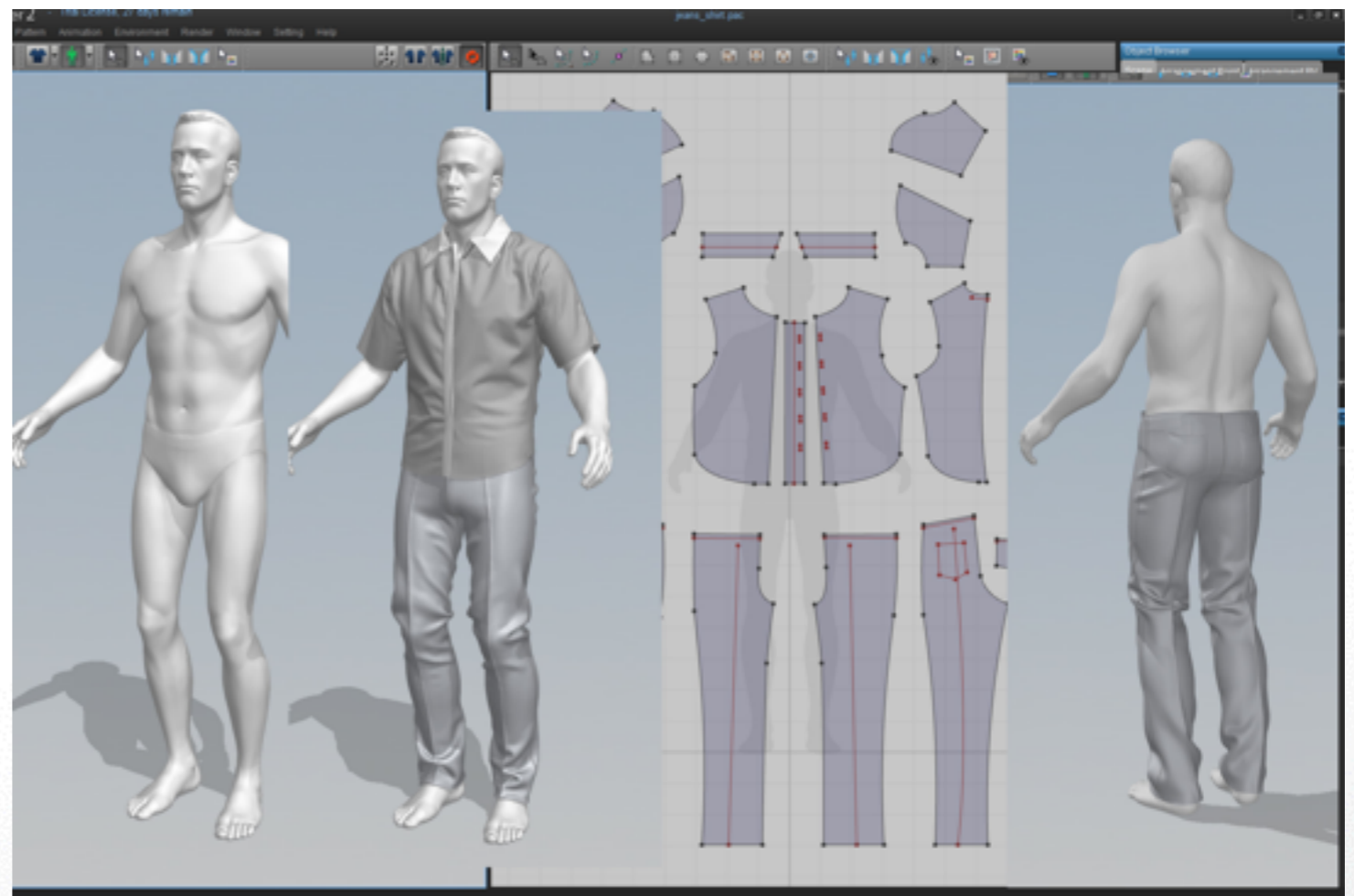
<http://cs420.hao-li.com>

# Mass-Spring Systems

- **The 101 of Physics Simulation**
- What do we want to simulate? **Deformable Objects**
- Design a model. **Mass points + springs.**
- Write differential equations. **Newton's 2nd Law (Hooke)**
- Discretize equations. **Integration methods for ODEs**
- Add interaction. **Collision detection + response**
- Simulate!

# Mass-Spring Systems

- Simulation of cloth based on deformable surfaces (Polygonal mesh)
- Realistic simulation of cloth with different fabrics such as wool, cotton, or silk for garment design



# Facial Animation

- Simulation of facial expressions based on deformable surfaces/volumes/muscles
- Animation of face models from speech and mimic parameters



