COL781: Computer Graphics

33. Constraints

Müller et al. 2020

Last class

Backward Euler gives us a system of equations in the unknown next state ($\mathbf{q}_{n+1}$, $\mathbf{v}_{n+1}$):

$$\mathbf{q}^{n+1} = \mathbf{q}^{n} + \mathbf{v}^{n+1}\,\Delta t$$
$$\mathbf{v}^{n+1} = \mathbf{v}^{n} + \mathbf{M}^{-1}\,\mathbf{f}(\mathbf{q}^{n+1}, \mathbf{v}^{n+1})\,\Delta t$$

Suppose you try to implement it with Newton's method starting at $\tilde{\mathbf{q}} = \mathbf{q}_n$, $\tilde{\mathbf{v}} = \mathbf{v}_n$, but you drop the force Jacobians:

$$(\tilde{\mathbf{q}}+\Delta\mathbf{q}) = \mathbf{q}_n + (\tilde{\mathbf{v}}+\Delta\mathbf{v})\,\Delta t$$

$$(\tilde{\mathbf{v}}+\Delta\mathbf{v}) \approx \mathbf{v}_n + \mathbf{M}^{-1}\left(\mathbf{f}(\tilde{\mathbf{q}}, \tilde{\mathbf{v}}) + \cancel{\frac{\partial\mathbf{f}}{\partial\mathbf{q}}(\tilde{\mathbf{q}}, \tilde{\mathbf{v}})\,\Delta\mathbf{q}} + \cancel{\frac{\partial\mathbf{f}}{\partial\mathbf{v}}(\tilde{\mathbf{q}}, \tilde{\mathbf{v}})\,\Delta\mathbf{v}}\right)\Delta t$$

What kind of time integration scheme do you get? Does it reduce to a known one?

$$(\mathbf{q}_n+\Delta\mathbf{q}) = \mathbf{q}_n + (\mathbf{v}_n+\Delta\mathbf{v})\,\Delta t$$
$$(\mathbf{v}_n+\Delta\mathbf{v}) = \mathbf{v}_n + \mathbf{M}^{-1}\,(\mathbf{f}(\mathbf{q}_n, \mathbf{v}_n) + \mathbf{0}\,\Delta\mathbf{q} + \mathbf{0}\,\Delta\mathbf{v})\,\Delta t$$

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \mathbf{v}_{n+1}\,\Delta t$$
$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{M}^{-1}\,\mathbf{f}(\mathbf{q}_n, \mathbf{v}_n)\,\Delta t$$

This is basically semi-implicit Euler again! Except the acceleration term is also explicit in velocity (we're using $\mathbf{f}(\mathbf{q}_n, \mathbf{v}_n)$ instead of $\mathbf{f}(\mathbf{q}_n, \mathbf{v}_{n+1})$)

Moral: Approximating backward Euler can still give you good behaviour.

Corollary: You can be explicit in some terms and implicit in others, for example

- use $\mathbf{f}(\mathbf{q}_{n+1}, \mathbf{v}_{n+1})$ only for strong forces that cause instability
- use $\mathbf{f}(\mathbf{q}_n, \mathbf{v}_n)$ for weak ones (especially if their Jacobian is hard to compute)

Constraints

Example: How would you model a pendulum?

Make it a spring with rest length ℓ_0, spring constant k_s, then take k_s very large?

Rule of thumb: explicit methods are only stable when
$\Delta t = O(T_{\text{fast}})$, where T_{fast} = fastest timescale of dynamics in the system

- Period of horizontal swing: $T_{\text{slow}} \approx O\left(\sqrt{l_0/g}\right)$
- Period of vertical vibration of spring: $T_{\text{fast}} \approx O\left(\sqrt{m/k_s}\right)$

When k_s is very very large, T_{fast} and stable Δt become very very small!

We only care about dynamics on the scale of T_{slow},
but we're forced to take time steps on the scale of $T_{\text{fast}} \ll T_{\text{slow}}$.

In such cases, we say the problem is **stiff**. This happens a lot in graphics…

https://www.youtube.com/watch?v=2R9u-tjhRYA

Constraints

Another general problem-solving strategy:
If a parameter being very large is causing problems, make it infinity instead.

What happens to the spring when $k_s \to \infty$?

$$\mathbf{f}_{ij} = -k_s \left(\|\mathbf{x}_{ij}\| - \ell_0\right) \hat{\mathbf{x}}_{ij}$$

Puzzle:

- Physically, does the behaviour of this system still make sense? (At least if started from a valid initial state, $\|\mathbf{x}_{ij}{}^0\| = \ell_0$)

- What can you say about the direction and magnitude of the spring force?

Original equations of motion:

