Cloth Animation

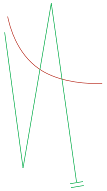
Christopher Twigg

March 4, 2003

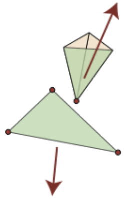
Outline

- Overview

- Models



- Integrating stiff systems

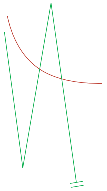


- Collision handling

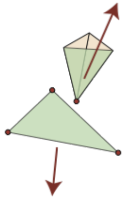
Outline

- Overview

- Models



- Integrating stiff systems



- Collision handling

What is cloth?

- 2 basic types: woven and knit
- We'll restrict to woven
  - Warp vs. weft

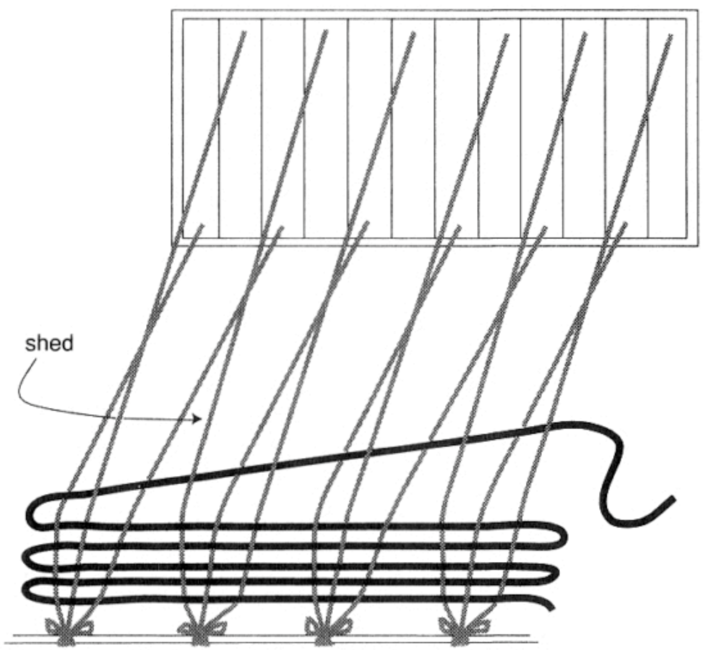
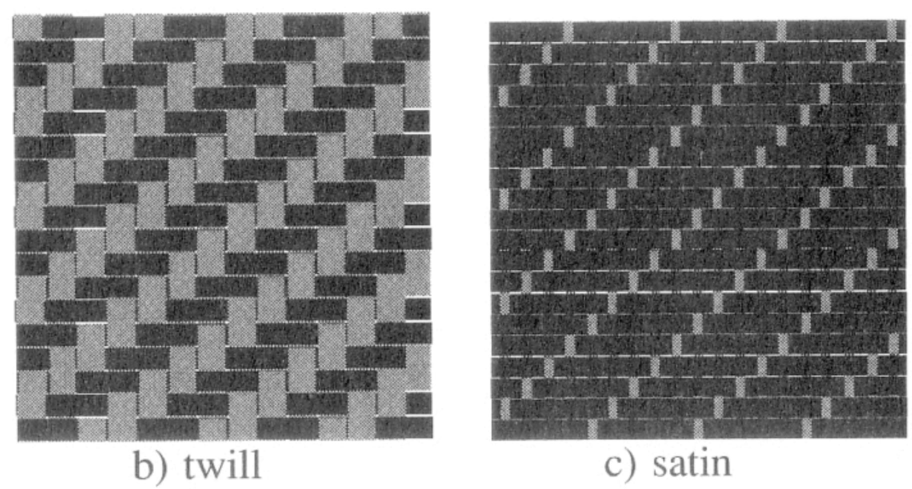


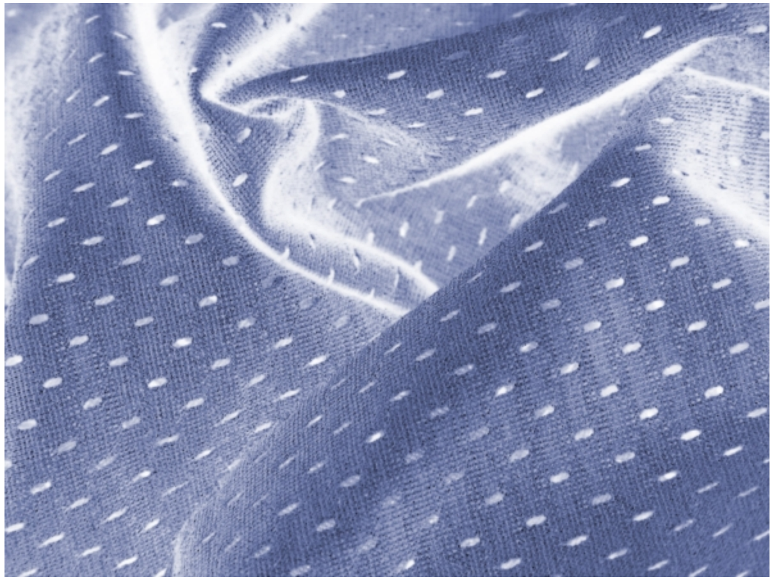
Figure 1.8. The weaving process.

House, Breen [2000]



What makes cloth special?

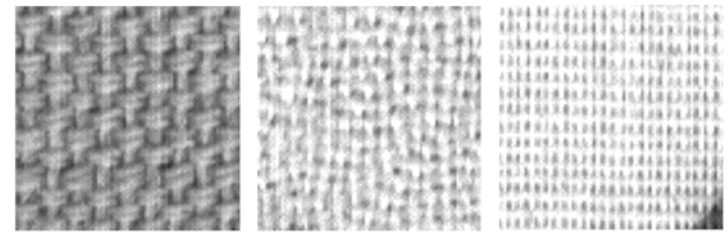
- Infinite number of varieties --
  - Thread type (wool, polyester, mixtures...)
  - Weave type (plain, twill, basket, satin...)
  - Weave direction (bias cut; warp vs. weft)
  - Seams (fashion design)
  - Hysteresis (ironed vs. crumpled in a suitcase)



From Ko, Choi [2002]

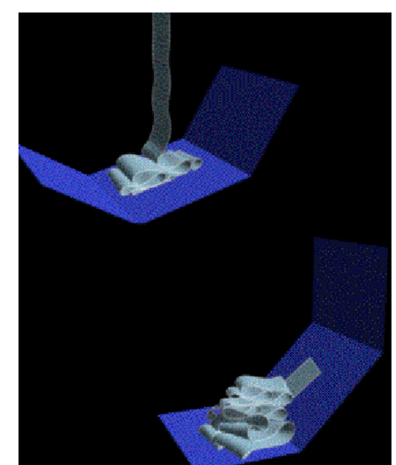
Challenges in cloth simulation

- Model
  - Complex microstructure
  - Realism
  - Simplicity
- Integrator
  - Dealing with stiffness
- Collision handling



a) 100% cotton b) 100% wool c) polyester/cotton

Breen, House, Wozny [1994]

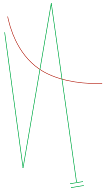


Vollino (sic), Courchesne, Magnenat-Thalmann [1998]

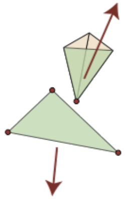
Outline

- Overview

- Models

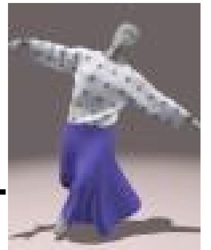


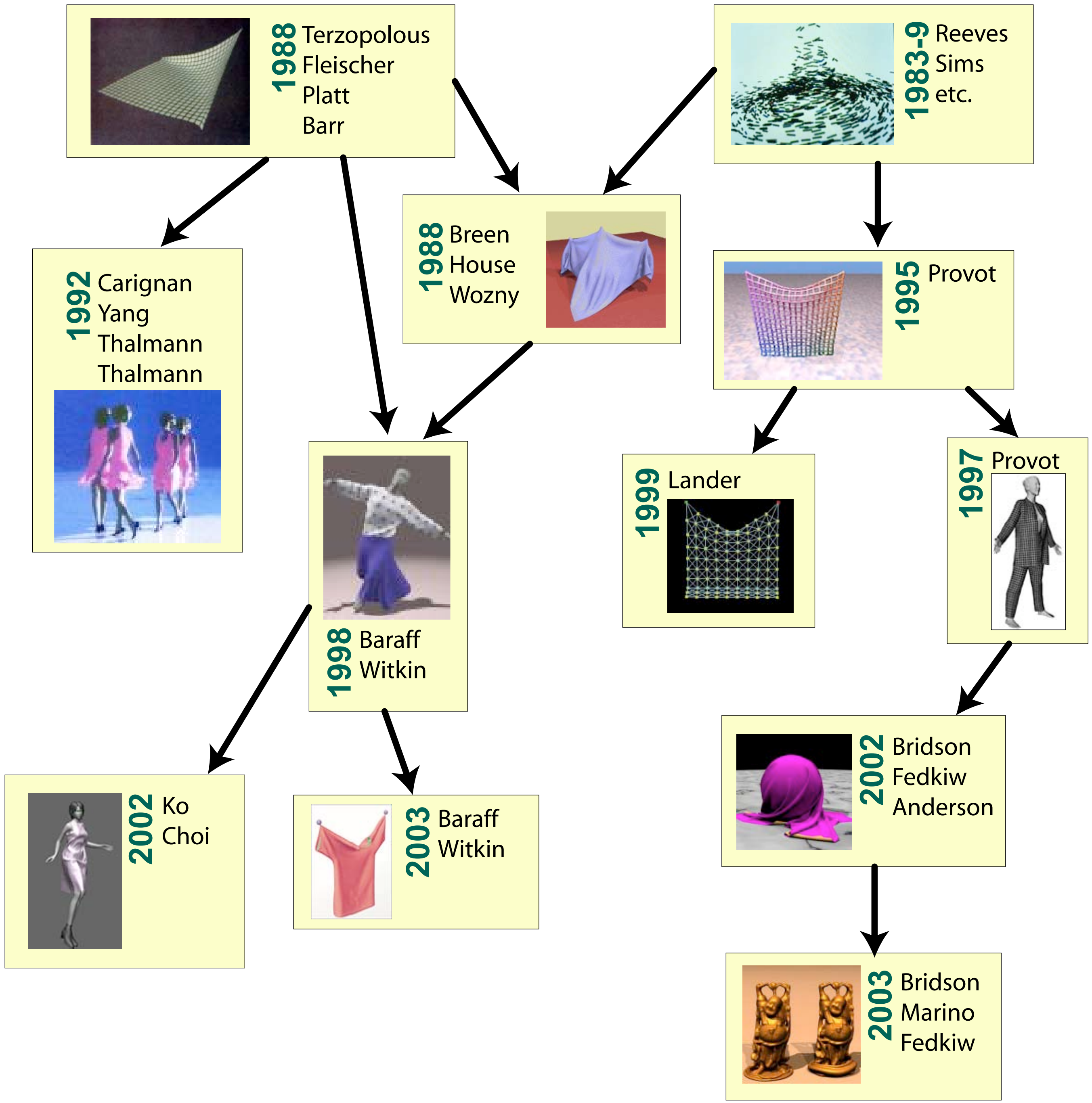
- Integrating stiff systems



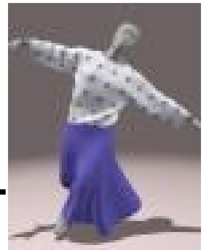
- Collision handling

An (abbreviated) cloth bestiary

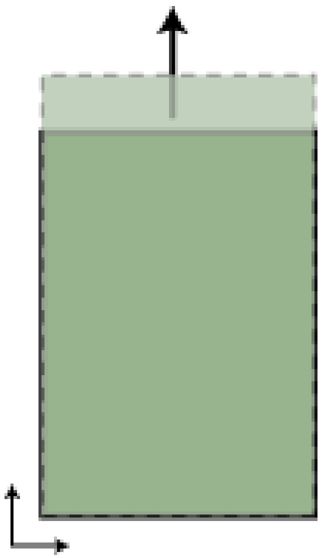




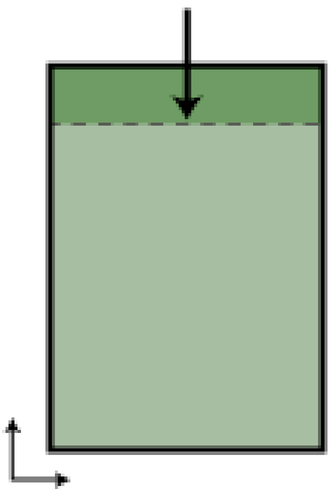
Cloth modeling basics



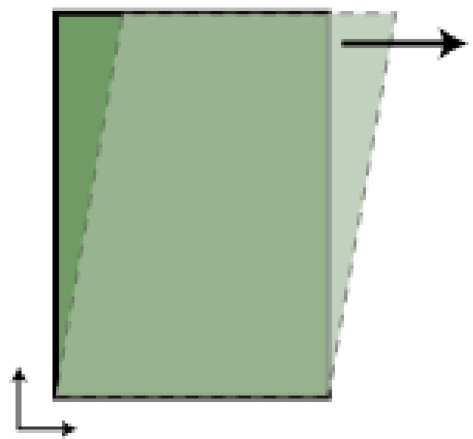
- In general, cloth resists motion in 4 directions:



