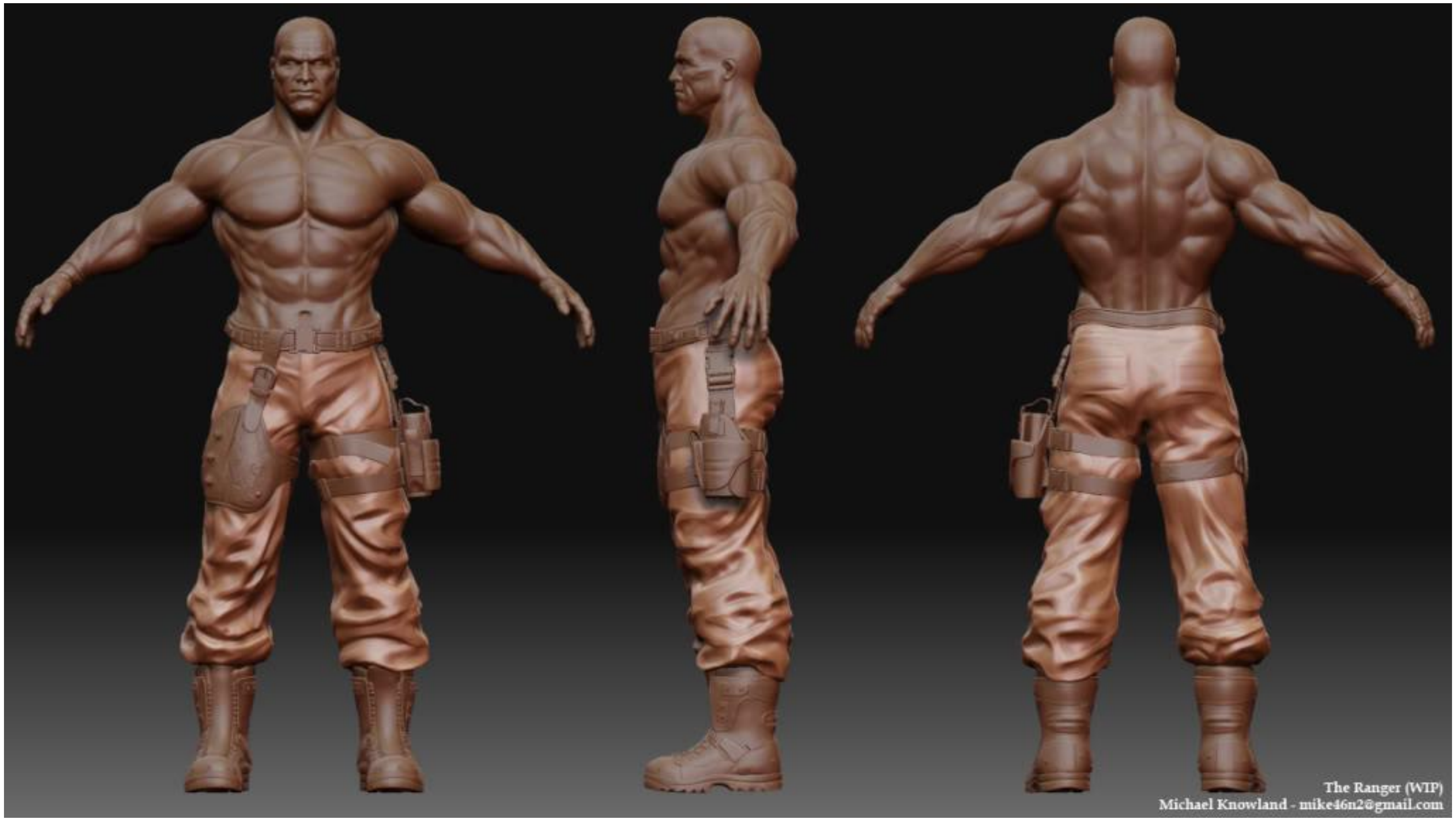
An introduction to Physically-based animation and ODEs

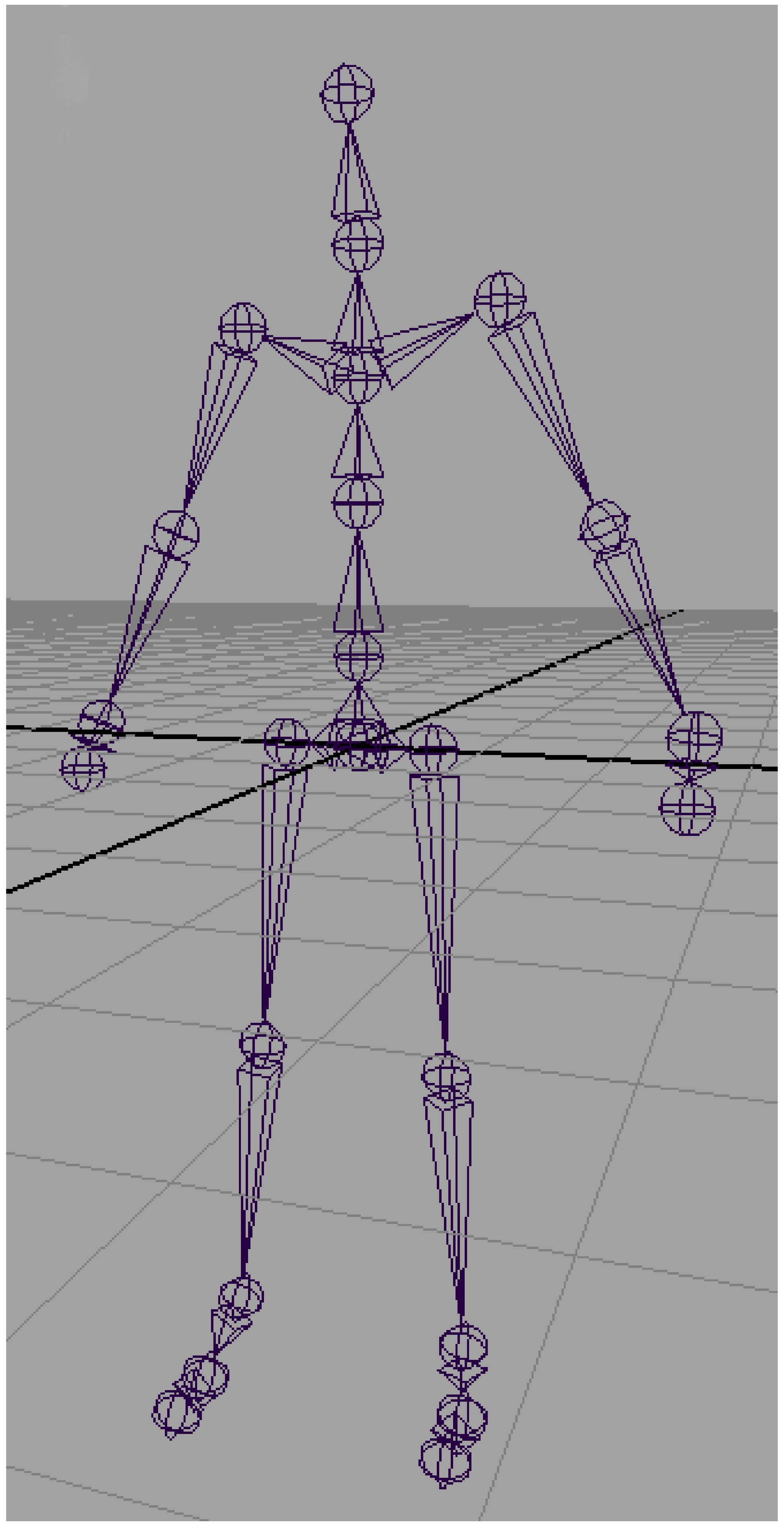