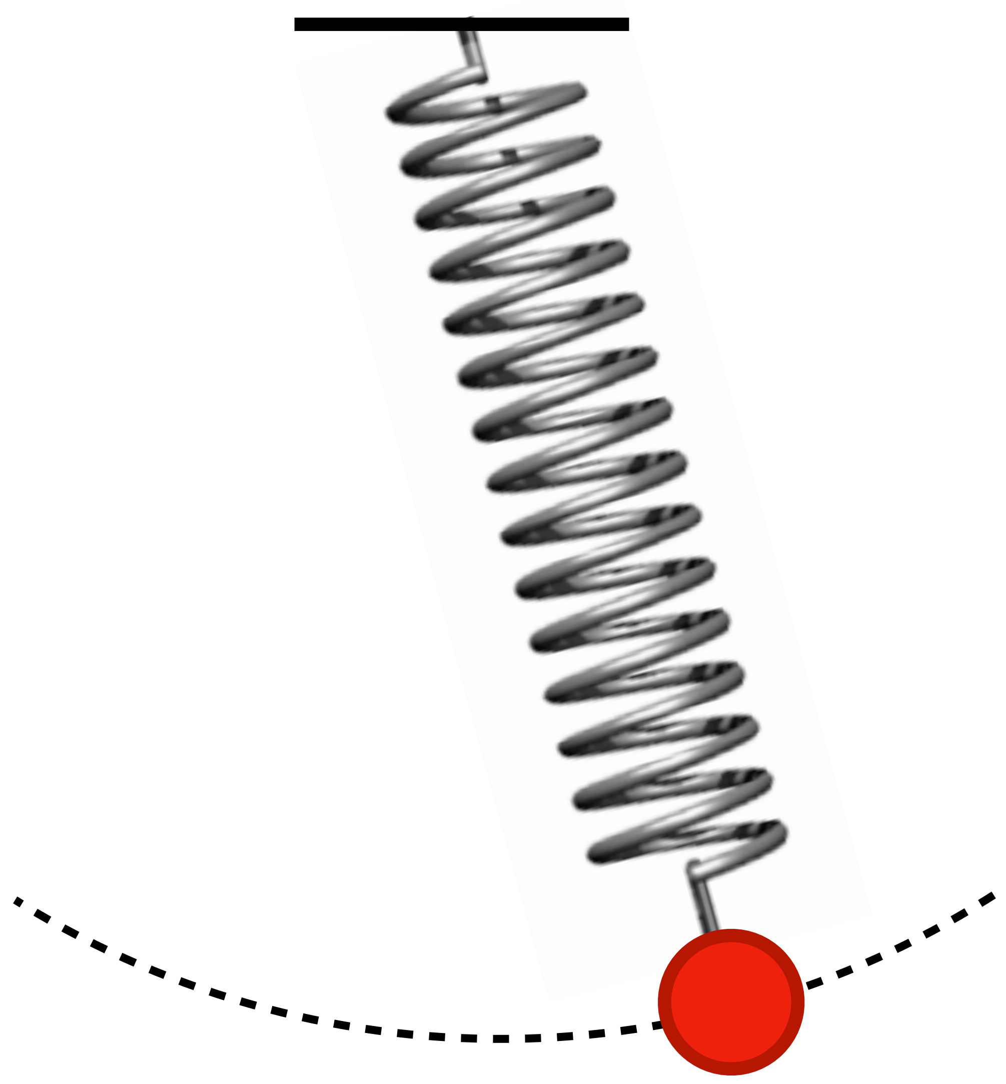
$$\ddot{\mathbf{x}} = \mathbf{g} - m^{-1}\, k_s\, (\|\mathbf{x}_{ij}\| - \ell_0)\, \hat{\mathbf{x}}$$

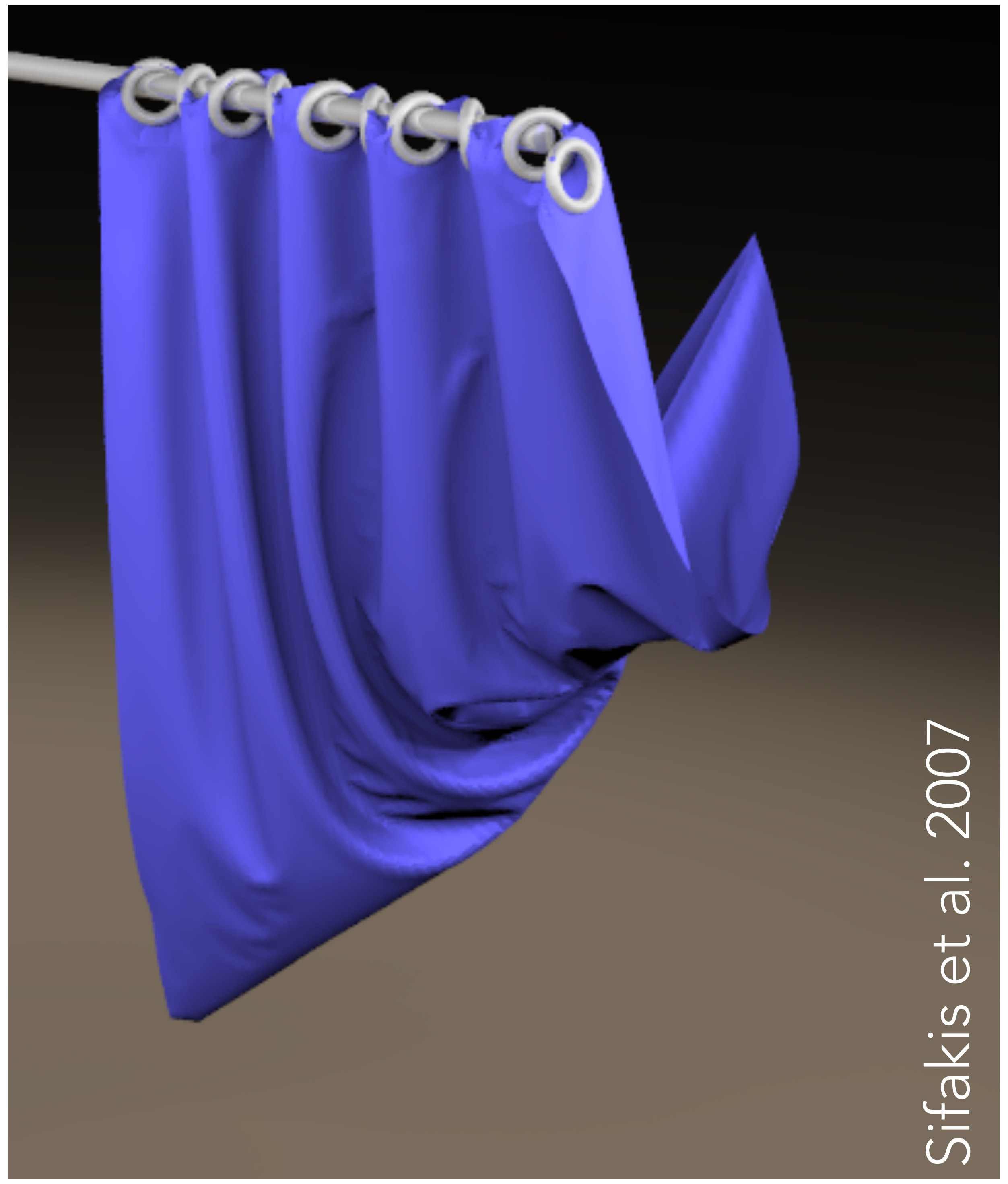
Constrained equations of motion:

$$\ddot{\mathbf{x}} = \mathbf{g} + m^{-1}\, \lambda\, \hat{\mathbf{x}}$$
$$\|\mathbf{x}_{ij}\| = \ell_0$$