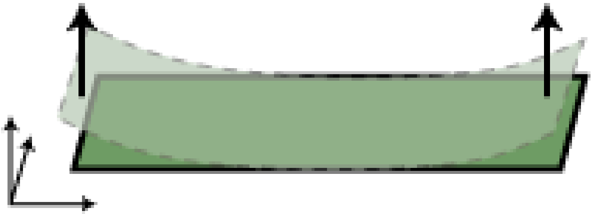
In-plane stretch



In-plane compression



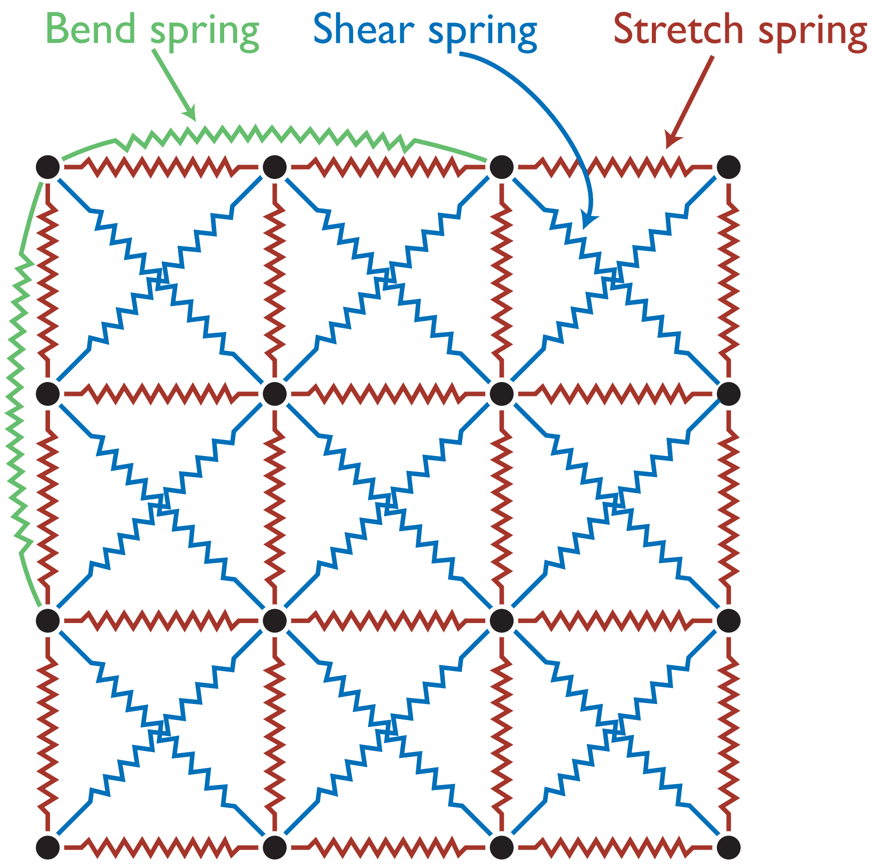
In-plane shear (trellising)



Out-of-plane bending

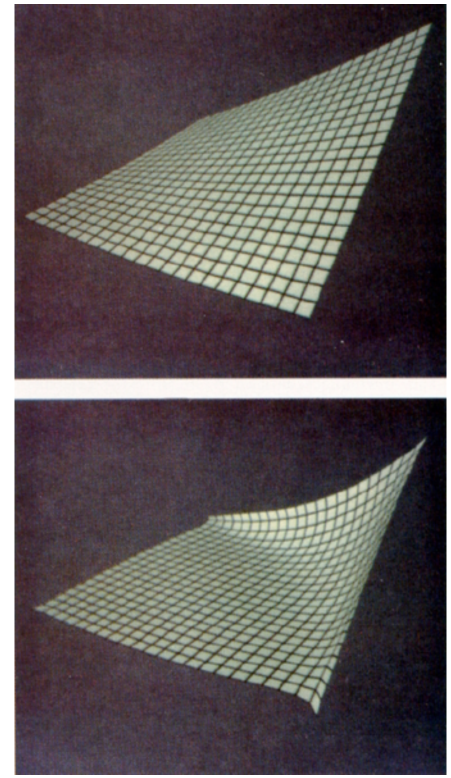
A basic mass-spring model

- Simple spring-mass system due to Provot [1995]
- You already know how to implement this



Early continuum models

- Various modifications to deal with collisions, etc.



Terzopolous, Platt, Barr, Fleischer [1987]

Carignan, Yang, Thalmann, Thalmann [1992]

Generally not used in practice (although many models use ideas from continuum physics)

Particle-based methods

- Breen [1992]: energy-based model

$$U_i = U_{repel_i} + U_{stretch_i} + U_{bend_i} + U_{trellis_i}$$

- Find final draping position by minimizing the total energy in the cloth
  - NOT dynamic!

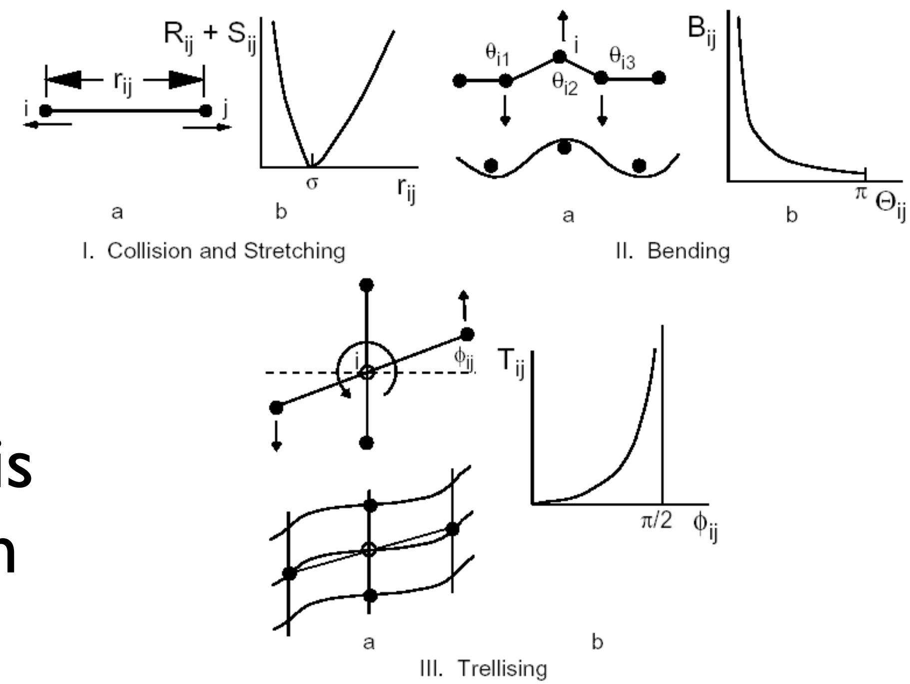


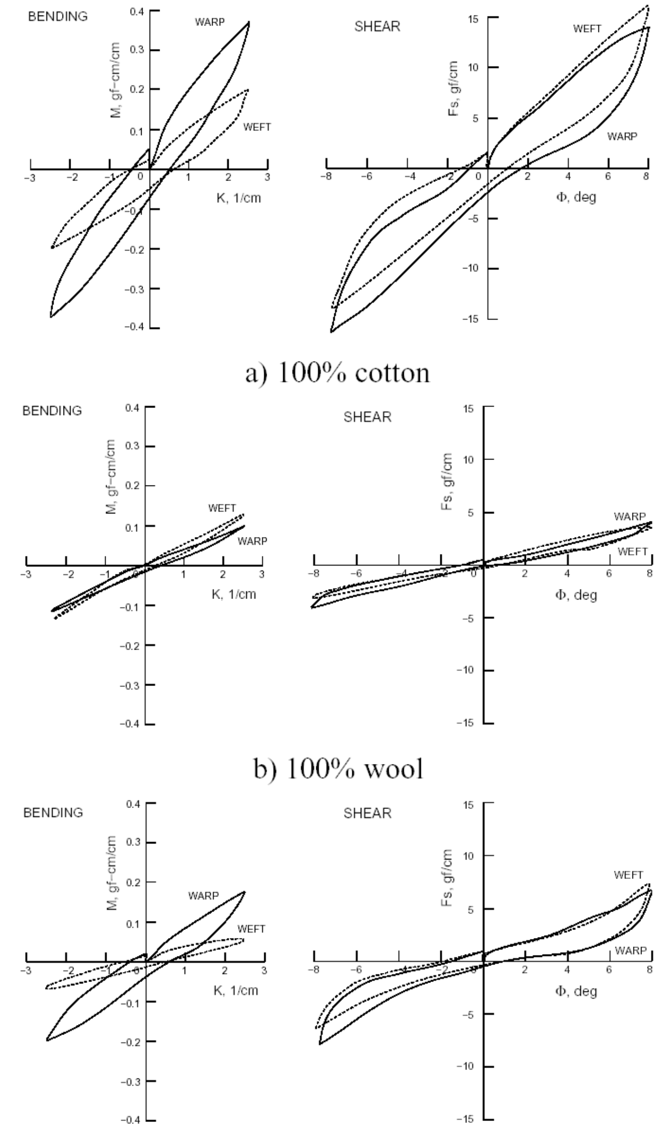
Figure 3: Cloth model energy functions

Note: You could convert this to a "normal" particle system model by differentiating energy w.r.t. position,

$$\mathbf{F} = -\nabla_{\mathbf{x}} U$$

Breen [1984]

- Tries to make the drape more realistic by measuring from reality
- Uses the Kawabata system
- Fit functions to the measured data



c) polyester/cotton

Kawabata plots for 3 different types of fabric (Breen, House, Wozny [1994])

(aside) The Kawabata system

- A system for measuring the parameters of cloth
  - Stretch
  - Shear
  - Bend
  - Friction
- Developed by Kawabata [1984], used heavily in the textile engineering industry

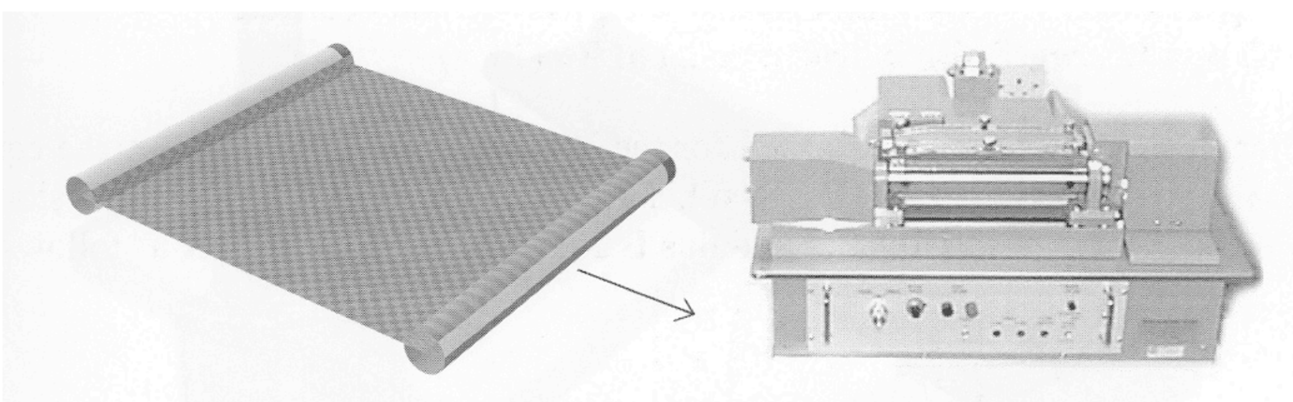
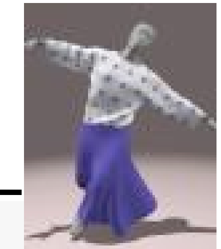


Fig. 2.4. The tensile test, and the KES-FB1 machine.