Last Time: Skeletal Animation

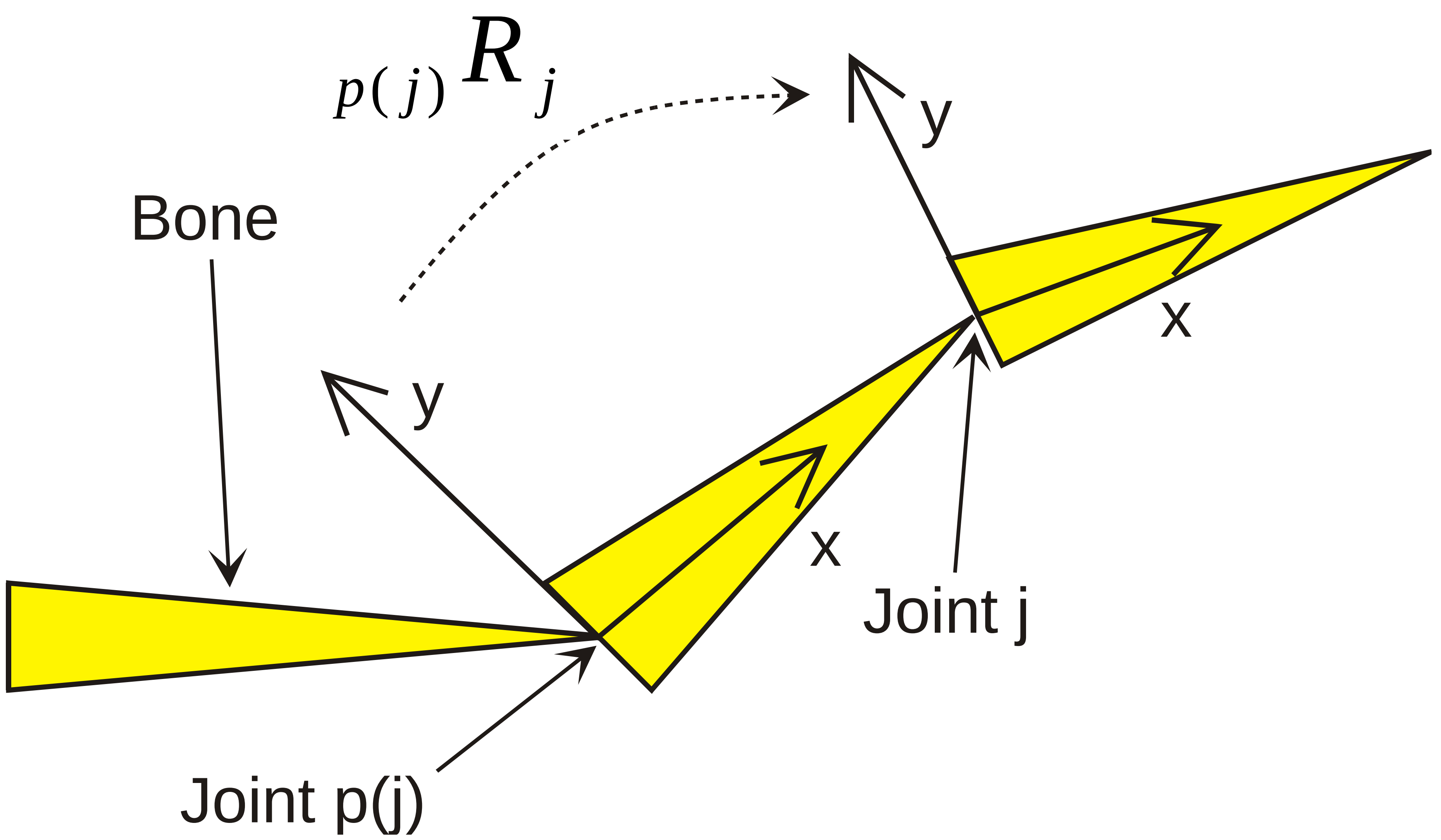
Key idea: animate just the skeleton (<< DOFs), have mesh "follow" automatically



Forward Kinematics (FK)

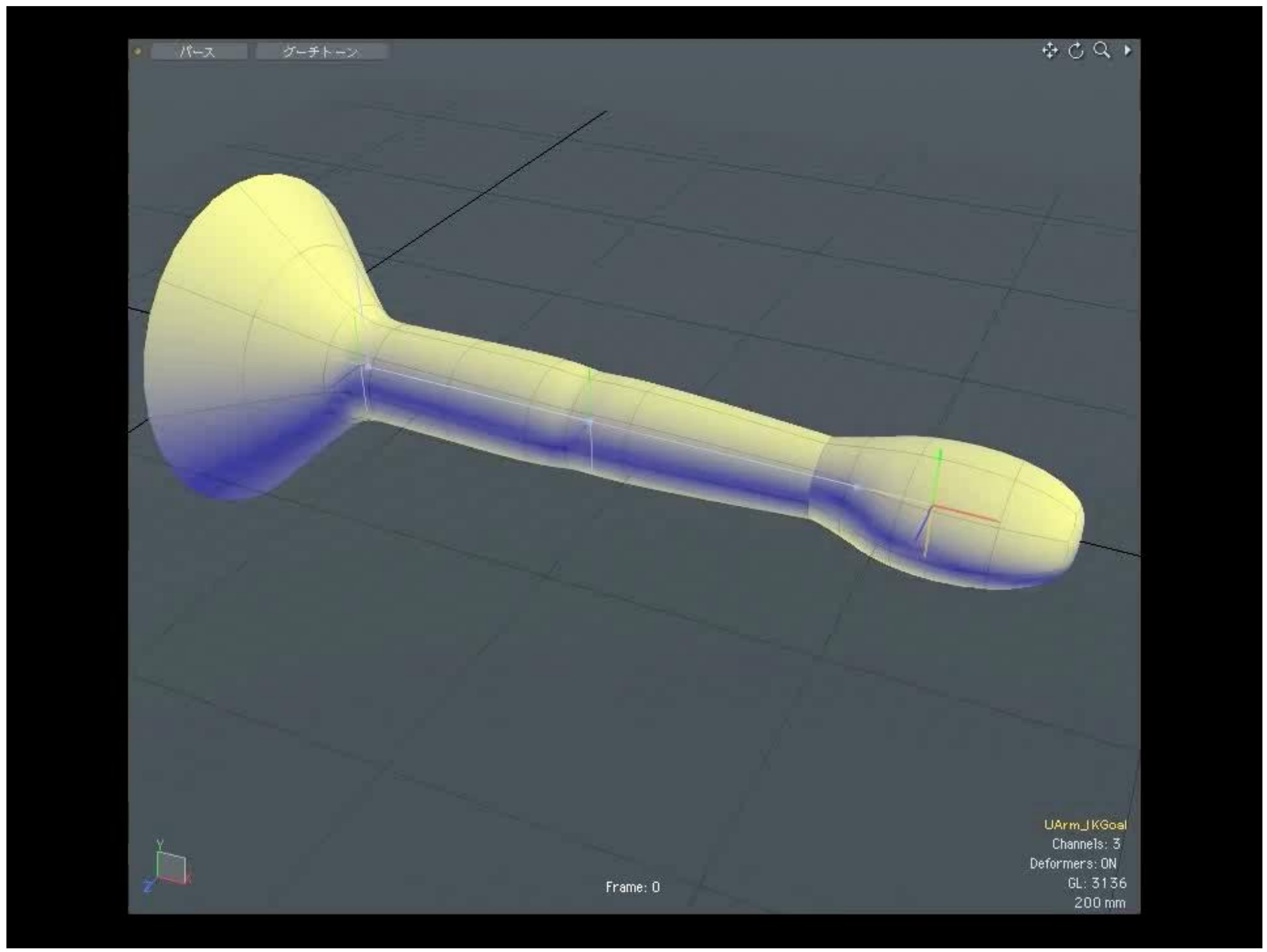