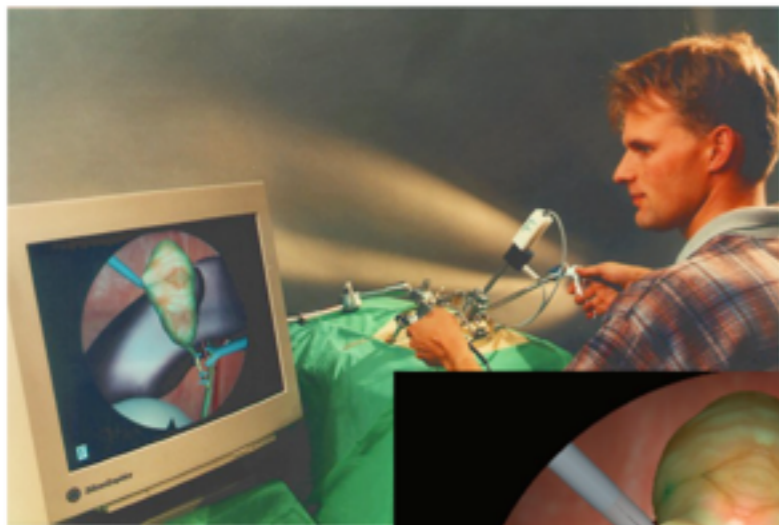
Thalmann



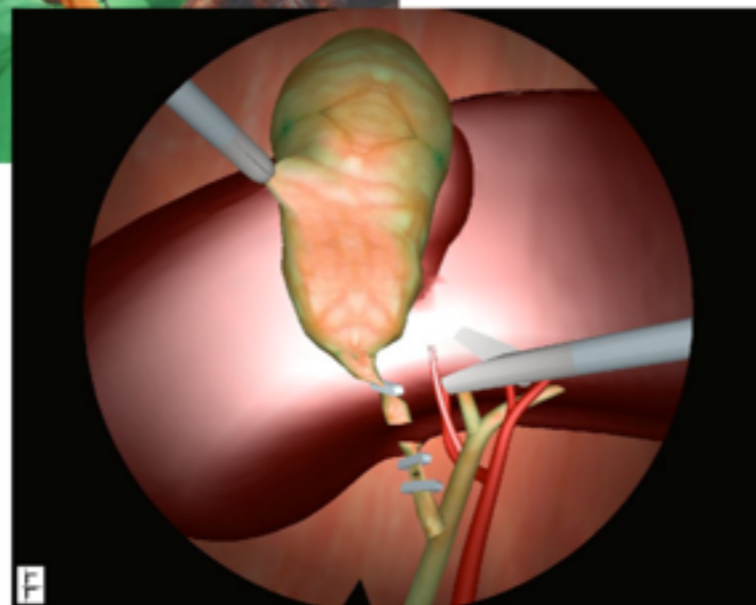
Terzopoulos

# Medical Simulation

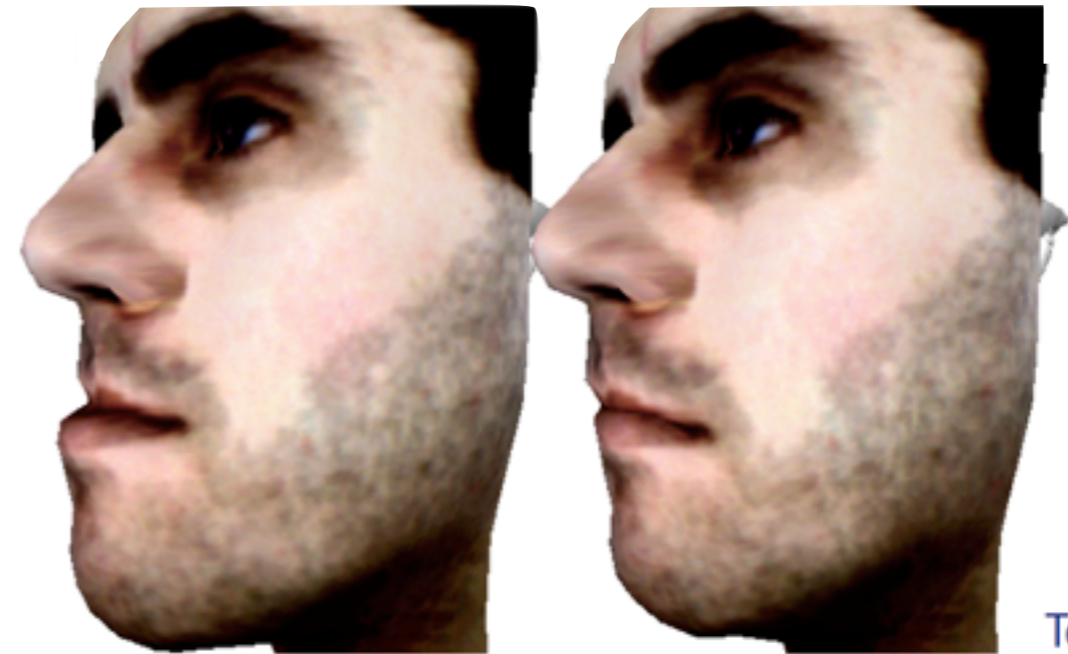
- Simulation of deformable soft tissue
- Surgical planning
- Medical training



Virtual endoscopy



Kuehnafel



Teschner

Prediction of the surgical outcome in craniofacial surgery

# Overview

- Model and Physics
- Implementation Hints
- Time-Discretization
- Collision Response
- (Simulation Loop)

# Mass-Point System

- Discretization of an object into mass points (gas, fluid, elastic object, inelastic object)
- System with multiple mass centers (Planetary System)
- Interaction between points  $i$  and  $j \neq i$  based on internal forces  $\mathbf{F}_{ij}^{\text{int}}$
- All other forces at point  $i$  are external forces  $\mathbf{F}_i^{\text{ext}}$
- Overall force  $\mathbf{F}_i = \mathbf{F}_{ij}^{\text{int}} + \mathbf{F}_i^{\text{ext}}$

$$\mathbf{F}_{ij}^{\text{int}} = -\mathbf{F}_{ji}^{\text{int}} \quad \sum_i \sum_j \mathbf{F}_{ij}^{\text{int}} = 0$$

# Mass-Point System

- Discretization of an object into mass points
- Representation of forces between masses by springs
- Computation of dynamics



# Mass-Points



Object sampled using mass points

Mass of object:  $M$

Number of points:  $n$

Mass of each point:  $m = M/n$

(if uniformly distributed)

Simulate the motion of each mass point

# Physically-based Equations

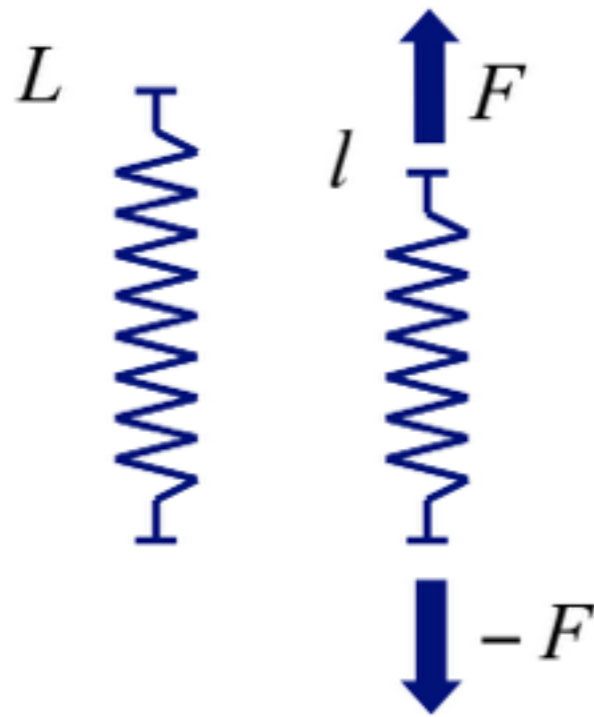
Equations that describe the behavior of the system (i.e. the mass points)

Physically-based model:  
Newton's 2<sup>nd</sup> Law

$$\sum \mathbf{F}_i^{\text{int}} + \mathbf{F}_i^{\text{ext}} = m\mathbf{a}_i$$

Next: Model the forces

# Elastic Forces: Springs



Spring stiffness is denoted as  $k$   
Initial spring length  $L$   
Current spring length  $l$

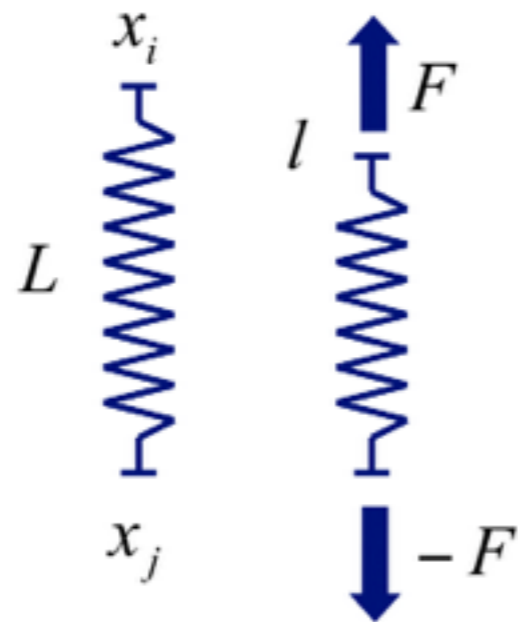
Deformation linear w.r.t. force:

$$F = -k(l - L) \quad \text{Hooke's Law}$$

**Elasticity:** Ability of a spring to return to its initial form when the deforming force is removed.

Simple mechanism for internal forces.

# Elastic Energies



Elastic energy:

$$E = \frac{1}{2}k(l - L)^2$$

Force = - Partial Derivative (Gradient)

$$F_i = -\frac{\partial E}{\partial x_i}$$

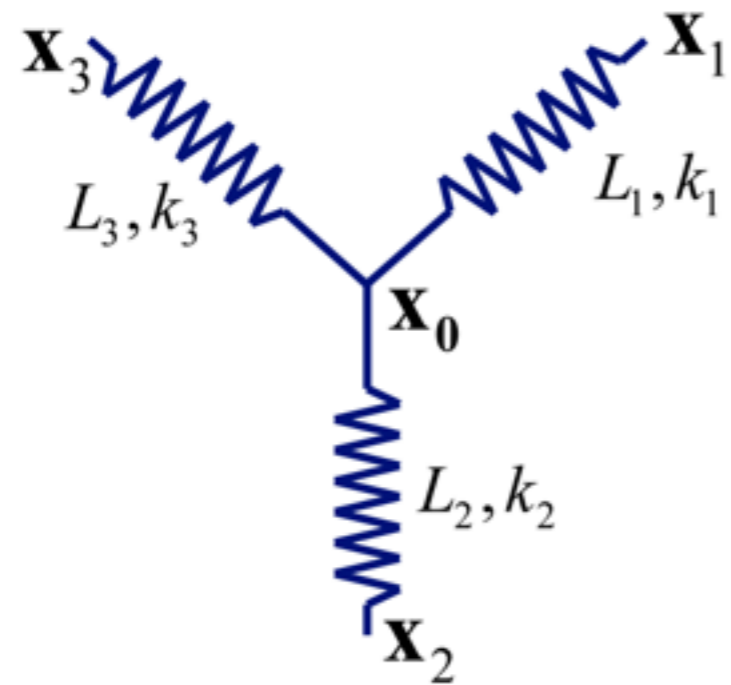
Force in vector notation:

$$F_i = -k(l - L)\frac{\mathbf{x}_i - \mathbf{x}_j}{l}$$

*Force-centered view versus energy-centered view*

# Forces at a Mass Point

## Internal forces $\mathbf{F}^{\text{int}}$



## External forces $\mathbf{F}^{\text{ext}}$

Gravity

Contact forces

All forces that are not caused by springs

## Resulting force at point $\dot{\mathbf{x}}$

$$\mathbf{F}_0^{\text{int}} = - \sum_{i \in \{1,2,3\}} k_i (l_i - L_i) \frac{\mathbf{x}_i - \mathbf{x}_0}{l_i}$$

$$\mathbf{F}_i = \mathbf{F}_i^{\text{int}} + \mathbf{F}_i^{\text{ext}}$$

# Dissipative Forces

## Dissipative forces

Damping  
Friction

$$\mathbf{F}^{damping}(t) = -\gamma \cdot \mathbf{v}(t)$$

# System Equations

**Equation of Motion** for one mass point (3 eqs.)

$$m_i \frac{d^2 \mathbf{x}_i(t)}{dt^2} = \mathbf{F}_i^{\text{int}}(t) + \mathbf{F}_i^{\text{ext}}(t)$$

**Equation of Motion** for a system of mass points (3n eqs.)

$$\mathbf{M} \frac{d^2 \mathbf{X}(t)}{dt^2} = \mathbf{F}^{\text{int}}(t) + \mathbf{F}^{\text{ext}}(t)$$

**M** is a diagonal matrix

# System Equations

Incorporation of **damping**

$$\mathbf{M} \frac{d^2 \mathbf{X}(t)}{dt^2} + \mathbf{D} \frac{d\mathbf{X}(t)}{dt} = \mathbf{F}^{\text{int}}(t) + \mathbf{F}^{\text{ext}}(t)$$



# Overview

- Model and Physics
- Implementation Hints
- Time-Discretization
- Collision Response
- (Simulation Loop)

# Elastic Spring

```
class SPRING
{
    public:
        POINT *point1;
        POINT *point2;
        float stiffness; // k
        float initialLength; // L
        float currentLength; // l

        . . .
}
```

# Mass Point

```
class POINT
{
    public:
        float mass;
        float position[3];
        float velocity[3];
        float force[3];
        float damping;
        . . .
}
```

# Force Computation

```
for all points
    point_i.ClearForce()
    point_i.AddGravityForce()

//Add other external forces

for all springs
    spring_i.ComputeElasticForce()
    spring_i.AddForceToEndPoints()
```

# Overview

- Model and Physics
- Implementation Hints
- Time-Discretization
- Collision Response
- (Simulation Loop)

# System Equations

System of  $3n$  2<sup>nd</sup> order Ordinary Differential Equations (ODE)

$$\mathbf{M} \frac{d^2 \mathbf{X}(t)}{dt^2} + \mathbf{D} \frac{d\mathbf{X}(t)}{dt} = \mathbf{F}^{\text{int}}(t) + \mathbf{F}^{\text{ext}}(t)$$

One 2<sup>nd</sup> order ODE (1-dimensional problem)

$$m \frac{d^2 x(t)}{dt^2} + \gamma \frac{dx(t)}{dt} = F(t)$$

Initial value problem:  $x(0)$  and  $v(0)$  are known

# Solution

a) Analytical solution (if we care about the exact state at time  $t$ )

b) **Discrete** solution

- Graphics: the goal is to **display** the state at  $t_i$

- Find solution at discrete time instants  $t_i$ , assuming that we know previous solutions  $t_{i-1}$ ,  $t_{i-2}$ , etc.

- We do not care about the steady state error, but we want **plausible behavior** and **response to external forces**

# Problem

- We have:
  - Initial position  $x$
  - Initial velocity  $v$
  - 2nd derivative of position  $x$  with respect to time

$$\frac{d^2 \mathbf{x}_i(t)}{dt^2} = \frac{\mathbf{F}_i(t) - \gamma \mathbf{v}_i(t)}{m_i}$$

- **Goal:** Computation of position  $x$  over time



# Numerical Integration Methods

- Explicit Integration
  - Euler
  - Leapfrog
  - Heun
  - Midpoint
  - Runge-Kutta methods
- Implicit Integration
  - Backward Euler
- Predictor-Corrector methods
  - Gear
- Methods for higher order ODEs
  - Verlet
  - Beeman
- Variable time-step methods

# Numerical Integration Methods

- Reduction of a second-order ODE to two coupled first-order ODEs.

$$m_i \frac{d^2 \mathbf{x}_i(t)}{dt^2} + \gamma \frac{d\mathbf{x}_i(t)}{dt} = \mathbf{F}_i$$



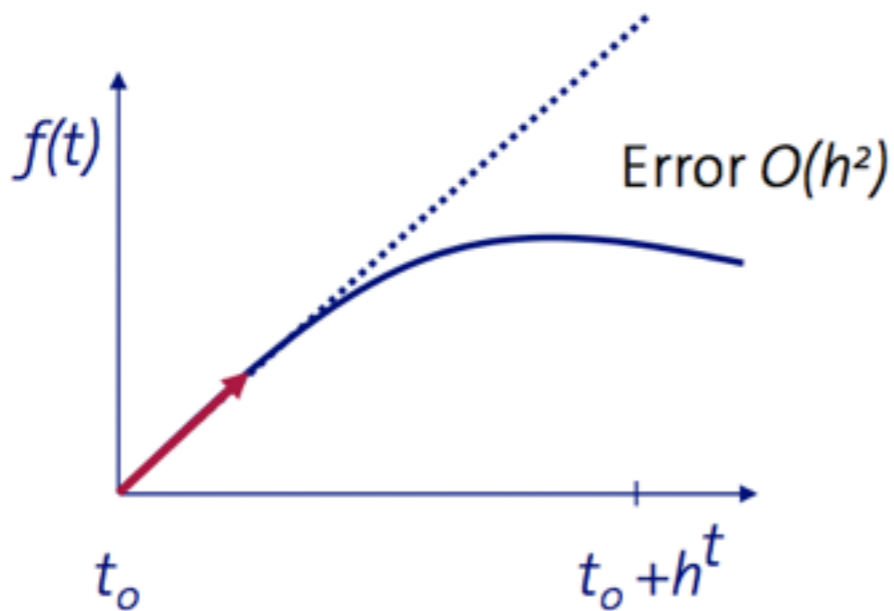
$$\frac{d\mathbf{x}_i(t)}{dt} = \mathbf{v}_i(t) \quad \frac{d\mathbf{v}_i(t)}{dt} = \frac{\mathbf{F}_i(t) - \gamma \mathbf{v}_i(t)}{m_i}$$

velocity

acceleration

8

# Explicit Integration



- Initial value  $f(t_0)$
- Compute the derivative at  $t_0$
- Move from  $t_0$  to  $t_0+h$  using the derivative at  $t_0$

## *Euler Method*

Leonard Euler:  
1707 (Basel) – 1783 (St. Petersburg)