From *Virtual Clothing* [Volino, Magnenat-Thalmann]

Breen [1984] (2)



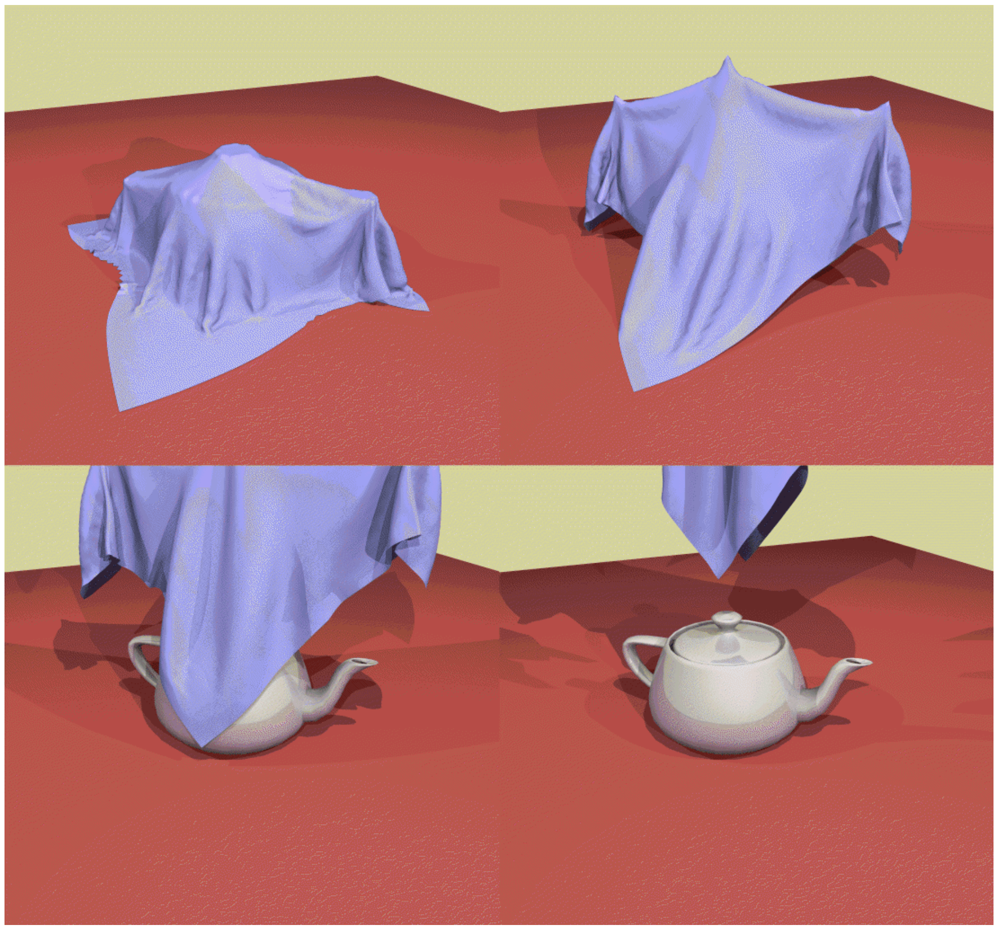


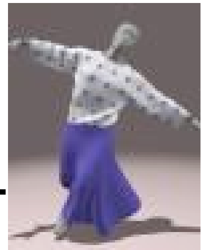
Figure 6: Actual (left) vs. simulated (right) cloth drape

- 8 -

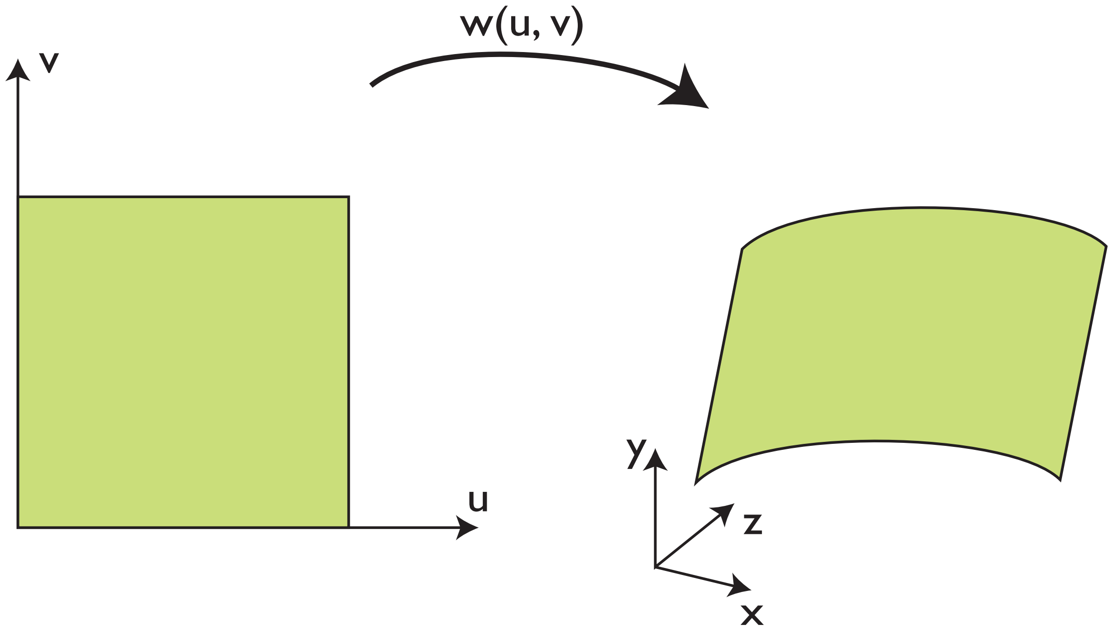
Baraff, Witkin [1998]

- A hybrid approach:
  - Energy-function-based (similar to Breen)
    - Sparse Jacobian
    - Linear forces for numerical reasons
  - Triangle-based
    - Energy functions defined over finite regions
    - But how do we determine stretch and shear on *triangles* (especially if we want to privilege warp and weft directions)?

Baraff, Witkin [1998] (2)



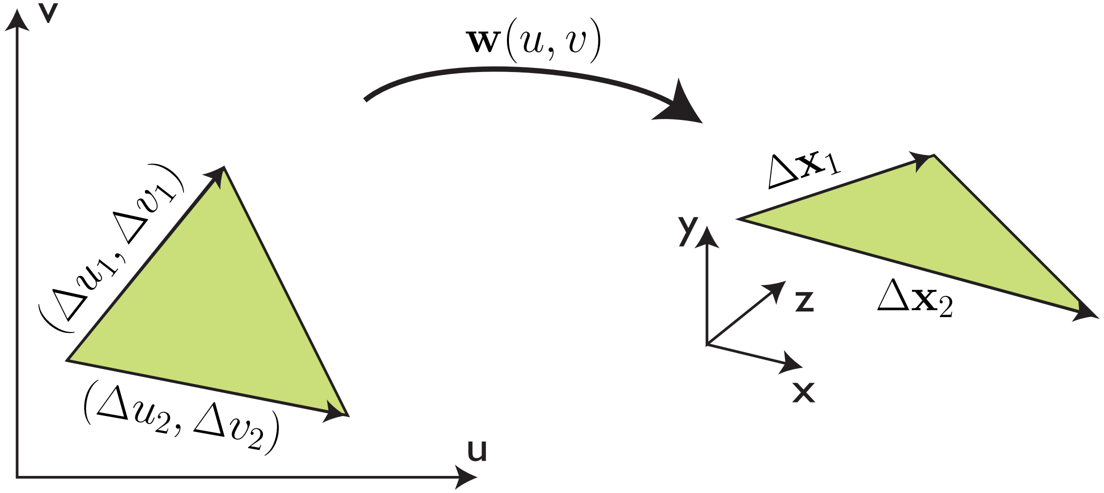
- Basic idea: treat the cloth as a 2-dimensional *manifold* embedded in $\mathbb{R}^3$



Note that this mapping only needs to be valid locally (useful for clothing)

Baraff, Witkin [1998] (3)

We are interested in the vectors $\mathbf{w}_u$ and $\mathbf{w}_v$



If we pretend that w is locally linear, we get

$$\Delta\mathbf{x}_1 = \mathbf{w}_u \Delta u_1 + \mathbf{w}_v \Delta v_1$$
$$\Delta\mathbf{x}_2 = \mathbf{w}_u \Delta u_2 + \mathbf{w}_v \Delta v_2$$

Baraff, Witkin [1998] (4)

- Energy functions are defined in terms of a (heuristic) “soft” constraint function $\mathbf{C}(\mathbf{x})$, e.g.

Stretch:

triangle area — rest length

$$\mathbf{C}(\mathbf{x}) = a \begin{pmatrix} ||\mathbf{w}_u(\mathbf{x})|| - b_u \\ ||\mathbf{w}_v(\mathbf{x})|| - b_v \end{pmatrix}$$

Shear:

$$\mathbf{C}(\mathbf{x}) = a\mathbf{w}_u(\mathbf{x})^T\mathbf{w}_v(\mathbf{x})$$

Bend:

angle between triangle faces

$$\mathbf{C}(\mathbf{x}) = \theta$$

Now, energy and force are defined as

$$E_{\mathbf{c}}(\mathbf{x}) = \frac{k}{2}\mathbf{C}(\mathbf{x})^T\mathbf{C}(\mathbf{x}) \qquad \mathbf{f}(\mathbf{x}) = -\frac{\partial E_{\mathbf{C}}}{\partial \mathbf{x}}$$

Baraff, Witkin [1998] (5)

- Damping forces turn out to be important both for realism and numerical stability
- Damping forces should
  - Act in direction of corresponding elastic force
  - Be proportional to the velocity in that direction

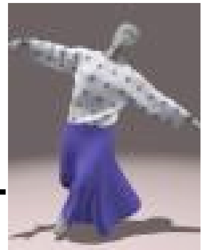
Hence, we derive (this should look familiar)

$$\mathbf{d} = -k_d \dot{\mathbf{C}}(\mathbf{x}) \frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}}$$

where

$$\dot{\mathbf{C}}(\mathbf{x}) = \frac{\partial \mathbf{C}(\mathbf{x})}{\partial t} = \overbrace{\frac{\partial \mathbf{C}(\mathbf{x})}{\partial \mathbf{x}}}^{\text{Direction of force}} \frac{\partial \mathbf{x}}{\partial t}$$

Baraff, Witkin [1998] (6)