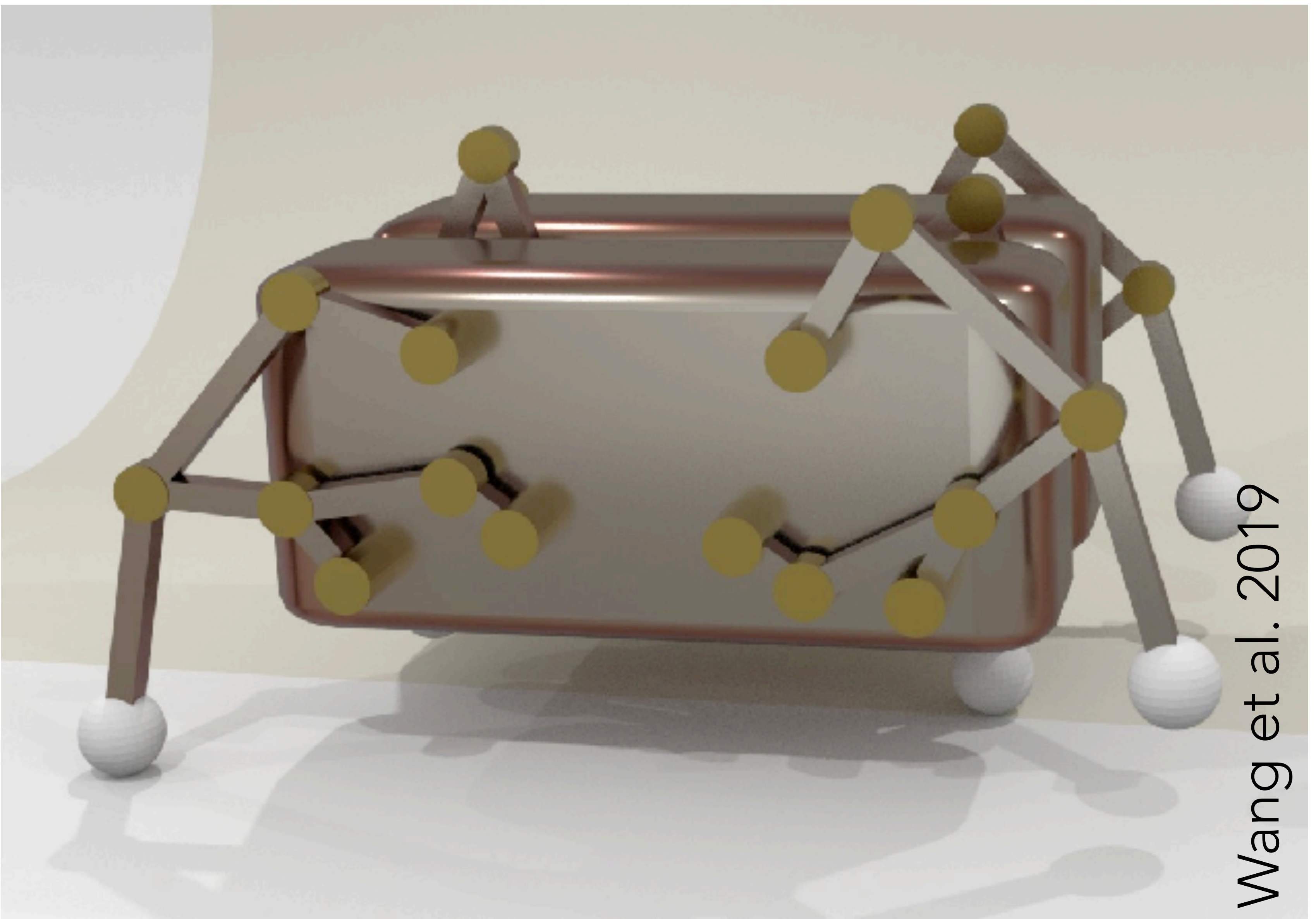
- One new unknown: constraint force magnitude λ.
- One new equation: constraint $\|\mathbf{x}_{ij}\| = \ell_0$.