# Explicit Integration

$$f(t_0 + h) = f(t_0) + h \cdot f'(t_0) + \frac{h^2}{2} f''(t_0) + \dots$$

Taylor series

$$f(t_0 + h) = f(t_0) + h \cdot f'(t_0) + O(h^2)$$

$$f(t_0 + h) \cong f(t_0) + h \cdot f'(t_0)$$

Euler method

# Explicit Integration

$$\mathbf{x}'(t) = \mathbf{v}(t) \quad \mathbf{v}'(t) = \frac{\mathbf{F}(t) - \gamma \mathbf{v}(t)}{m}$$

Start with initial values	$\mathbf{x}(t_0) = \mathbf{x}_0 \quad \mathbf{v}(t_0) = \mathbf{v}_0$
Compute	$\mathbf{v}'(t_0) \quad \mathbf{x}'(t_0)$
Assume	$\mathbf{v}'(t) = \mathbf{v}'(t_0) \quad \mathbf{x}'(t) = \mathbf{x}'(t_0) \quad t_0 \leq t \leq t_0 + h$
Compute	$\mathbf{x}(t_0 + h) = \mathbf{x}(t_0) + h\mathbf{x}'(t_0) = \mathbf{x}(t_0) + h\mathbf{v}(t_0)$
Compute	$\mathbf{v}(t_0 + h) = \mathbf{v}(t_0) + h\mathbf{v}'(t_0) = \mathbf{v}(t_0) + h \frac{\mathbf{F}(t_0) - \gamma \mathbf{v}(t_0)}{m}$

$\mathbf{F}(t)$  is computed from  $\mathbf{x}(t)$  and external forces!

# Error Accumulation

$$\mathbf{x}'(t) = \mathbf{v}(t) \quad \mathbf{v}'(t) = \frac{\mathbf{F}(t) - \gamma \mathbf{v}(t)}{m}$$

Euler step from  $t_0$  to  $t_0+h$

$$\mathbf{x}(t_0 + h) = \mathbf{x}(t_0) + h\mathbf{v}(t_0) \quad \mathbf{v}(t_0 + h) = \mathbf{v}(t_0) + h \frac{\mathbf{F}(t_0) - \gamma \mathbf{v}(t_0)}{m}$$

$$\mathbf{x}(t_0 + 2h) = \mathbf{x}(t_0 + h) + h\mathbf{v}(t_0 + h)$$

Euler step  
from  $t_0+h$  to  $t_0+2h$

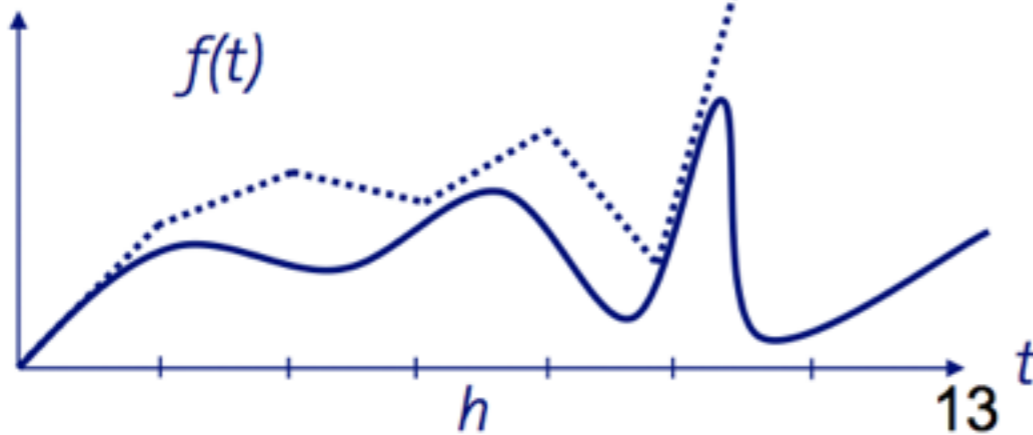
$$\mathbf{v}(t_0 + 2h) = \mathbf{v}(t_0 + h) + h \frac{\mathbf{F}(t_0 + h) - \gamma \mathbf{v}(t_0 + h)}{m}$$

# Problems

- Numerical integration is inaccurate.

$$\underbrace{f(t+h) = f(t) + f'(t)h}_{\text{Euler step}} + \boxed{O(h^2)}_{\text{Error}}$$

- Inaccuracy can cause instability.



Error

$$0 \leq e < \frac{h^2}{2} \cdot f''(t_e), \quad t_e \in [t, t+h]$$

# Improving Accuracy - Leap Frog

$$\mathbf{v}(t + h/2) = \mathbf{v}(t - h/2) + h \cdot \mathbf{a}(t)$$

$$\mathbf{x}(t + h) = \mathbf{x}(t) + h \cdot \mathbf{v}(t + h/2)$$

Error  $O(h^3)$

time step  $h$  is significantly larger compared to Euler

## Implementation

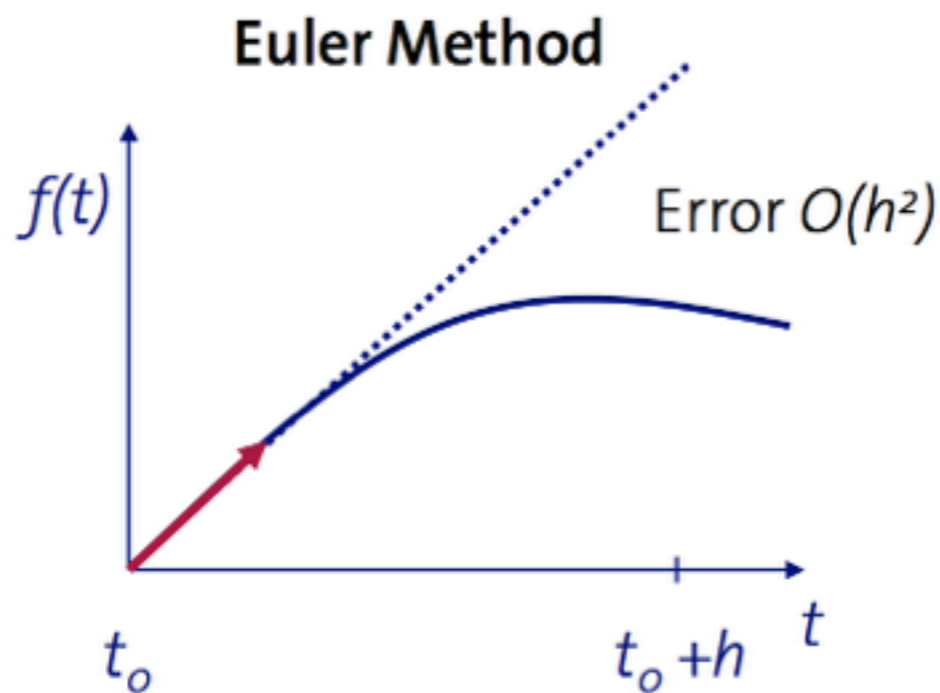
Euler	Leapfrog
<pre>... addForces(); // F(t) positionEuler(h); // x=x(t+h)=x(t)+hv(t) velocityEuler(h); // v=v(t+h)=v(t)+ha(t) ...</pre>	<pre>initV() // v(o) = v(o) - h/2a(o) ... addForces(h); // F(t) velocityEuler(h); // v=v(t+h)=v(t)+ha(t) positionEuler(h); // x=x(t+h)=x(t)+hv(t+h) ...</pre>

(In practice, it is irrelevant that velocities are computed at mid time steps)