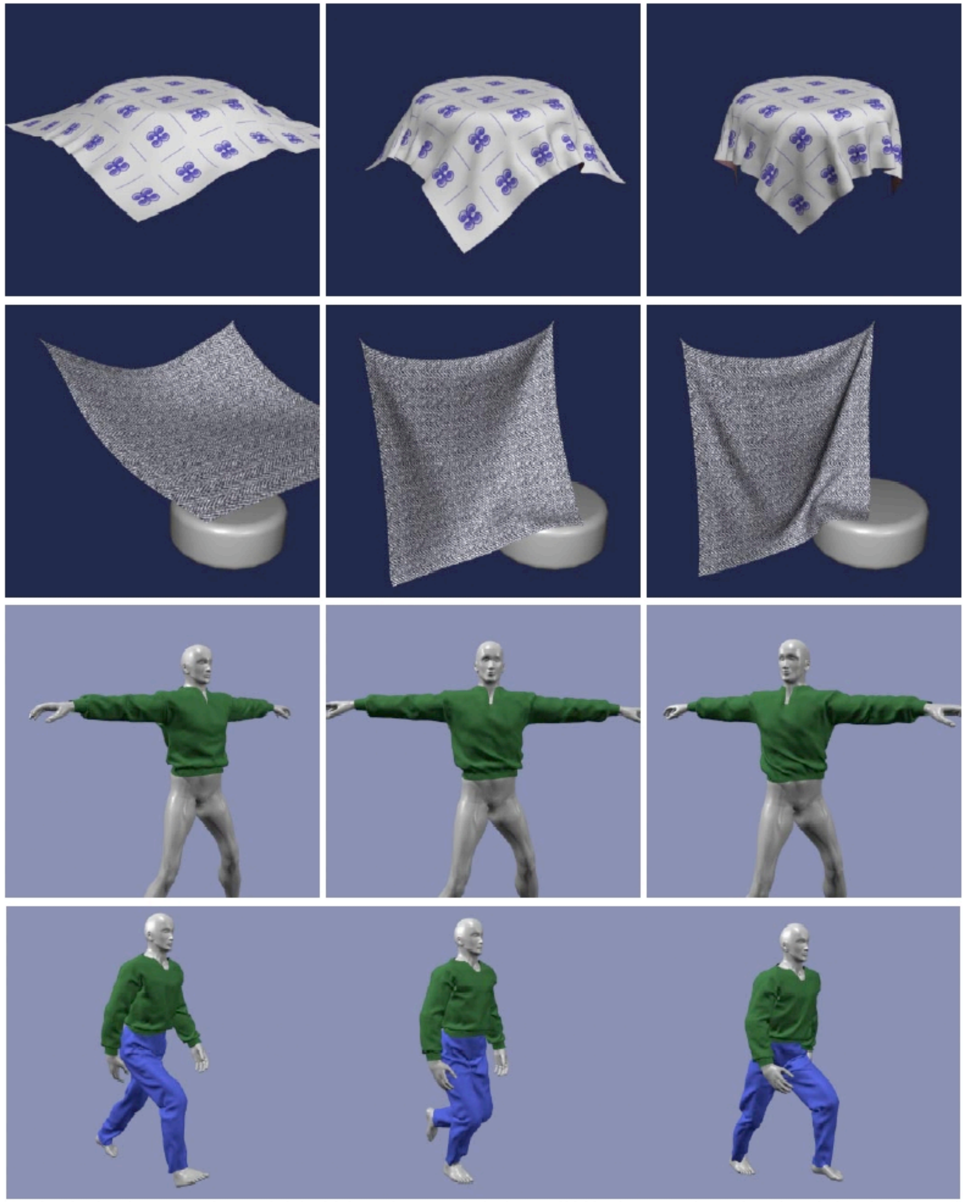


Figure 1 (top row): Cloth draping on cylinder; frames 8, 13 and 35. Figure 2 (second row): Sheet with two fixed particles; frames 10, 29 and 67. Figure 3 (third row): Shirt on twisting figure; frames 1, 24 and 46. Figure 4 (bottom row): Walking man; frames 30, 45 and 58.

Figure 5 (top row): Dancer with short skirt; frames 110, 136 and 155. Figure 6 (middle row): Dancer with long skirt; frames 185, 215 and 236. Figure 7 (bottom row): Closeups from figures 4 and 6.

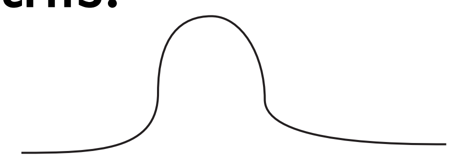
Baraff, Witkin [1998] (7)

- Use by Alias|Wavefront in Maya Cloth
- Something similar used by Pixar

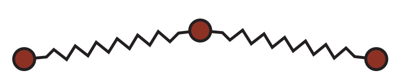
Ko, Choi [2002]

Basic problem: when we push on a piece of cloth like this,

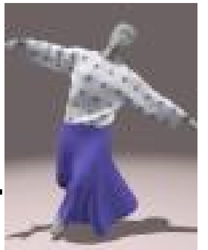
we expect to see this:



But, in our basic particle system model, we have to make the compression forces very stiff to get significant out-of-plane motion. This is expensive.



Ko, Choi [2002] (2)



Ko, Choi use *column buckling* as their basic model.

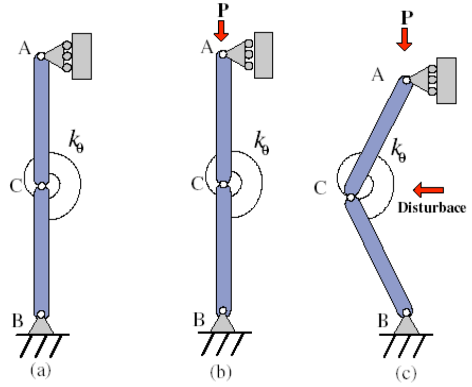
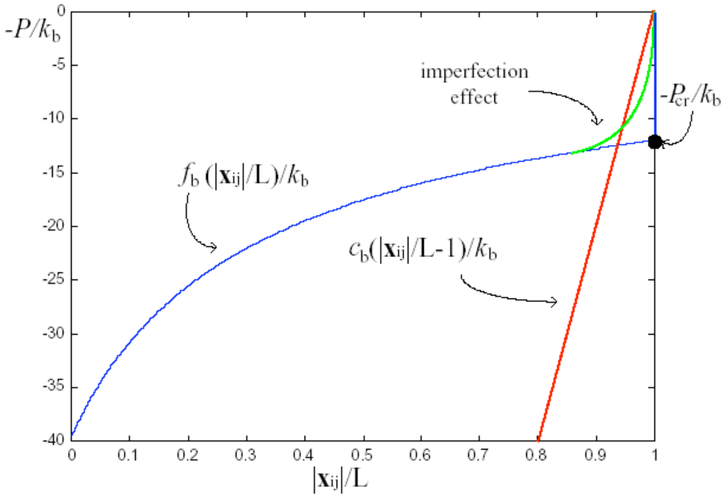
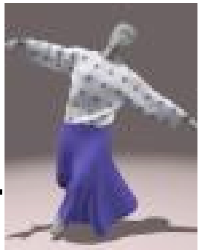


Figure 3: Column Buckling

They replace bend and compression forces with a single nonlinear model.



Ko, Choi [2002] (3)



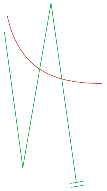
Ko, Choi [2002] (4)

Ko, Choi [2002] (5)

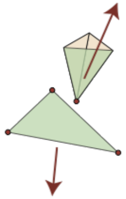
Outline

- Overview

- Models



- Integrating stiff systems



- Collision handling

Stiffness in ODEs

Recall

> "Loosely speaking, the initial value problem is referred to as being stiff if the absolute stability requirement dictates a much smaller time step than is needed to satisfy approximation requirements alone." (Ascher, Petzold [1997])

What does this mean?

Stiffness in ODEs -- example

Consider the following ODE:

$$\frac{dx}{dt} = -kx, \;\; k \gg 1$$

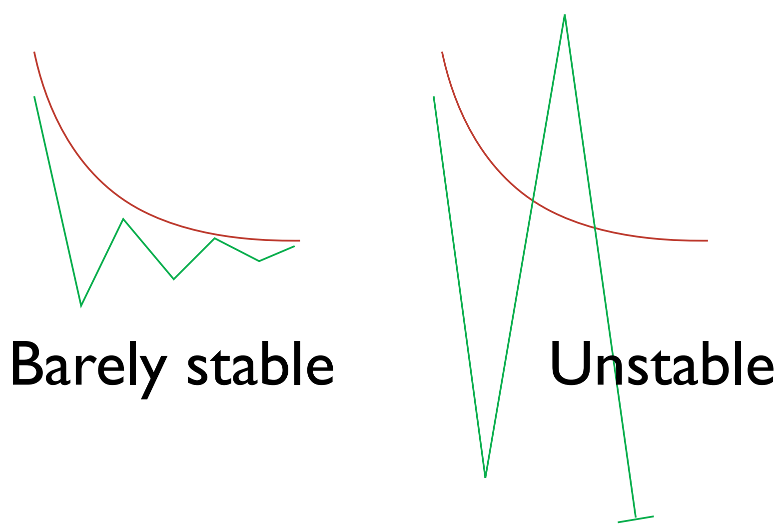
The analytical solution is

$$x(t) = Ce^{-kt}$$

If we solve it with Euler's method,

$$x_{t+h} = x_t - hkx_t = (1 - hk)x_t$$

What happens when $hk \gg 1$?



Stiffness in cloth

- In general, cloth stretches little if at all in the plane
- To counter this, we generally have large in-plane stretch forces (otherwise the cloth looks "wiggly")
- The result: stiffness!

Implicit Euler

- The solution is to use *implicit methods* (Terzopolous et al. [1987], Baraff/Witkin [1998])
- Basic idea: express the derivatives at the current timestep in terms of the system state at the next timestep; e.g., backward Euler:

$$\mathbf{y}_{t+h} = \mathbf{y}_t + h\mathbf{f}(t+h, \mathbf{y}_{t+h})$$

We can apply this to our test equation,

$$x_{t+h} = x_t + h(-kx_{t+h})$$
$$x_{t+h}(1+hk) = x_t$$
$$x_{t+h} = \frac{x_t}{1+hk}$$

And, voila! For any $hk > 0$, |x| actually decreases as a function of time.

Implicit Euler (2)

The drawback is that if we look at our equation,

$$\mathbf{y}_{t+h} = \mathbf{y}_t + h\mathbf{f}(t+h, \mathbf{y}_{t+h})$$

$\mathbf{y}_{t+h}$ appears on both sides of the equation -- hence the name "implicit."

Solution: rewrite it as

$$\mathbf{g}(\mathbf{y}_{t+h}) = \mathbf{y}_{t+h} - \mathbf{y}_t - h\mathbf{f}(t+h, \mathbf{y}_{t+h}) = \mathbf{0}$$

and use Newton's method.

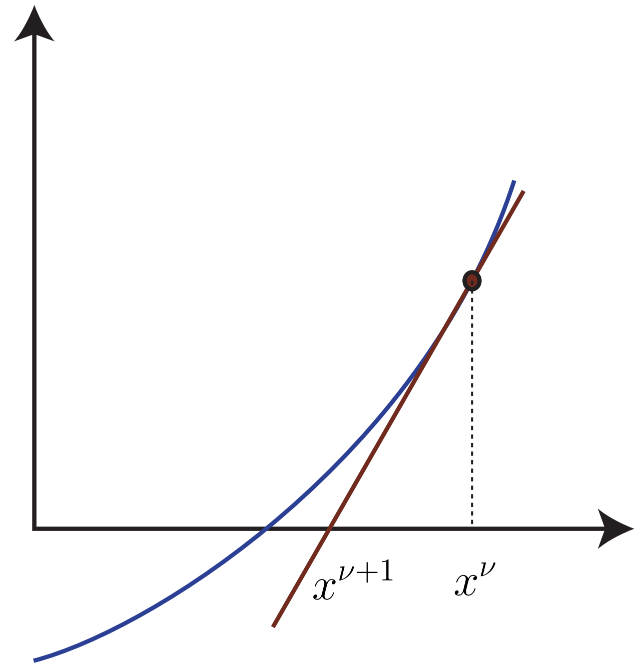
Newton's method

For a nonlinear equation

$$g(x) = 0$$

with some initial guess x^0, we can iterate: for a given iterate x^ν, we find the next by solving the linear equation

$$0 = g(x^\nu) + g'(x^\nu)(x - x^\nu)$$



Newton's method (2)

In m dimensions, this becomes

$$\mathbf{g}(\mathbf{x}) = \mathbf{0}$$

$$\mathbf{x}^{\nu+1} = \mathbf{x}^{\nu} - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\mathbf{x}^{\nu})\right)^{-1} \mathbf{g}(\mathbf{x}^{\nu}),\ \nu = 0, 1, \ldots$$

Or, rearranging to make it easier to solve,

$$\frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\mathbf{x}^{\nu+1} - \mathbf{x}^{\nu}) = -\mathbf{g}(\mathbf{x}^{\nu}),\ \nu = 0, 1, \ldots$$

We can use solve this with our favorite linear systems solver.