λ is such that constraint remains satisfied over time…





Sliding on a fixed
line / curve / surface



Joints between
rigid parts

Inextensible cloth

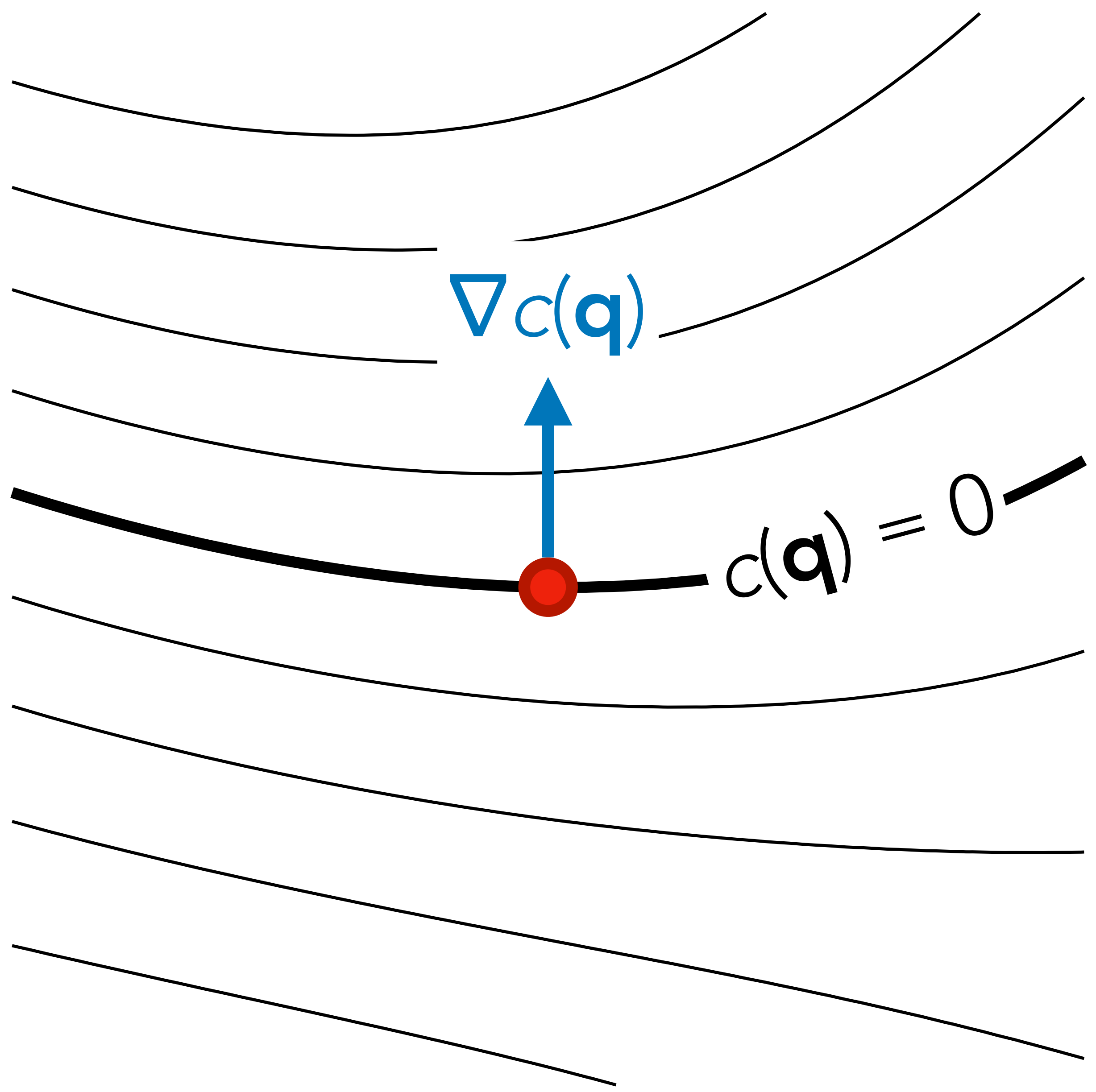
In general, we may have lots of constraints on the system, each of the form

$$c_j(\mathbf{q}) = 0$$

Constraint force:

$$\mathbf{f}_j = \lambda_j \nabla c_j(\mathbf{q})$$

Force is orthogonal to constraint surface
⇒ only resists moving away from constraint, not along constraint



Exercise: verify that the inextensible spring constraint from before is of this form.

$$c_j(\mathbf{q}) = 0$$
$$\mathbf{f}_j = \lambda_j \nabla c_j(\mathbf{q})$$

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\left(\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) + \sum \mathbf{f}_j\right)$$

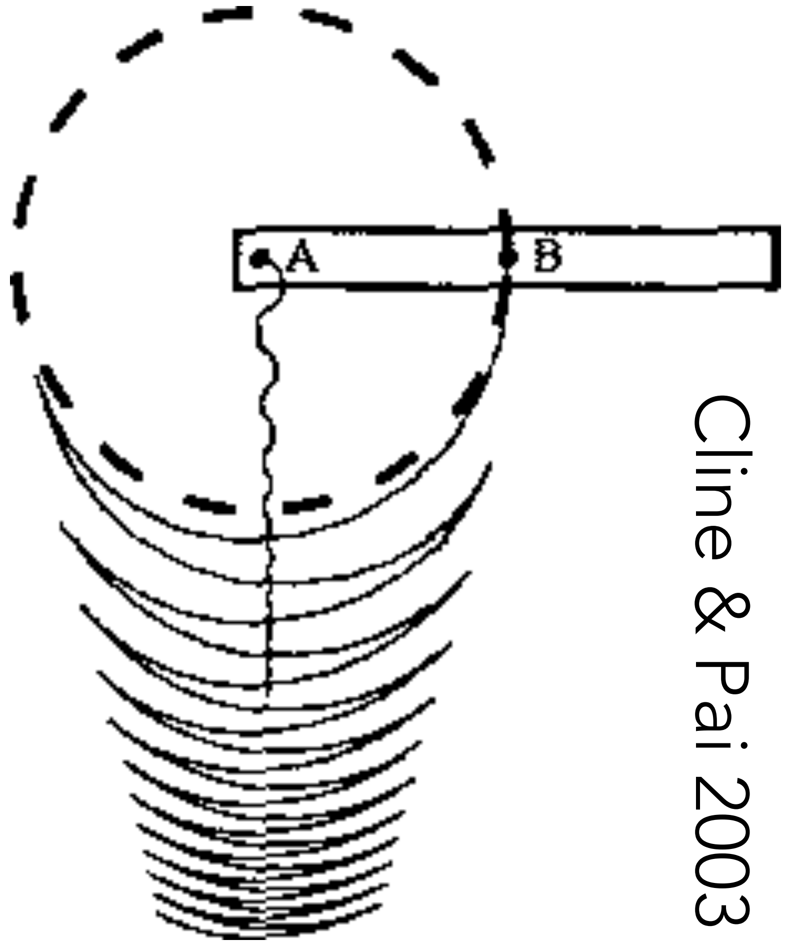
How to actually do time stepping of such a system?