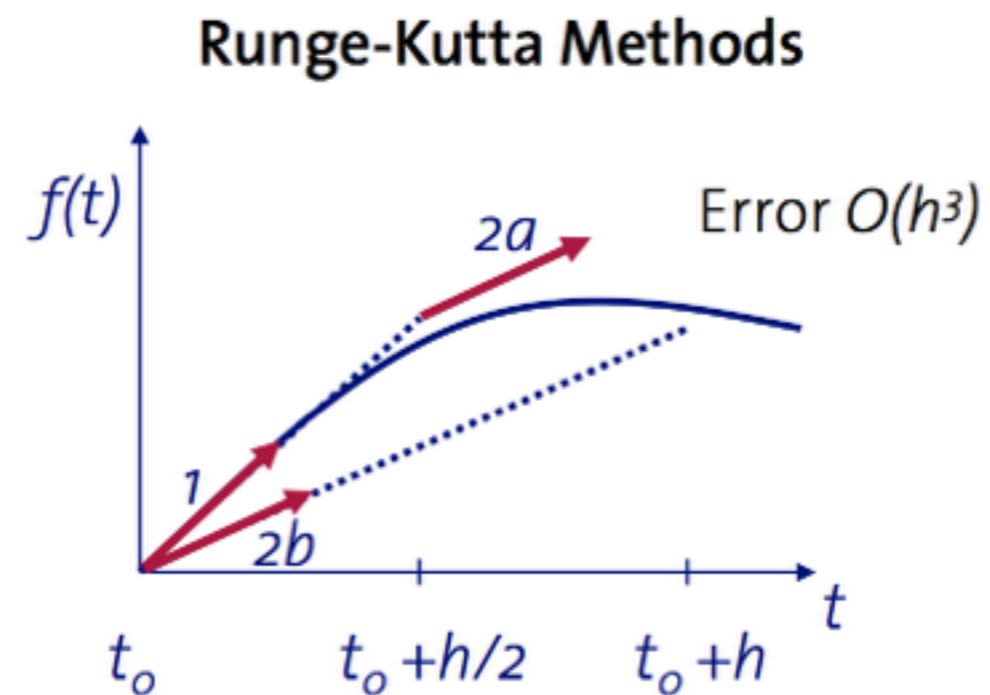


# Improving Accuracy - Runge Kutta

## 2nd order (midpoint method)

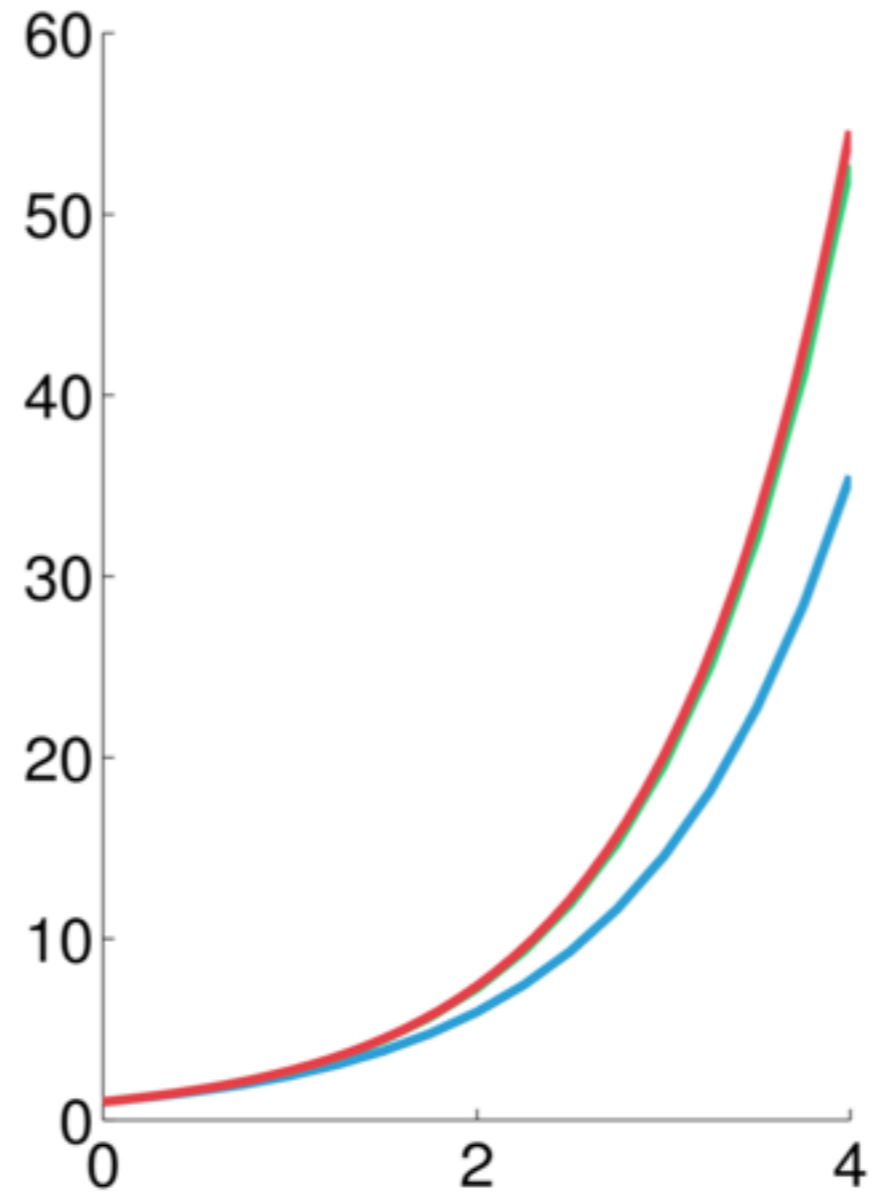
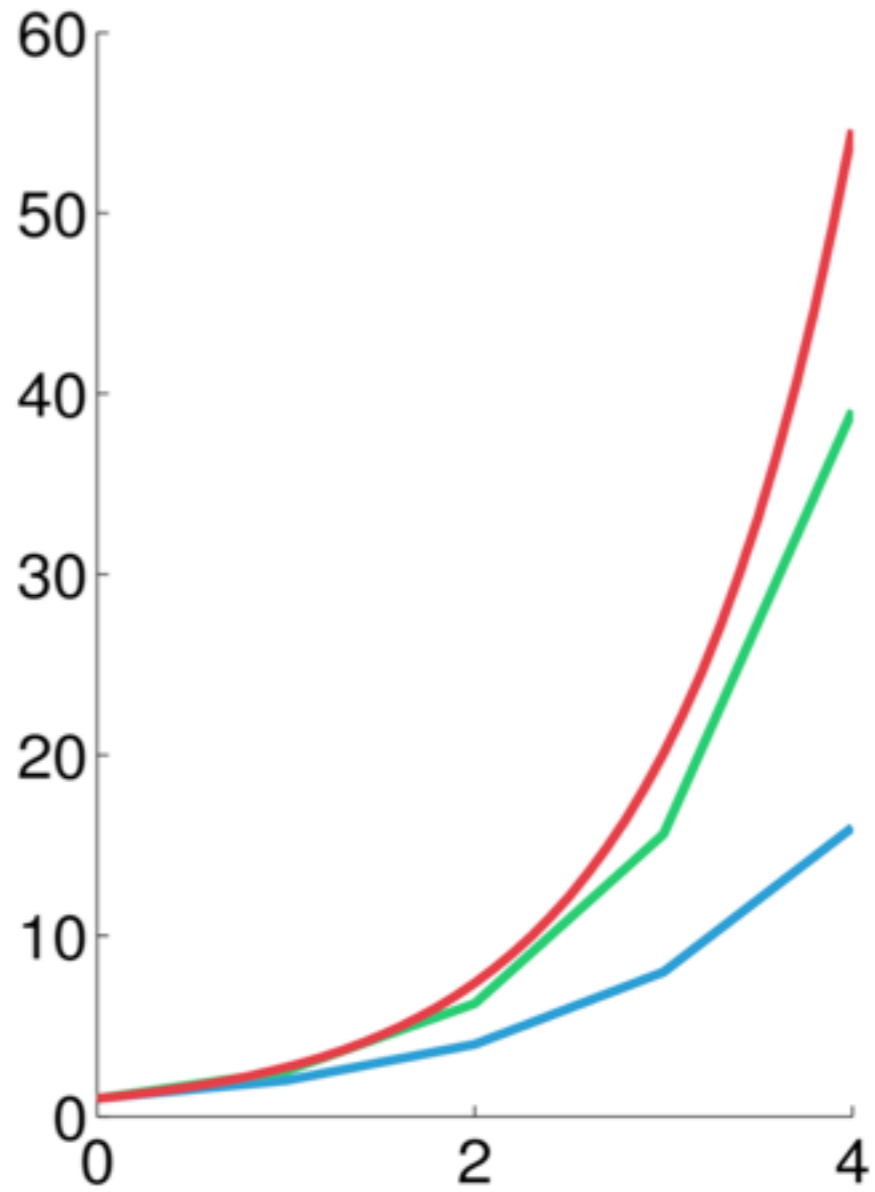