Implicit Euler (3)

Newton's method on the equation

$$\mathbf{g}(\mathbf{y}_{t+h}) = \mathbf{y}_{t+h} - \mathbf{y}_t - h\mathbf{f}(t+h, \mathbf{y}_{t+h}) = \mathbf{0}$$

results in the equation

$$\mathbf{y}_{t+h}^{\nu+1} = \mathbf{y}_{t+h}^{\nu} - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right)^{-1} \mathbf{g}(\mathbf{y}_{t+h}^{\nu})$$

or

$$\mathbf{y}_{t+h}^{\nu+1} = \mathbf{y}_{t+h}^{\nu} - \left(\mathbf{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right)^{-1} (\mathbf{y}_{t+h}^{\nu} - \mathbf{y}_t - h\mathbf{f}(t+h, \mathbf{y}_{t+h}^{\nu}))$$

Rewriting as usual to eliminate the matrix inverse,

$$\left(\mathbf{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right) (\mathbf{y}_{t+h}^{\nu+1} - \mathbf{y}_{t+h}^{\nu}) = -\mathbf{y}_{t+h}^{\nu} + \mathbf{y}_t + h\mathbf{f}(t+h, \mathbf{y}_{t+h}^{\nu})$$

With the initial guess $\mathbf{y}_{t+h}^{0} = \mathbf{y}_t$, the first iteration is

$$\left(\mathbf{I} - h\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right) (\mathbf{y}_{t+h} - \mathbf{y}_t) = h\mathbf{f}(t+h, \mathbf{y}_t)$$

Implicit Euler in Baraff/Witkin

Recall that our differential equation for cloth is (in state-space formulation),

$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}\\ \mathbf{v}\end{bmatrix} = \begin{bmatrix}\mathbf{v}\\ \mathrm{M}^{-1}\mathbf{f}(\mathbf{x},\mathbf{v})\end{bmatrix}$$

The implicit Euler method is

$$\begin{bmatrix}\mathbf{x}_{t+h}\\ \mathbf{v}_{t+h}\end{bmatrix} = \begin{bmatrix}\mathbf{x}_t\\ \mathbf{v}_t\end{bmatrix} + h\begin{bmatrix}\mathbf{v}_{t+h}\\ \mathrm{M}^{-1}\mathbf{f}(\mathbf{x}_{t+h},\mathbf{v}_{t+h})\end{bmatrix}$$

Take the first Newton iteration only (for speed):

$$\left(\mathbf{I} - h\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathrm{M}^{-1}\frac{\partial\mathbf{f}}{\partial\mathbf{x}} & \mathrm{M}^{-1}\frac{\partial\mathbf{f}}{\partial\mathbf{v}}\end{bmatrix}\right)\begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{v}\end{bmatrix} = h\begin{bmatrix}\mathbf{v}\\ \mathrm{M}^{-1}\mathbf{f}(\mathbf{x}_t,\mathbf{v}_t)\end{bmatrix}$$

Baraff and Witkin reduce the dimensionality by back-substituting Δx into the equation for Δv

Implicit Euler in B/W (2)

The final equation they solve is

$$\left(\mathbf{I} - h\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\Delta\mathbf{v} = h\mathbf{M}^{-1}\left(\mathbf{f}_0 + h\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{v}_0\right)$$

$$\left(\mathbf{M} - h\frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\Delta\mathbf{v} = h\left(\mathbf{f}_0 + h\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{v}_0\right)$$

Assuming a reasonable force model, this is (almost) symmetric and positive definite, so it can be solved using conjugate gradient.

Conjugate Gradient in B/W

In many cases, we would actually like certain masses to be ∞, e.g., for constraints.

In this case, the matrix M^{-1} is rank deficient, multiplying by M is meaningless

Solution: use the "unconstrained" M matrix in PCG -- but after every iteration project back onto the constraint manifold.

For details, consult Baraff and Witkin [1998]. Also:

Ascher, U. and Boxerman, E. "On the modified conjugate gradient method in cloth simulation." http://www.cs.ubc.ca/spider/ascher/papers/ab.pdf

Higher-order implicit methods

Implicit Euler has only first-order accuracy

More recently, people have been using 2nd-order backward differences (Ko/Choi [2002], Bridson et al [2002]).

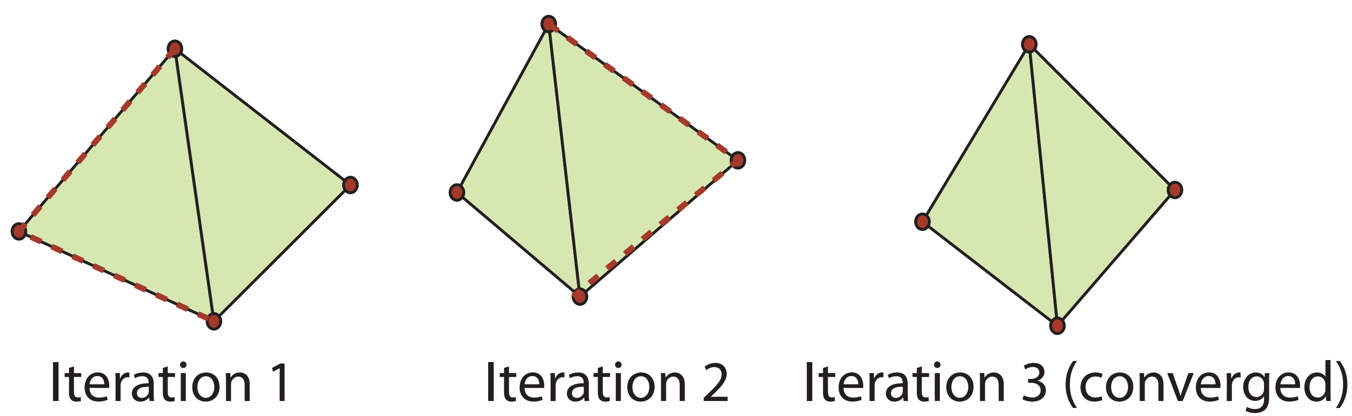
- Multistep
- 2nd order accuracy

Avoiding stiffness

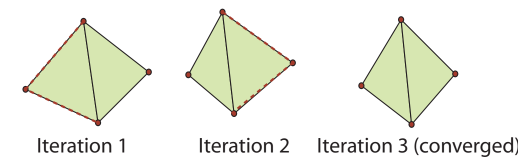
An alternative approach is to avoid stiffness altogether by applying only non-stiff spring forces and then "fixing" the solution at the end of the timestep.

(Provot [1995], Desbrun et al [1999], Bridson et al [2002])

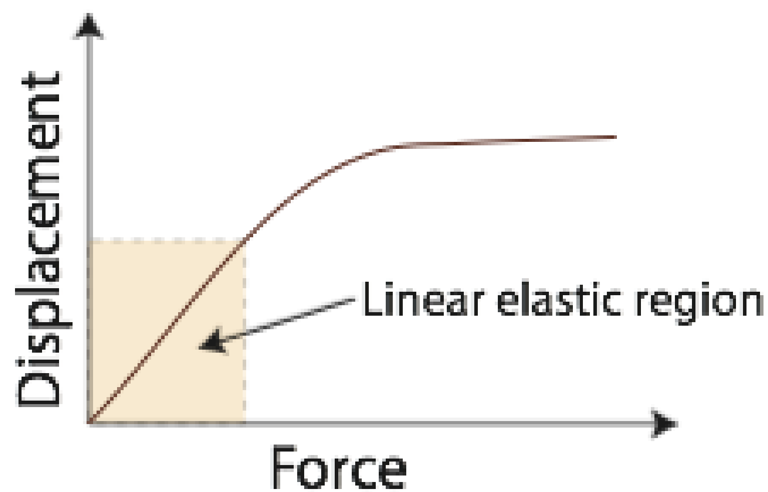
We can do this with impulses and Jacobi iteration.



Avoiding stiffness (2)



- Popular for interactive applications
- Justification
  - Biphasic spring model



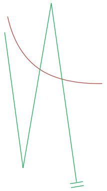
From Desbrun, Meyer, Barr [2000]