- Try to estimate instantaneous λ_j at each t_n $\Rightarrow$ drift

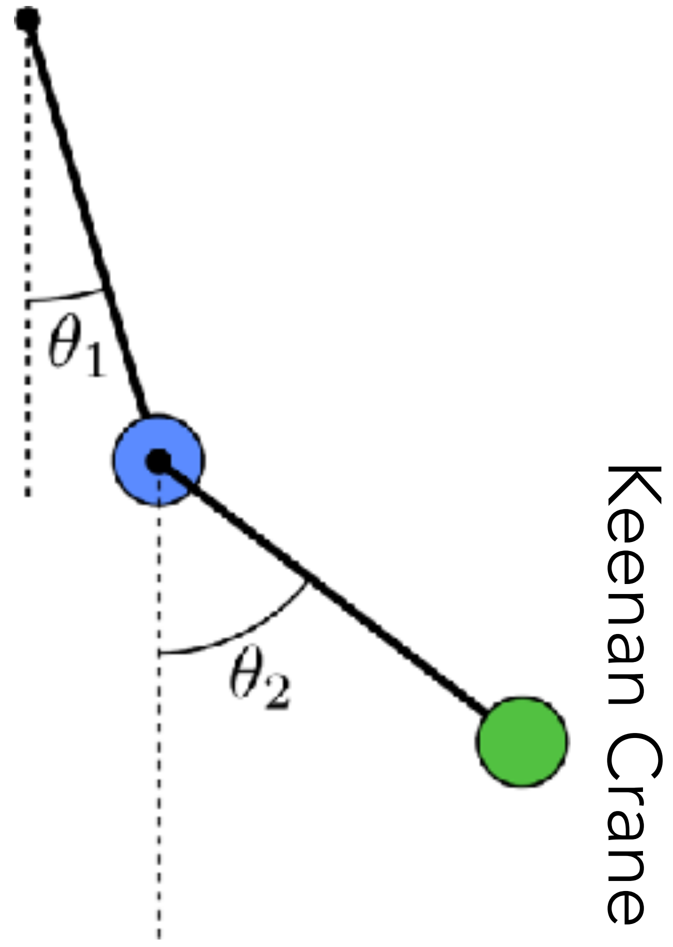
- Replace with **penalty force**: $\lambda_j = -k\, c_j(\mathbf{q})$ $\Rightarrow$ soft constraints

- Choose parameterization that automatically satisfies constraints
  $\Rightarrow$ **reduced coordinates**

- Treat constraint forces implicitly: solve for all λ_j's so that all $c_j(\mathbf{q}_{n+1}) = 0$



Cline & Pai 2003



Keenan Crane

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\left(\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) + \sum \lambda_j \nabla c_j(\mathbf{q})\right)$$

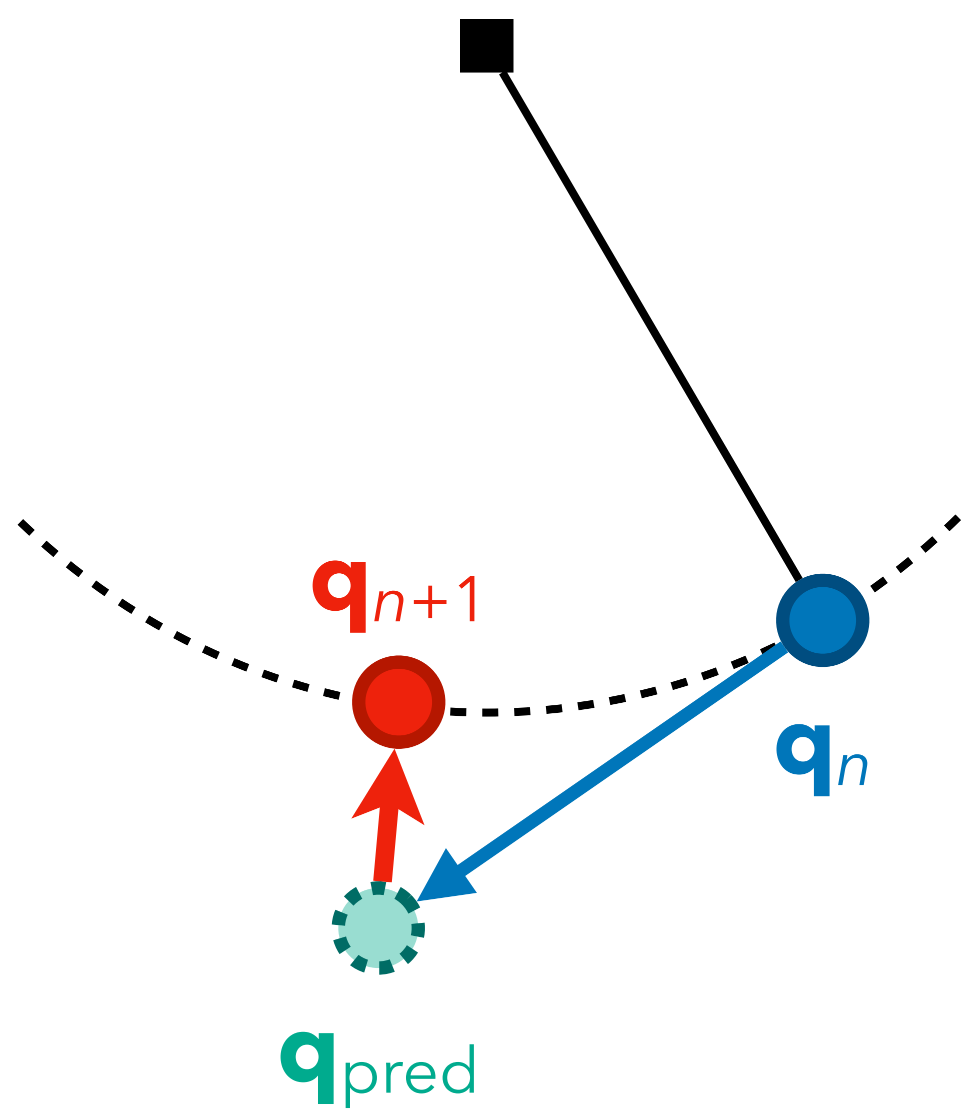
$$c_j(\mathbf{q}) = 0$$

Suppose we treat the external forces explicitly and the constraint forces implicitly.