- Compute the derivative at  $t_0$
- Move from  $t_0$  to  $t_0+h$  using the derivative at  $t_0$



- Compute the derivative at  $t_0$
- Move to  $t_0+h/2$
- Compute the derivative at  $t_0+h/2$
- Move from  $t_0$  to  $t_0+h$  using the derivative at  $t_0+h/2$
- Second order R-K also called "midpoint"

# Midpoint vs Euler



- Green = Midpoint
- Blue = Euler
- $h=1$  vs.  $h=1/4$

# Implementation

## Euler Method

Straightforward:

- Compute spring forces
- Add external forces
- Update positions
- Update velocities

## Runge-Kutta Methods

- Compute spring forces
- Add external forces
- Compute **auxiliary** positions and velocities
  - once for second-order
  - three times for fourth-order
  - requires **additional data copies**
- Update positions
- Update velocities

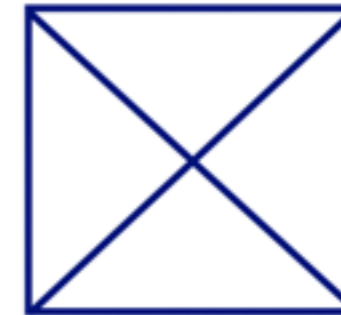
# Avoiding Instability

- No general solution to avoid instability for complex mass-point systems.
- A smaller time step increases the chance for stability.
- A larger time step speeds up the simulation.
- Parameters and topology of the mass-point system, and external forces influence the stability of a system.
- Increasing damping does not always help.

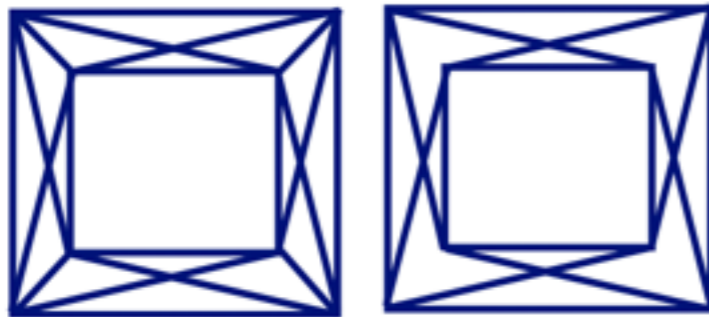
# Topology and Stability

- Stable model topologies with respect to deformation

not stable



stable  
but not  
general

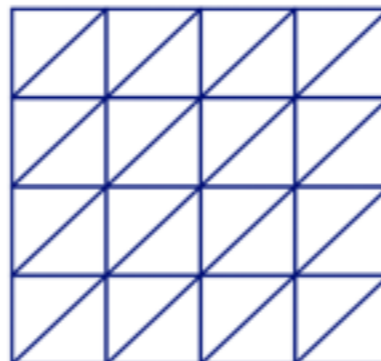




stable

can be generated automatically by copying the surface to an inner layer and connecting both – **layered model**

in the extreme case, consider the inner layer to be just a point

- Design problem



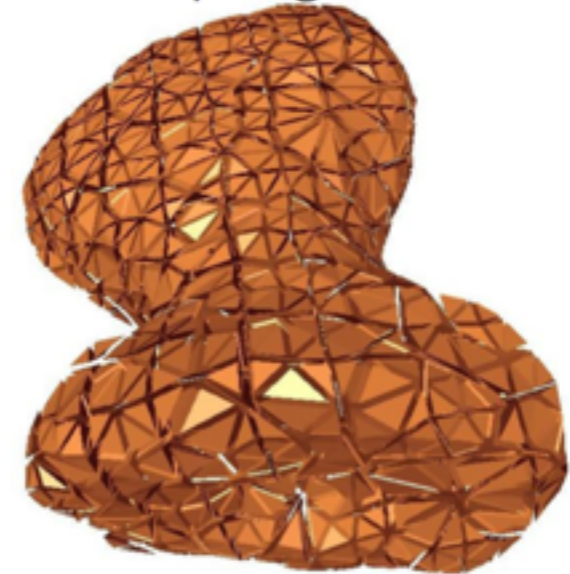
much more resistant in  direction than in  direction.

# Volumetric Models - Tet Meshes

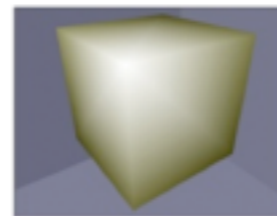
2949 mass points  
10257 tets  
15713 springs



1349 mass points  
4562 tets  
6888 springs

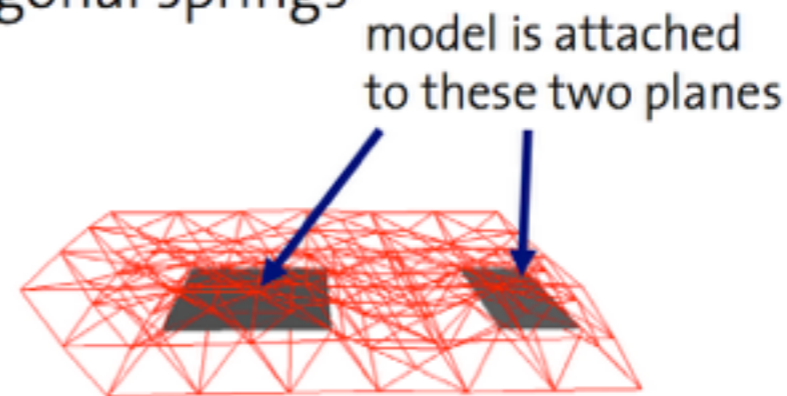
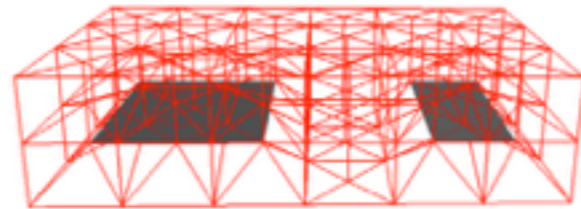


8 mass points  
5 tets

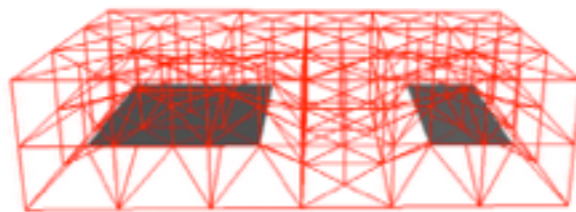


# Topology Ambiguity Problem

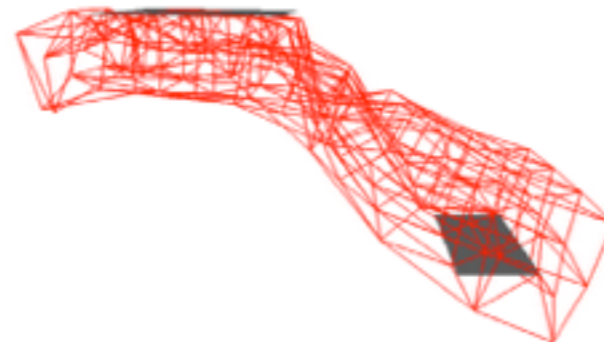
- Unappropriate topology without diagonal springs
- No force penalty for shearing