  - Plausible dynamics

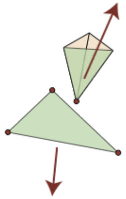
Outline

- Overview

- Models



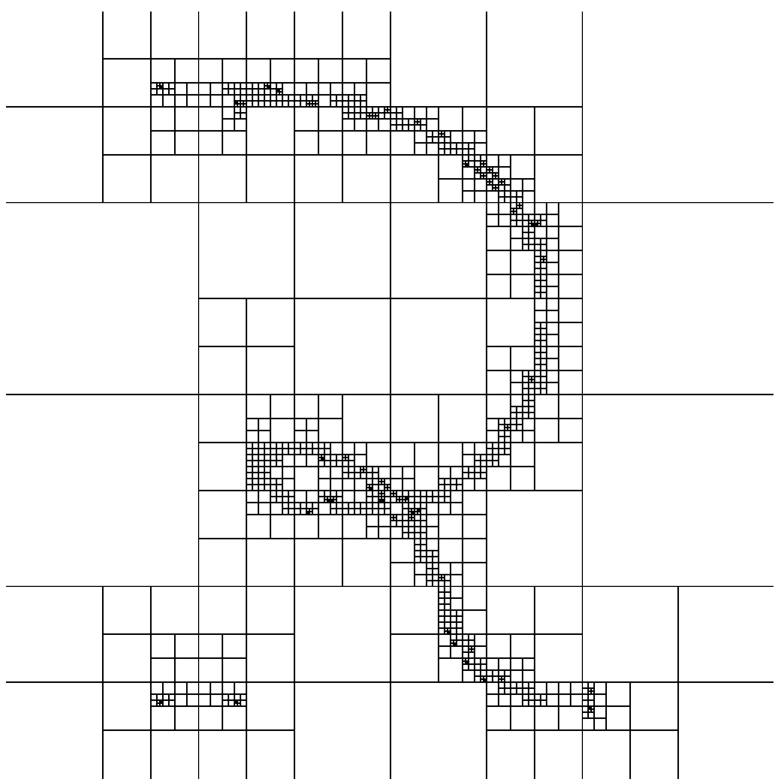
- Integrating stiff systems



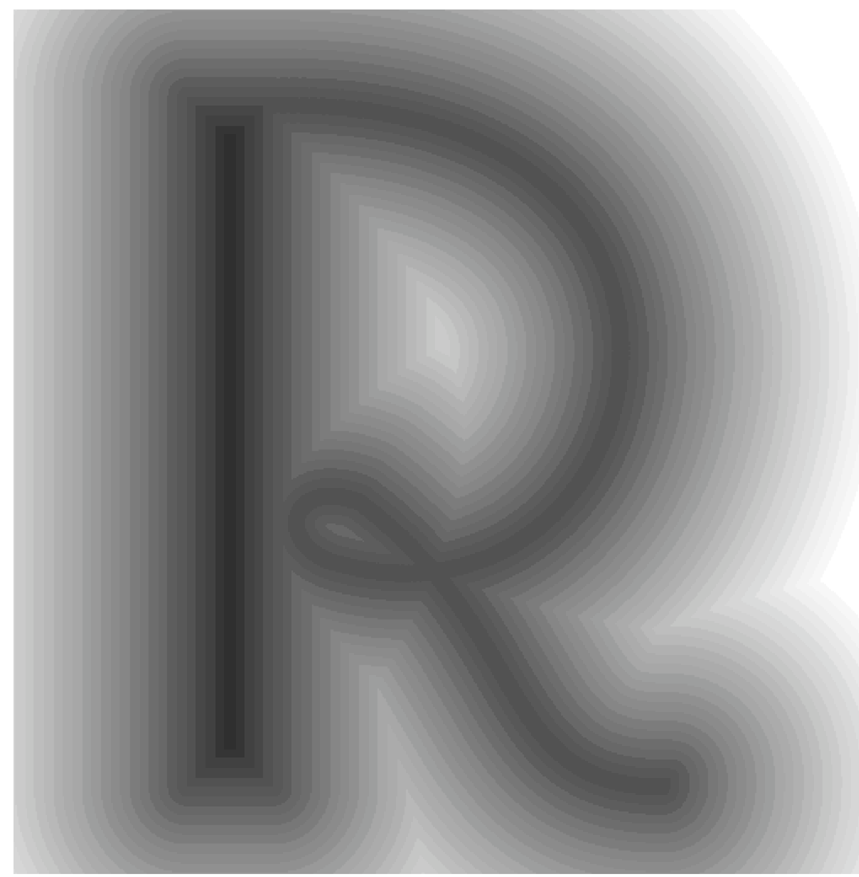
- Collision handling

Collisions with rigid objects

- Current best practice: use implicit surfaces

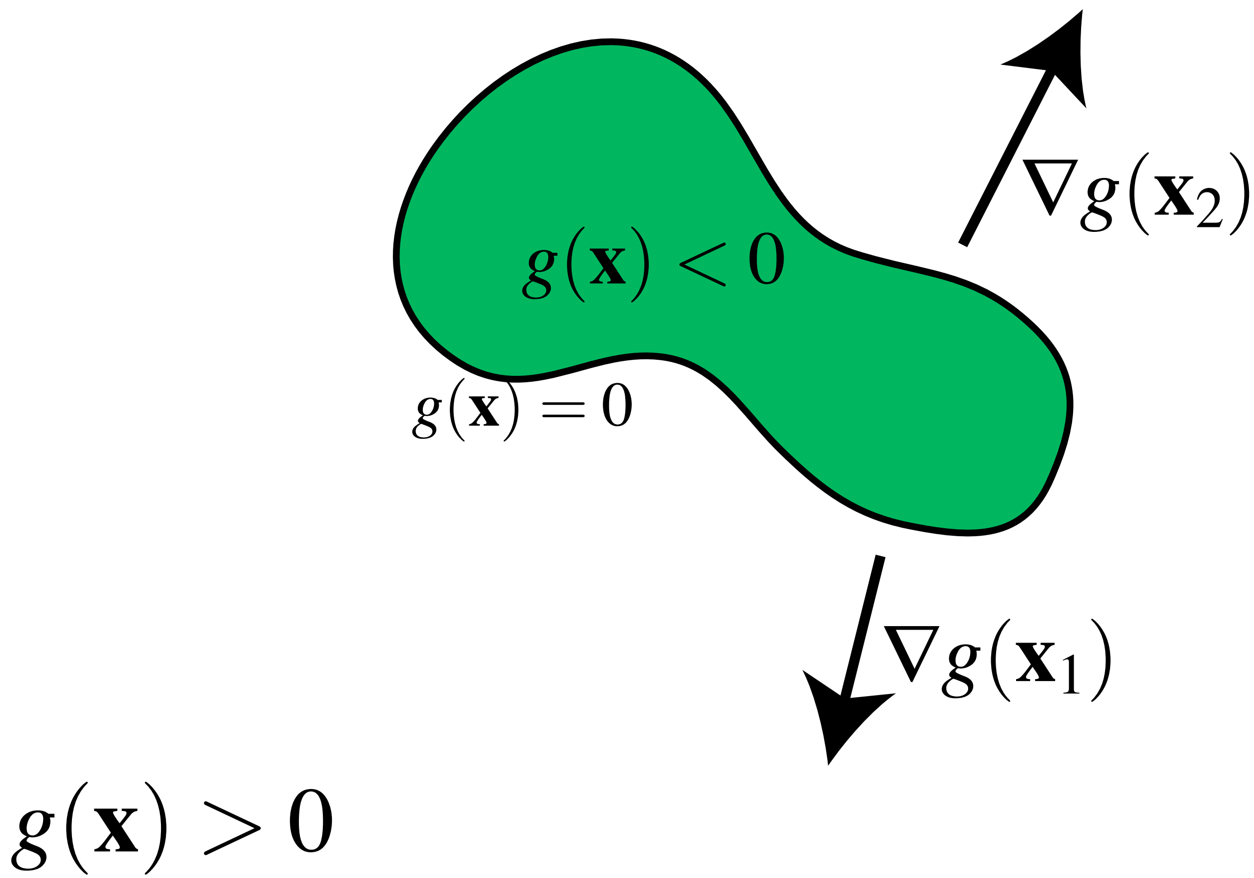


[Frisken, Perry, Rockwood, Jones 2000]



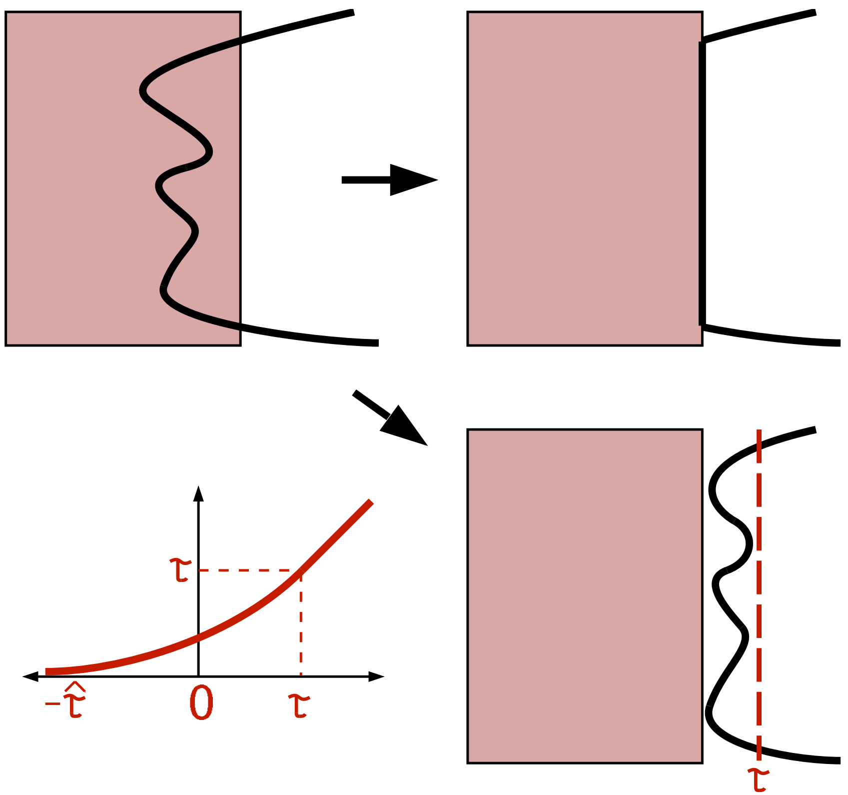
Collisions with rigid objects (2)

$$\mathbf{f}(\mathbf{x}) \propto \nabla g(\mathbf{x})$$



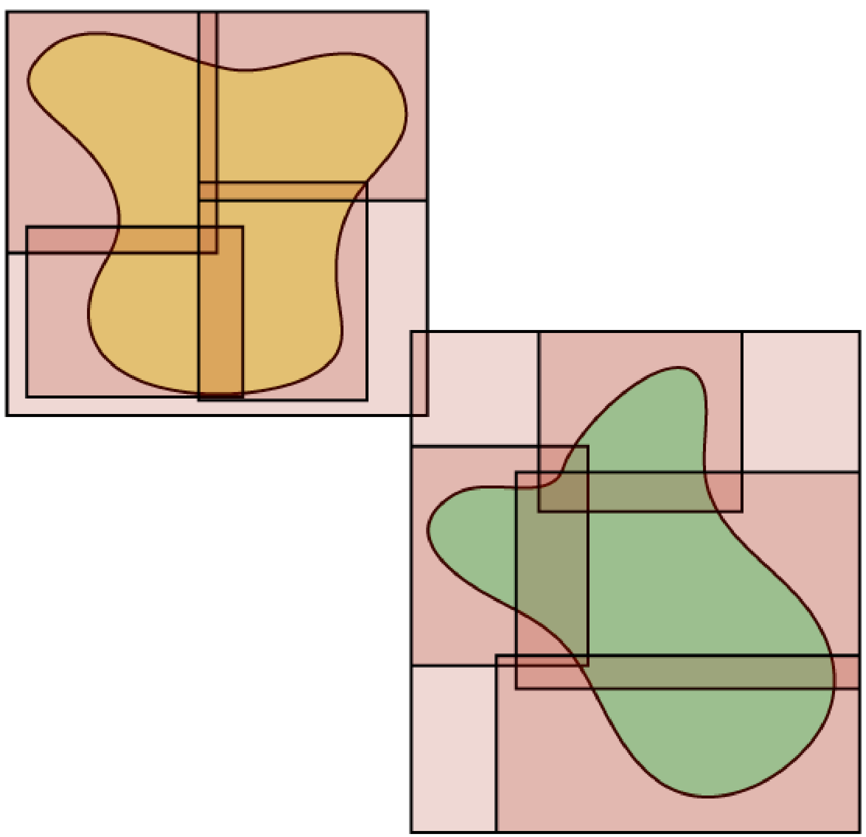
Collisions with rigid objects (3)

- See also [Bridson, Marino, Fedkiw 2003]

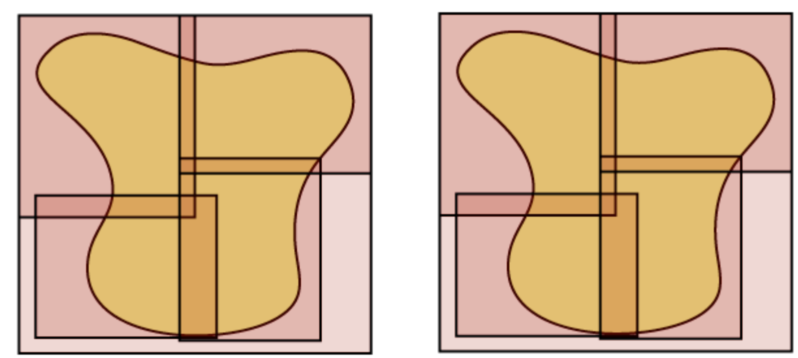


Self-collisions

- First problem: detection

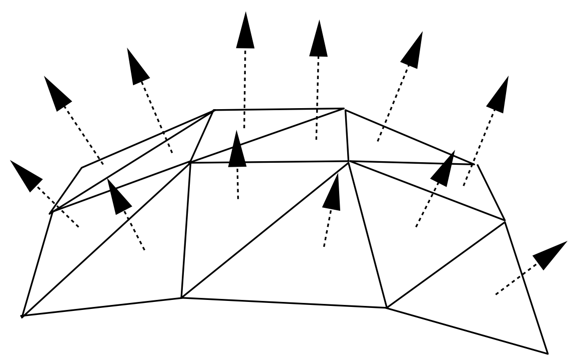


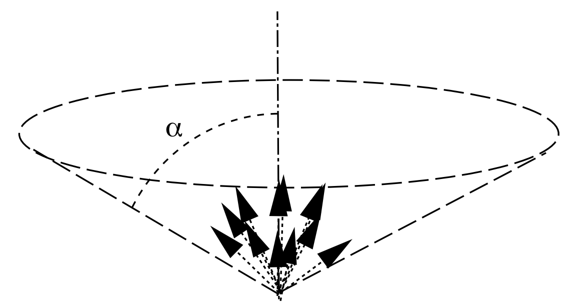
vs.

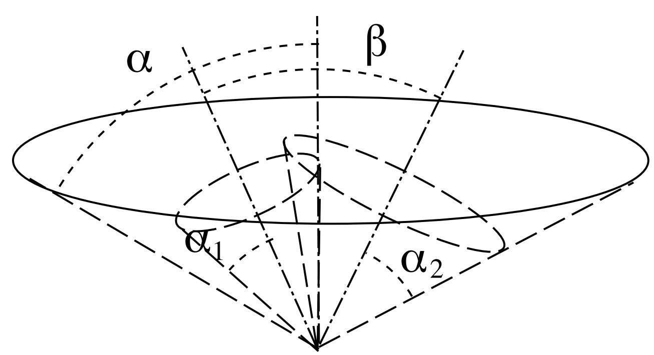


Self-collisions: detection

- Solution: store curvature information in the bounding volume hierarchy ([Volino and Magnenat-Thalmann 1994] [Provot 1997])