We can also eliminate $\mathbf{v}_{n+1}$:

$$\mathbf{q}_{n+1} = \mathbf{q}_{\text{pred}} + \sum \mathbf{M}^{-1} \lambda_j \nabla c_j(\mathbf{q}_{n+1})\, \Delta t^2$$
$$c_j(\mathbf{q}_{n+1}) = 0$$

where $\mathbf{q}_{\text{pred}} = \mathbf{q}_n + \mathbf{v}_n \Delta t + \mathbf{M}^{-1}\mathbf{f}(\mathbf{q}_n, \mathbf{v}_n)\, \Delta t^2$.



Solve for $\mathbf{q}_{n+1}$ and $\lambda_1, \lambda_2, \ldots$ simultaneously using Newton's method

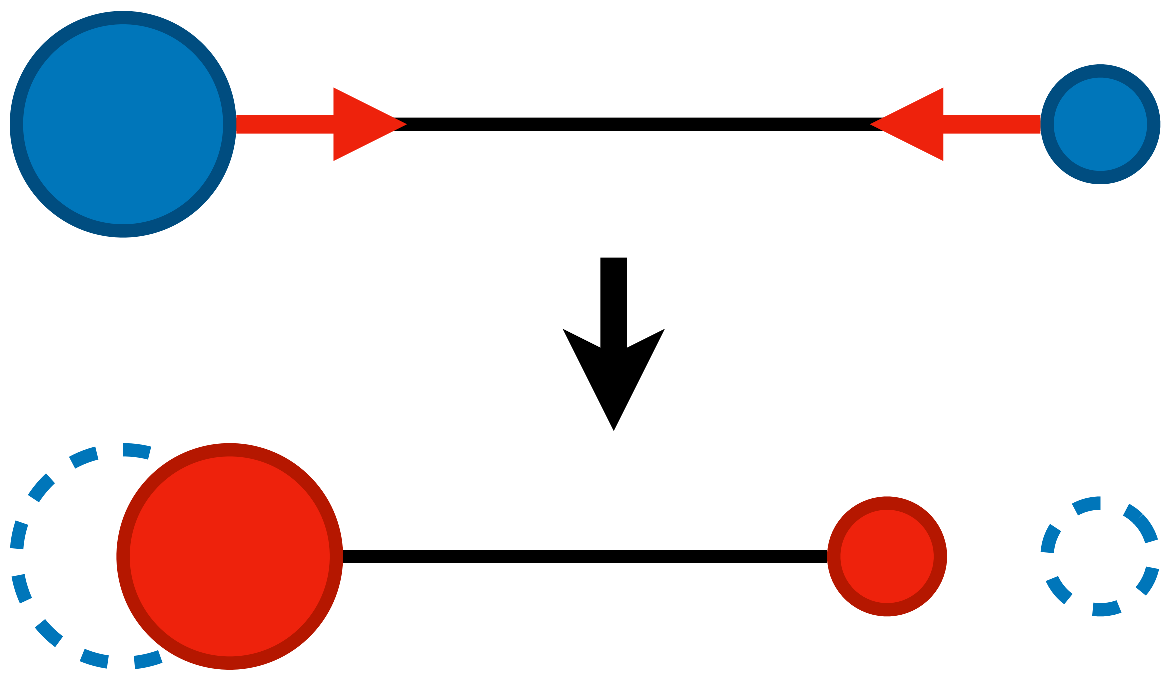
…Then update $\mathbf{v}_{n+1} = (\mathbf{q}_{n+1} - \mathbf{q}_n)/\Delta t$

Position-based dynamics

For real-time graphics, solving a big linear system for all λ's is too expensive!
But it's easy to solve one constraint at a time:

Example: Inextensible spring between particles i and j

$$\|\mathbf{x}_{ij}\| = \ell_0$$



$$\mathbf{f}_{ij} = \lambda\, \hat{\mathbf{x}}_{ij}$$

Recall $\mathbf{q}_{n+1} = \mathbf{q}_{\text{pred}} + \sum \mathbf{M}^{-1} \lambda_j \hat{\mathbf{x}}_{ij} \Delta t^2$

$$\Delta\mathbf{q}_{n+1} = \mathbf{M}^{-1} \Delta\lambda\, \hat{\mathbf{x}}_{ij} \Delta t^2$$

Find $\Delta\lambda$ which makes updated positions satisfy $\|\tilde{\mathbf{x}}_{ij} + \Delta\mathbf{x}_{ij}\| = \ell_0$

In general, we have a guess of the next positions: $\tilde{\mathbf{q}}$

1. Applying a constraint force $\Delta\lambda_j$ changes the positions by $\Delta\mathbf{q} = \mathbf{M}^{-1}\,\Delta\lambda_j\,\nabla c_j(\tilde{\mathbf{q}})\,\Delta t^2$
2. Solve for $\Delta\lambda_j$ so that $c_j(\tilde{\mathbf{q}} + \Delta\mathbf{q}) = 0$
3. Update the positions (**constraint projection**): $\tilde{\mathbf{q}} \leftarrow \tilde{\mathbf{q}} + \Delta\mathbf{q}$
4. Repeat for other constraints

Projecting one constraint makes other constraints violated!

- Loop over all constraints = 1 iteration. Have to repeat many iterations
- If not enough iterations, constraints appear soft!