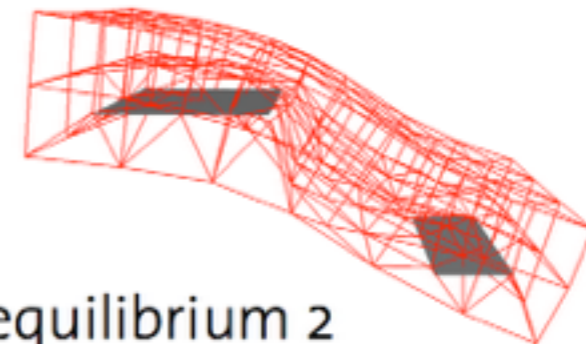
- Appropriate topology with diagonal springs
- However, self-collision problem, springs have no notion of volume



original



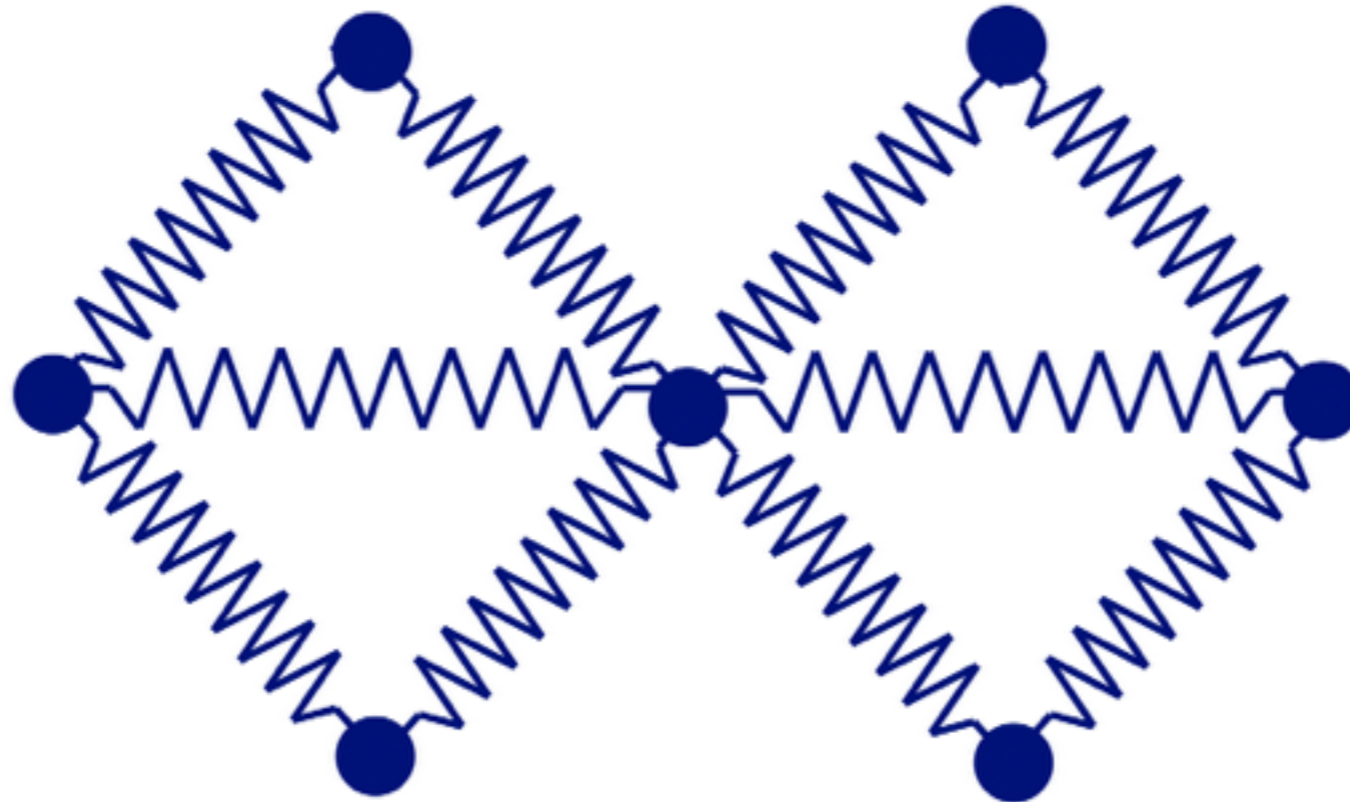
equilibrium 1



equilibrium 2

# Cloth Forces

- Types of forces in cloth: **stretch**, **bending**, **shear**
- Bending cannot be modeled with a simple network of springs