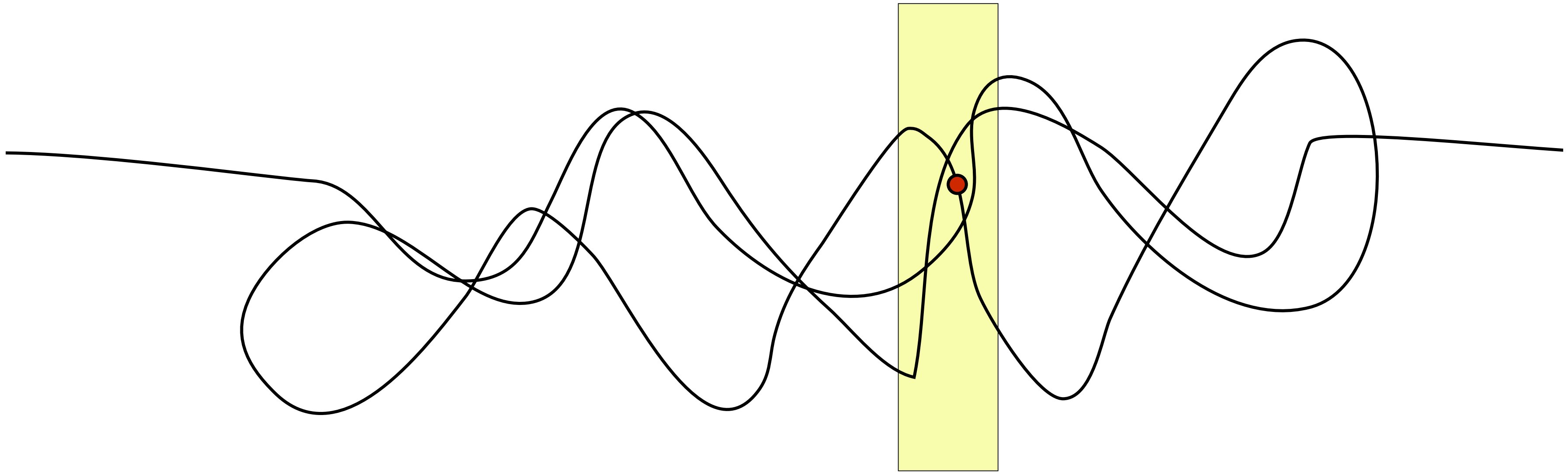






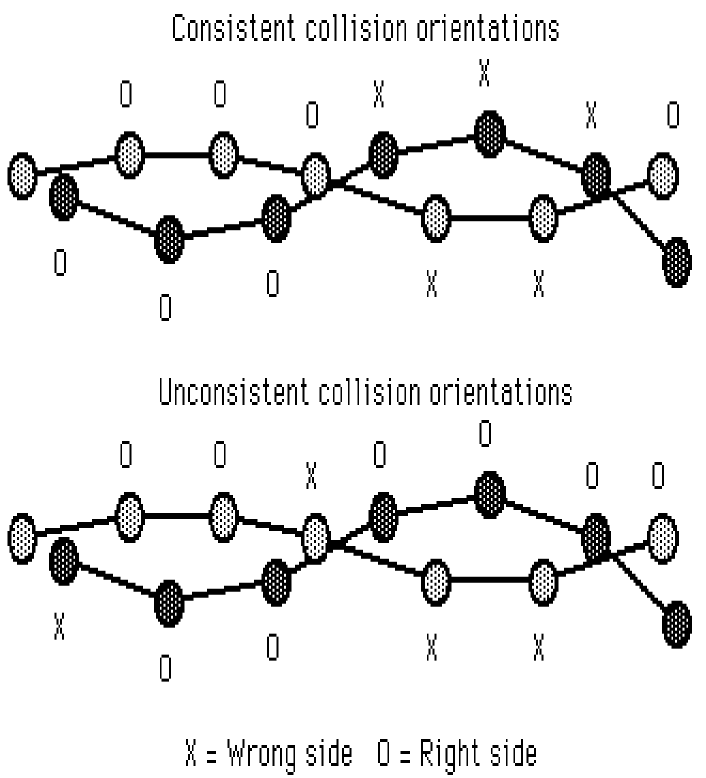
The “infinite thinness” problem

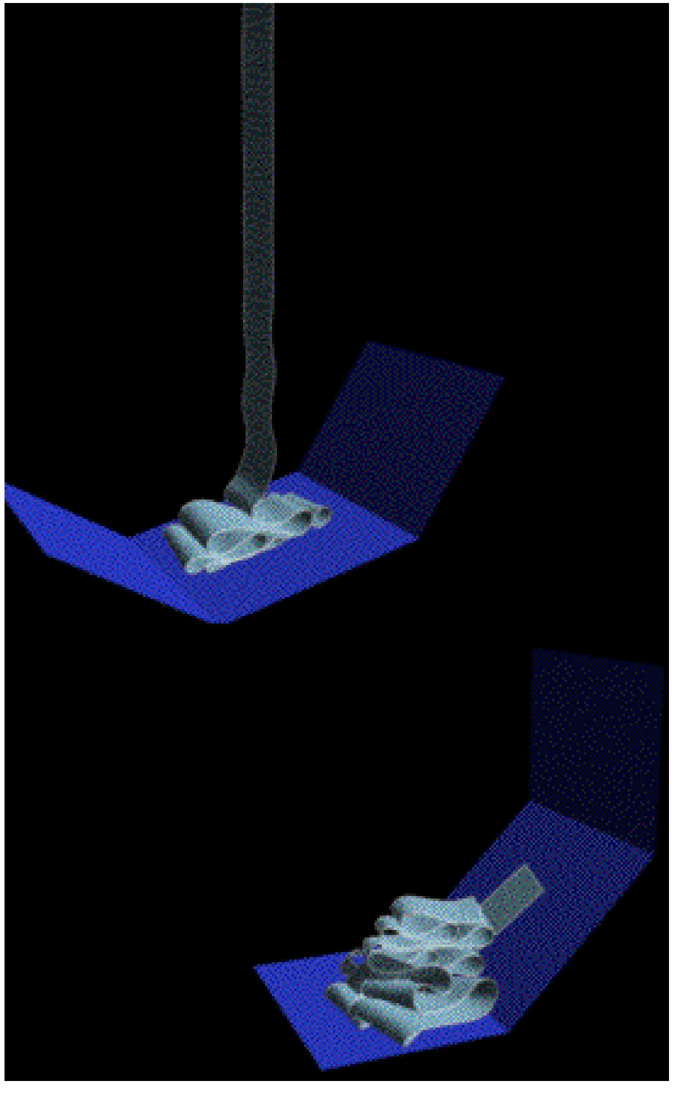
Which way is “out”?



The "infinite thinness" problem

- Solution 1: algorithmically infer orientation

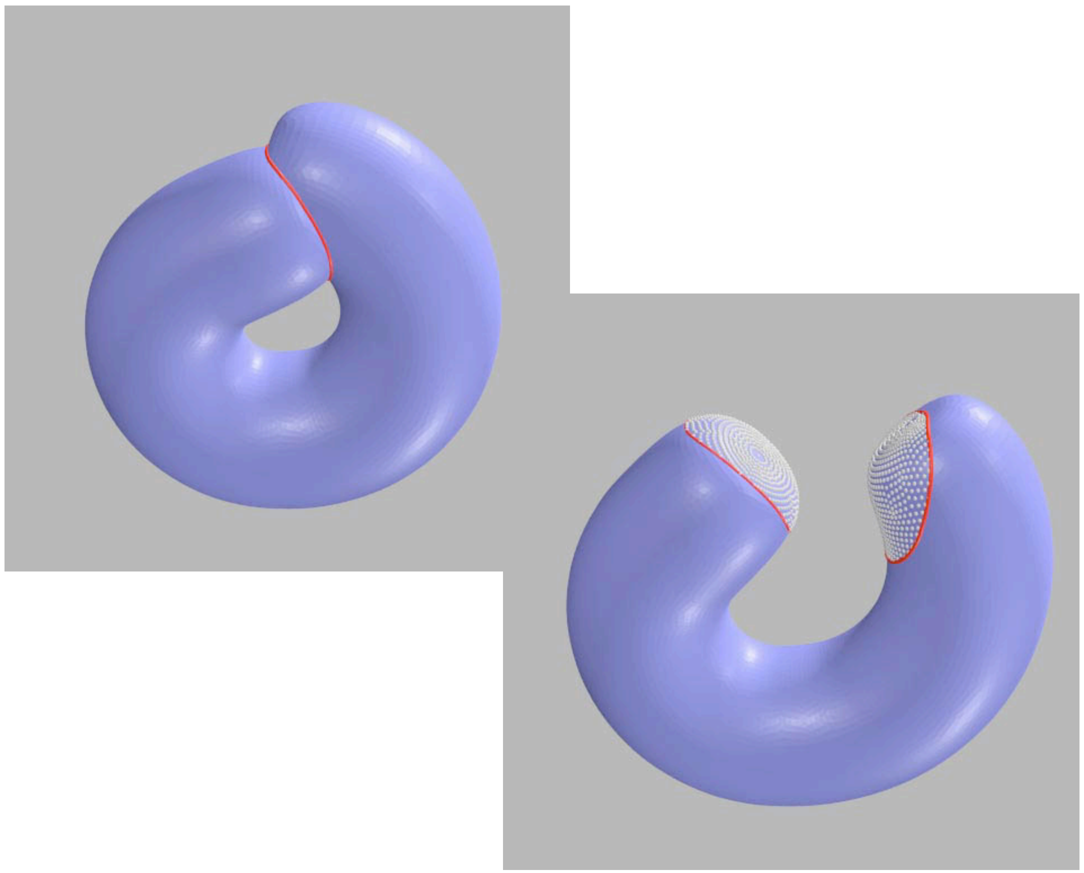




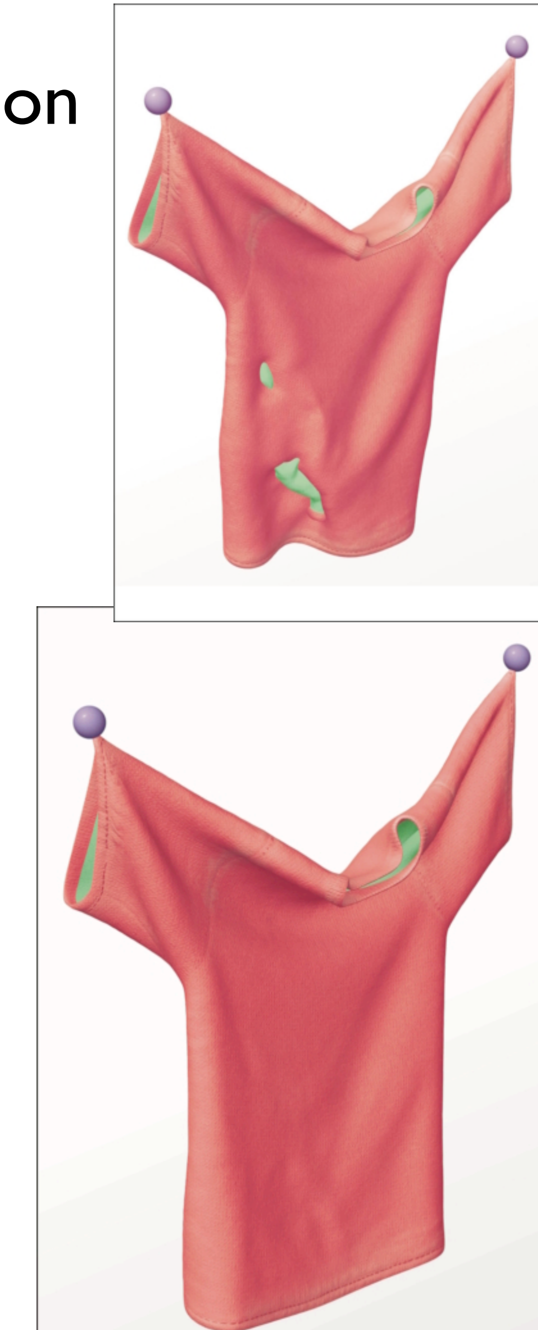
Locally [Volino, Courchesne, Magnenat-Thalmann 1995]

The “infinite thinness” problem

- Solution 1: algorithmically infer orientation



Globally [Baraff, Witkin 2003]

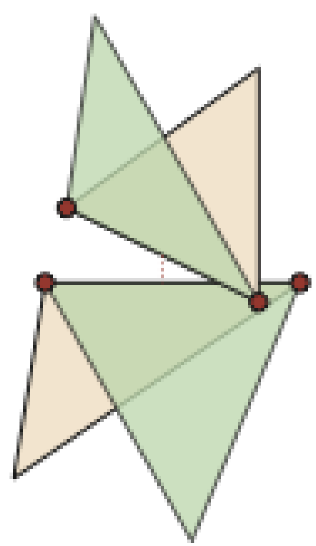


The “infinite thinness” problem

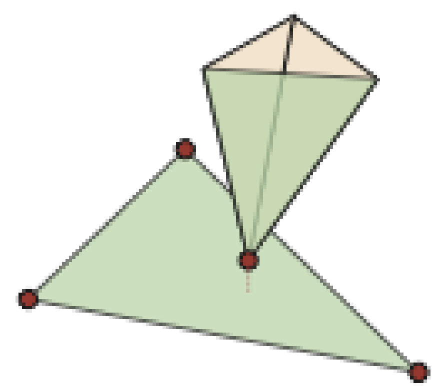
- Solution 2: assume everything starts consistent, never allow anything to pass through
  - but how?

Collision detection

- Generally need to do triangle-triangle collision checks:



Edge-edge collision



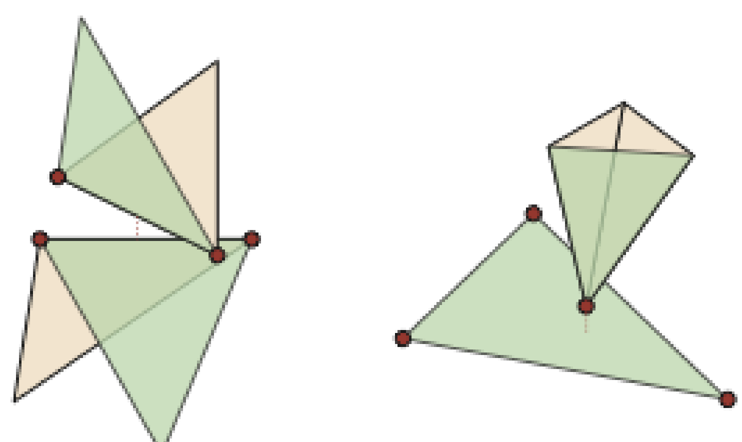
Point-face collision

Robust collision detection

If triangles are moving too fast, they may pass through each other in a single timestep.