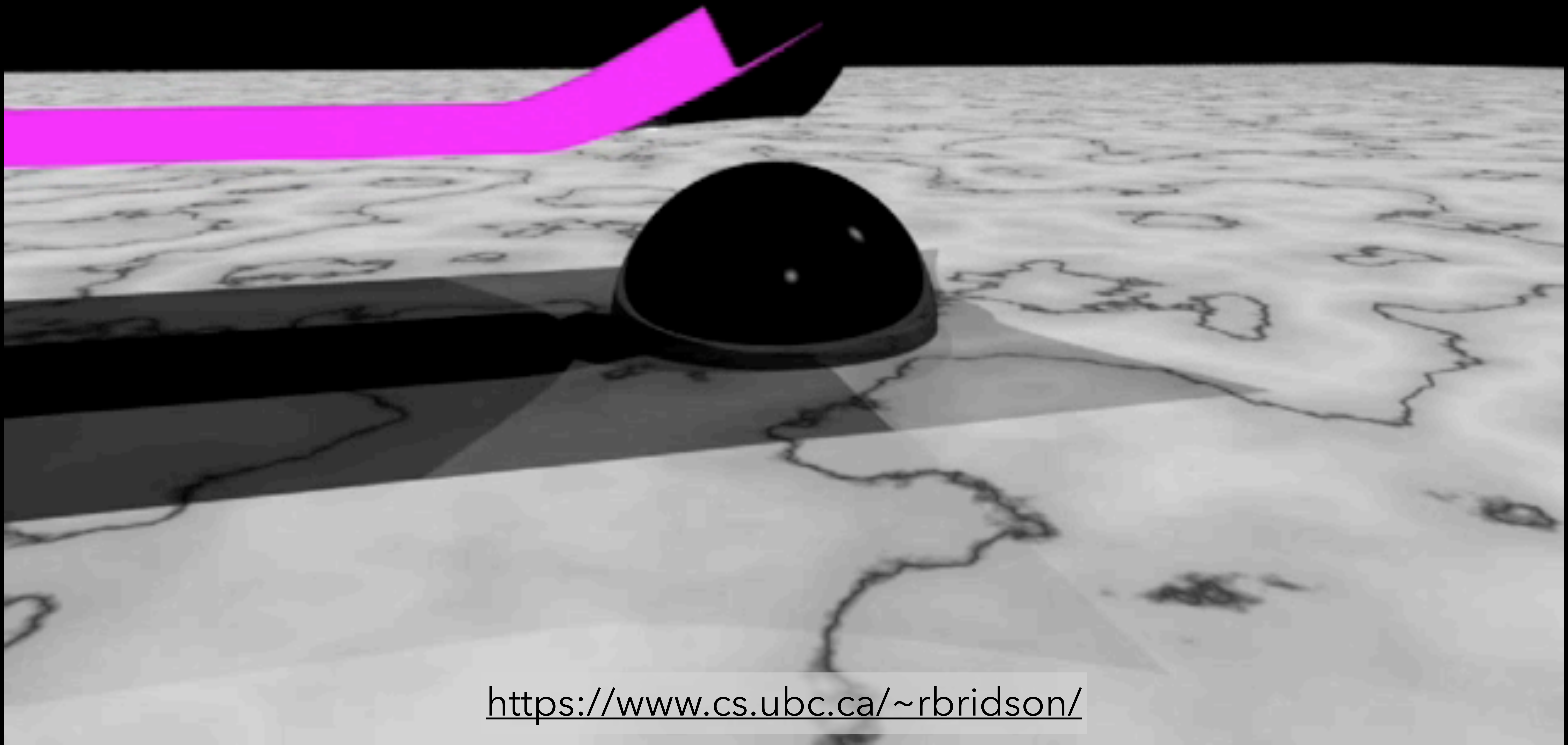
https://www.youtube.com/watch?v=j5igW5-h4ZM