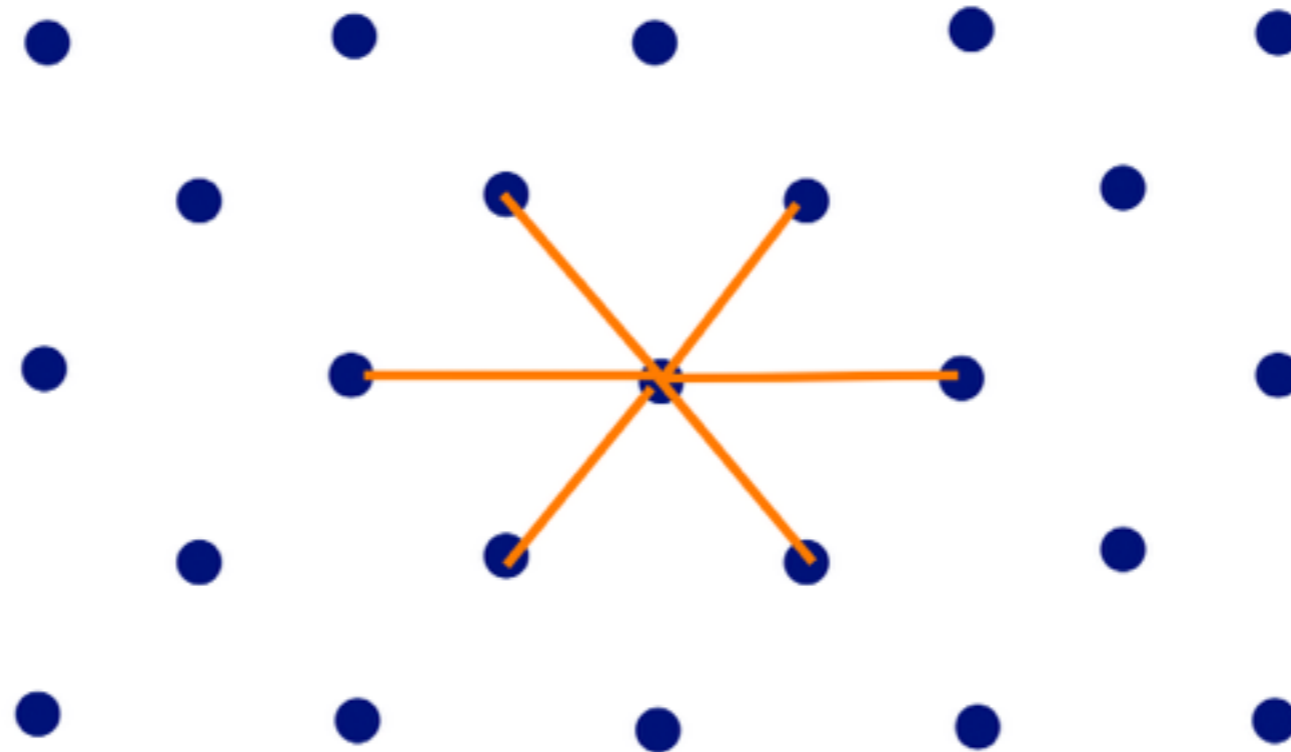




# Cloth Springs

- Combine level-1 and level-2 springs

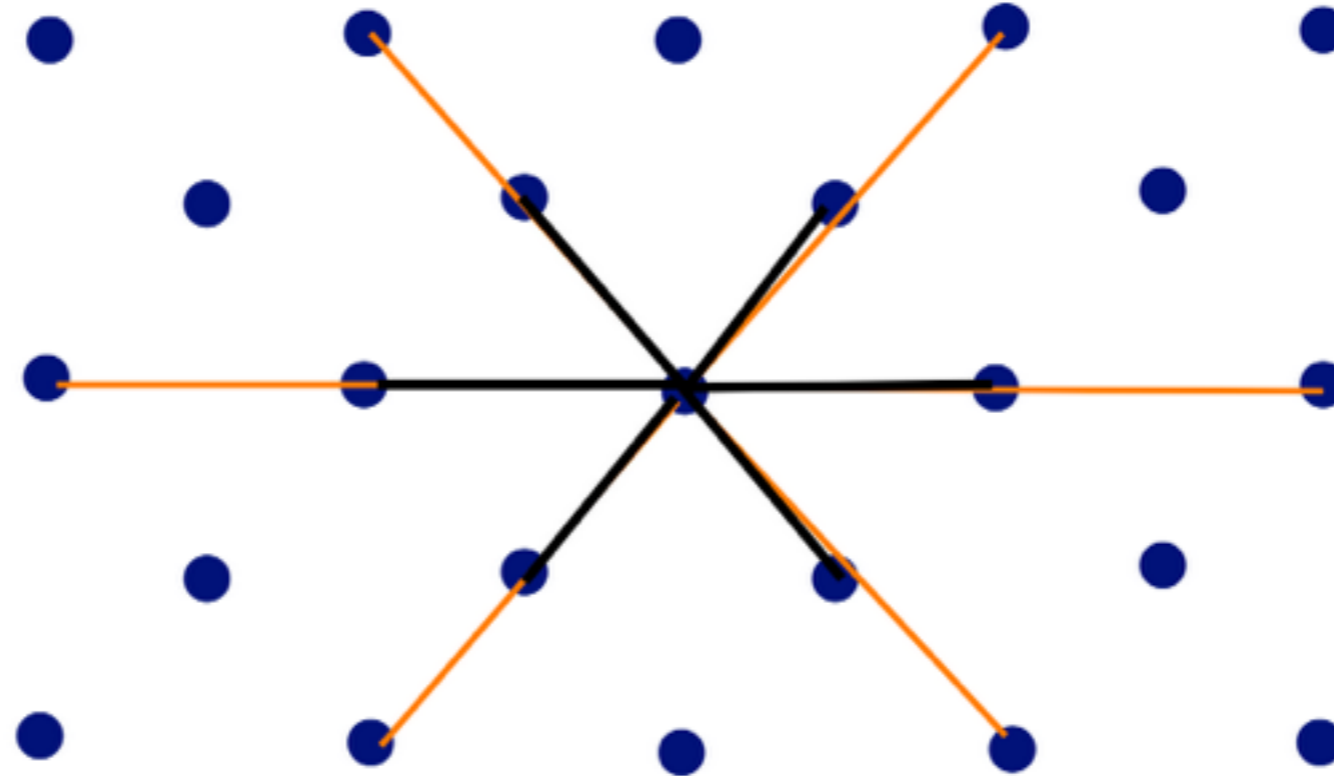
Level 1 for stretch



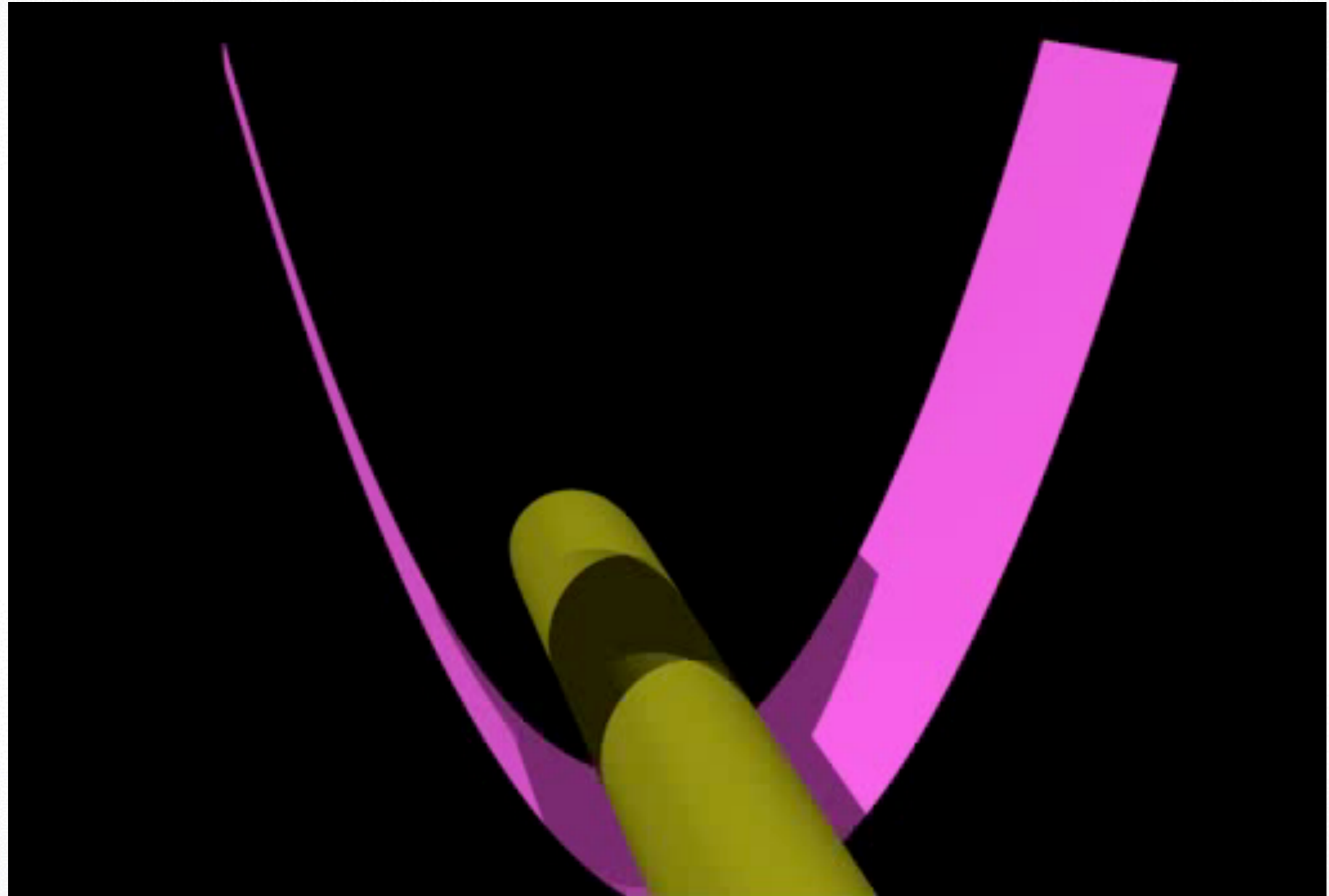
# Cloth Springs

- Combine level-1 and level-2 springs

Level 2 for bending



# Cloth Springs



<http://cs420.hao-li.com>

# Thanks!