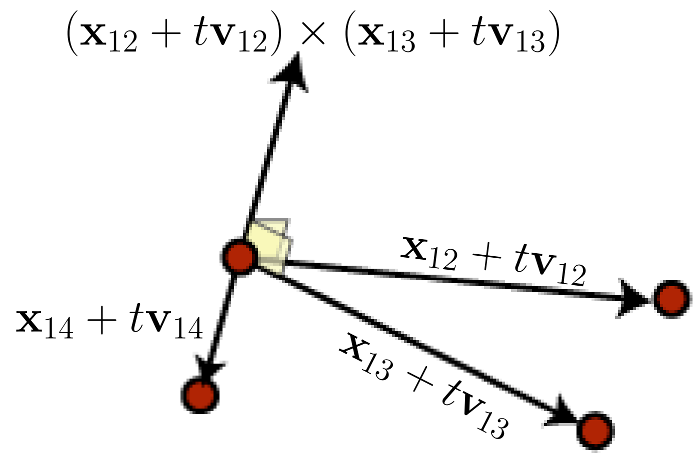
We can prevent this by checking for *any* collisions during the timestep (Provot [1997])

Note first that both point-face and edge-edge collisions occur when the appropriate 4 points are *coplanar*



Robust collision detection (2)

Detecting time of coplanarity - assume linear velocity throughout timestep:



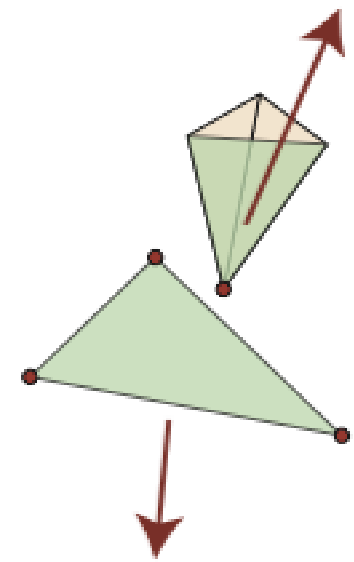
So the problem reduces to finding roots of the cubic equation

$$\left((\mathbf{x}_{12} + t\mathbf{v}_{12}) \times (\mathbf{x}_{13} + t\mathbf{v}_{13})\right) \cdot (\mathbf{x}_{14} + t\mathbf{v}_{14})$$

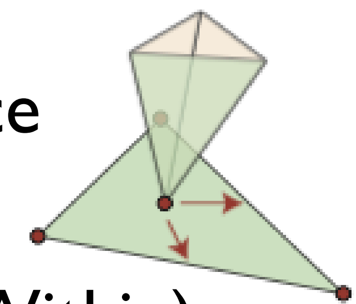
Once we have these roots, we can plug back in and test for triangle adjacency.

Collision response

- 4 basic options:
  - Constraint-based
  - Penalty forces
  - Impulse-based
  - Rigid body dynamics (will explain)



Constraint-based response

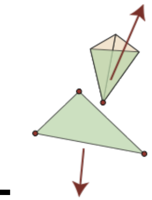


- Assume totally inelastic collision
- Constrain particle to lie on triangle surface
- Benefits:
  - Fast, may not add stiffness (e.g., Baraff/Witkin)
  - No extra damping needed
- Drawbacks
  - Only supports point-face collisions
  - Constraint attachment, release add discontinuities (constants hard to get right)
  - Doesn't handle self-collisions (generally)
- Conclusion: a good place to start, but not robust enough for heavy-duty work

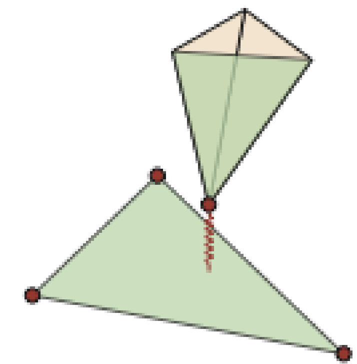
Constraint-based response (4)

- Must keep track of constraint forces in the simulator -- that is, the force the simulator is applying to maintain the constraint
- If constraint force opposes surface normal, need to release particle

Penalty forces



- Apply a spring force that keeps particles away from each other
- Benefits:
  - Easy to fit into an existing simulator
  - Works with all kinds of collisions (use barycentric coordinates to distribute responses among vertices)



- Drawbacks:
  - Hard to tune: if force is too weak, it will sometimes fail; if force is too strong, it will cause the particles to “float” and “wiggle”

Penalty forces (2)

- In general, penalty forces are not inelastic (springs store energy)
- Can be made less elastic by limiting force when particles are moving away
- Some kind of additional damping may be needed to control deformation rate along surface

Impulses

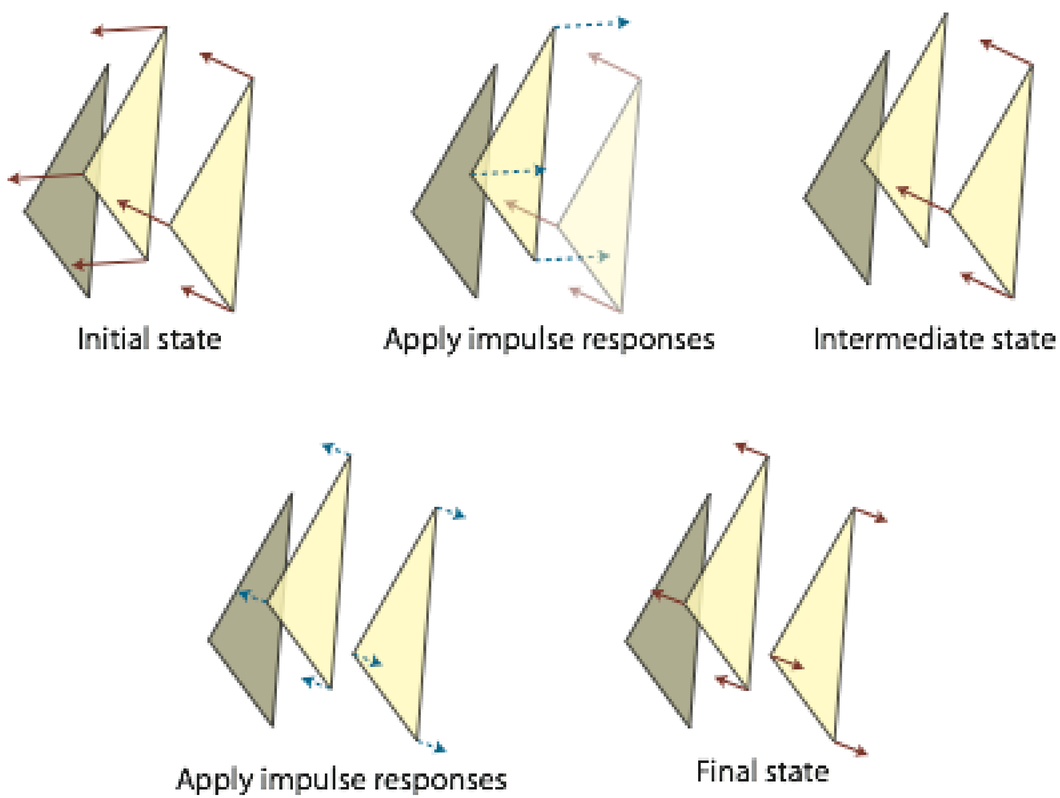
- "Instantaneous" change in momentum

$$\mathbf{J} = \int_{t_i}^{t_f} \mathbf{F}\, dt = \mathbf{p}_f - \mathbf{p}_i$$

- Generally applied outside the simulator timestep (similar to strain limiting)
- Benefits
  - Correctly stops all collisions (no sloppy spring forces)
- Drawbacks
  - Can have poor numerical performance
  - Handles persistent contact poorly

Impulses (2)

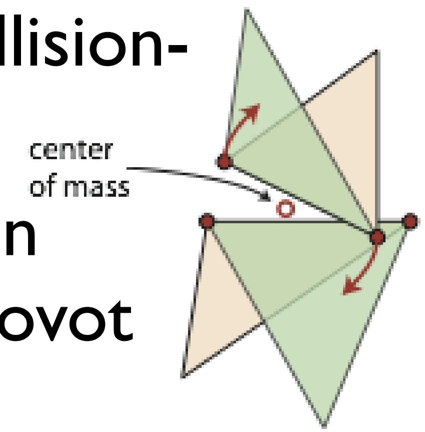
Iteration is generally necessary to remove all collisions.



Convergence may be slow in some cases.

Rigid collision impact zones

- Basic idea: if a group of particles *start* timestep collision-free, and move as a rigid body throughout the timestep, then they will *end* timestep collision-free.
- We can group particles involved in a collision together and move them as a rigid body (Provot [1997] -- error?, Bridson [2002])



$$x_{CM} = \frac{\sum_i m_i \mathbf{x}_i}{m_i} \qquad v_{CM} = \frac{\sum_i m_i \mathbf{v}_i}{m_i}$$ Center of mass frame

$$\mathbf{L} = \sum_i m_i(\mathbf{x}_i - \mathbf{x}_{CM}) \times (\mathbf{v}_i - \mathbf{v}_{CM})$$ Momentum

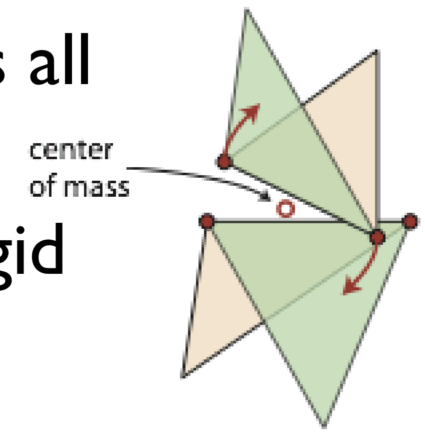
$$\mathbf{I} = \sum_i m \left(|\mathbf{x}_i - \mathbf{x}_{CM}|^2 \delta - (\mathbf{x}_i - \mathbf{x}_{CM}) \otimes (\mathbf{x}_i - \mathbf{x}_{CM}) \right)$$ Inertia tensor

$$\omega = \mathbf{I}^{-1}\mathbf{L}$$ Angular velocity

$$\mathbf{v}_i = \mathbf{v}_{CM} + \omega \times (\mathbf{x}_i - \mathbf{x}_{CM})$$ Final velocity

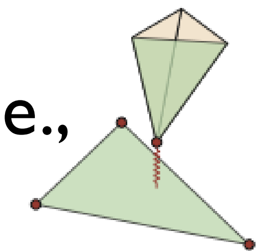
Rigid collision impact zones (2)

- Note that this is totally failsafe
- We will need to iterate, and merge impact zones as we do (e.g. until the impact zone includes all colliding particles)
- This is best used as a last resort, because rigid body cloth can be unappealing.

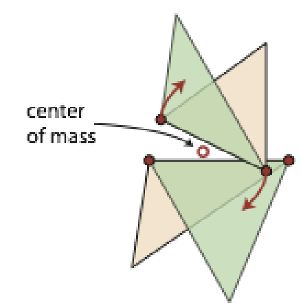


Combining methods

- So we have:
  - penalty forces - not robust, not intrusive (i.e., integrates with solver)



  - impulses - robust (esp. with iteration), intrusive - but may not converge
  - rigid impact zones - completely robust, guaranteed convergence, but very intrusive



Solution? Use all three! (Bridson et al [2002])

Combining methods (2)

Basic methodology (Bridson et al [2002]):

1. Apply penalty forces (implicitly)
2. While there are collisions left
   1. Check robustly for collisions
   2. Apply impulses
3. After several iterations of this, start grouping particles into rigid impact zones
4.

Objective: guaranteed convergence with minimal interference with cloth internal dynamics

Bridson et al. [2002]

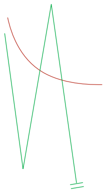
Bridson et al. [2002]

Bridson 2003 (?)

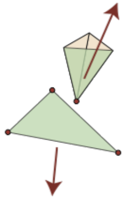
Summary

- Overview

- Models



- Integrating stiff systems



- Collision handling