Müller et al. 2006

https://www.youtube.com/watch?v=j5igW5-h4ZM

Müller et al. 2006

Collisions

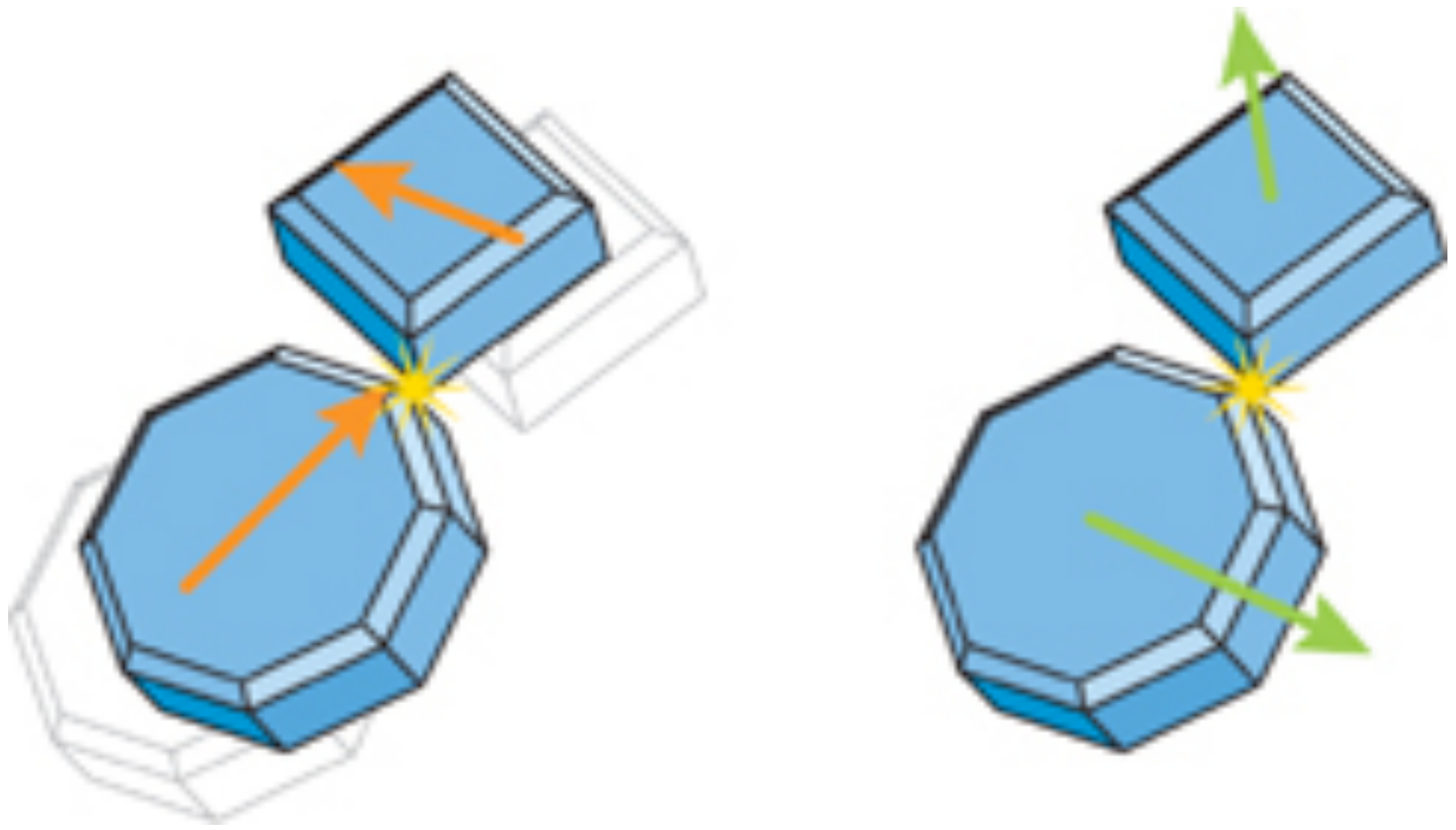


https://www.cs.ubc.ca/~rbridson/

Bridson et al. 2002

Collision detection: find out which particles / bodies / etc. are colliding

Purely a geometric problem

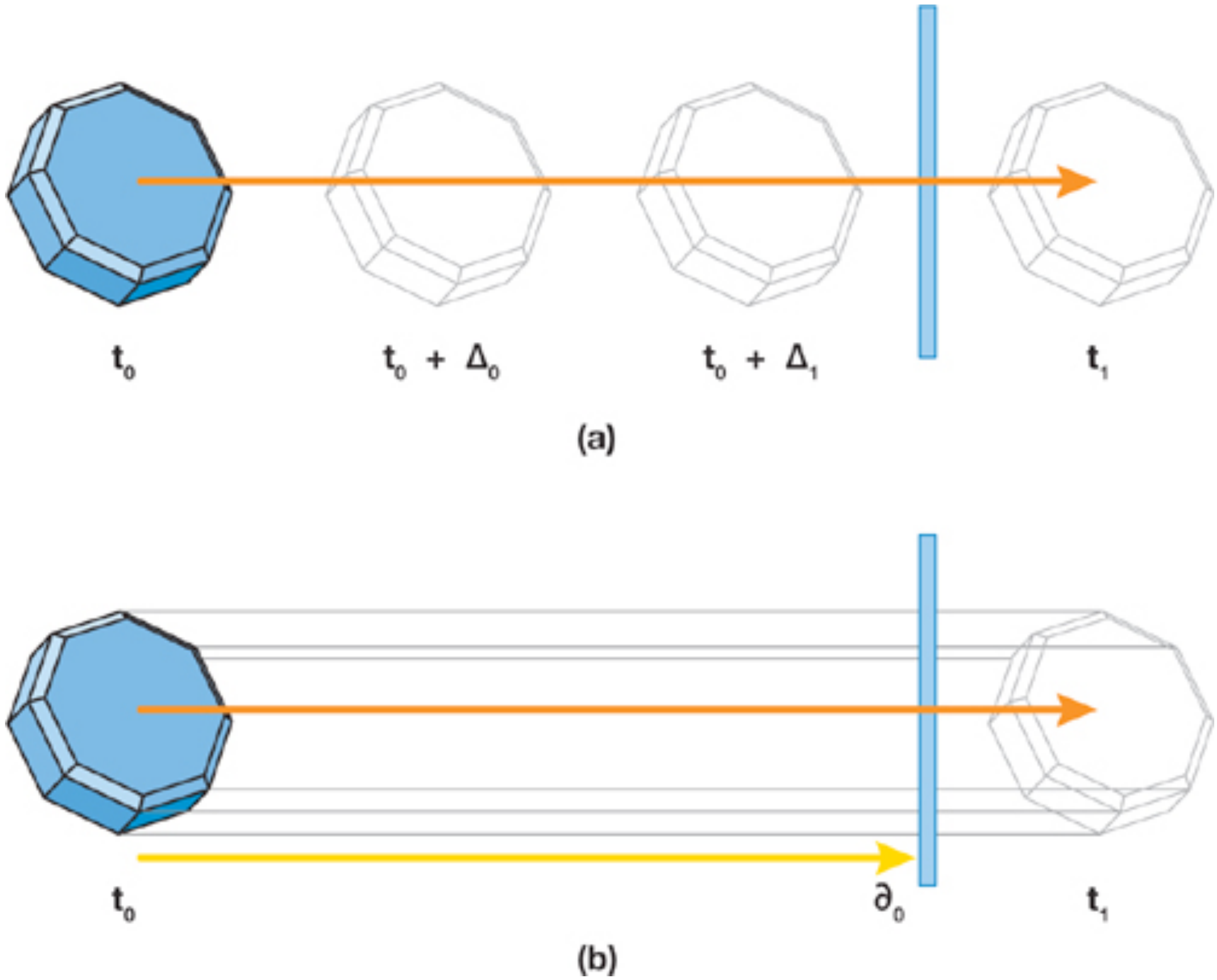


Peter Kipfer

Collision response: figure out how to update their velocities / positions

Involves physics of contact forces, friction, etc.

Collision detection: discrete vs. continuous



Peter Kipfer

Example: Suppose I have an infinite cylinder along the x-axis with radius R.

I also have a particle with radius r moving to positions $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ at times $t_0, t_1, t_2, \ldots$

1. How can I do discrete collision detection between the particle and the cylinder?

2. How can I do continuous collision detection between them?

3. If I model a sheet of cloth as a mass-spring system, is it enough to check that none of the particles are colliding with the cylinder?