- Given joint angles, compute configuration of the skeleton





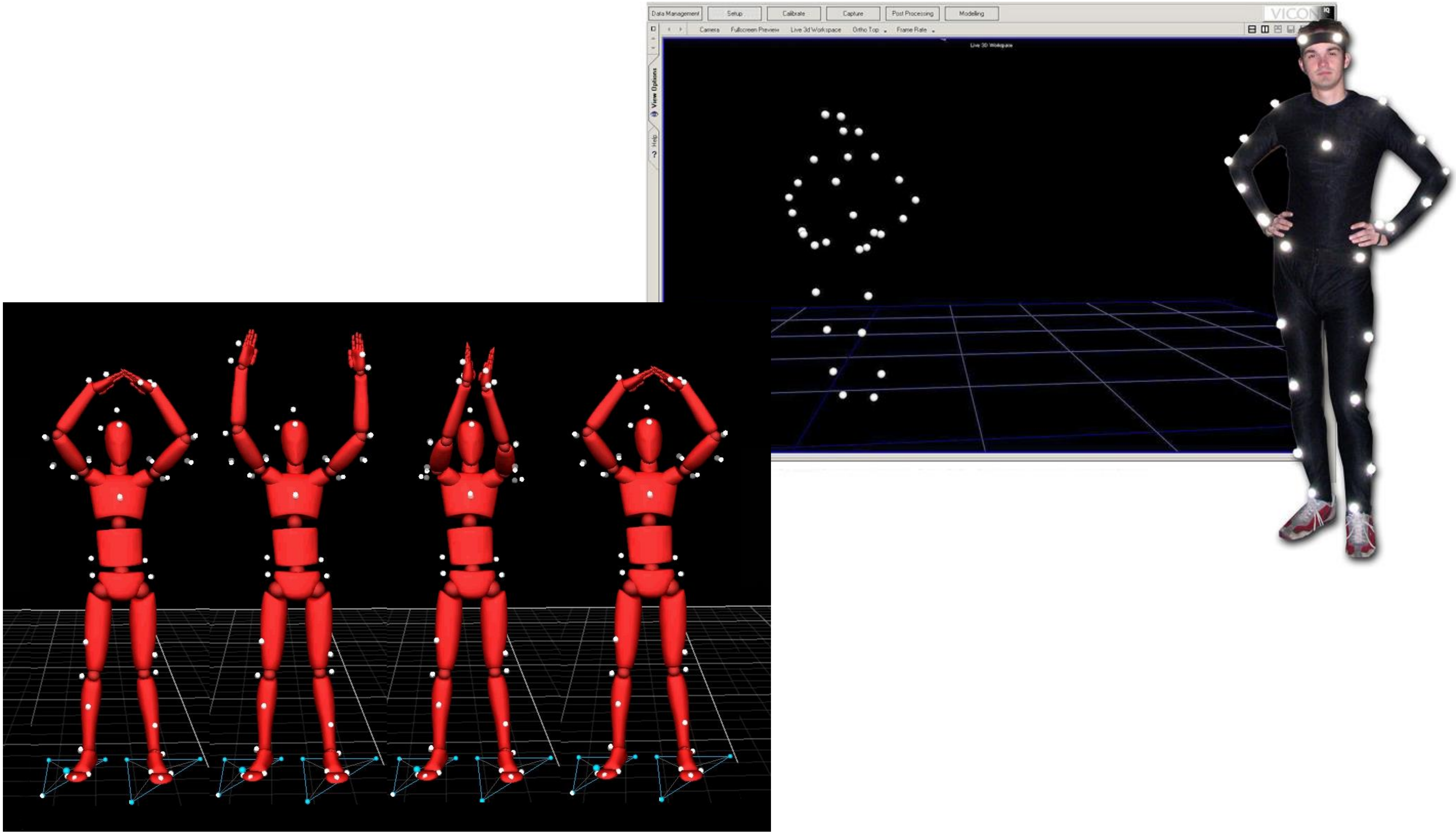
Inverse Kinematics (IK)

- Given goal position for “end effector” compute joint angles
- Very important technique in animation & robotics!

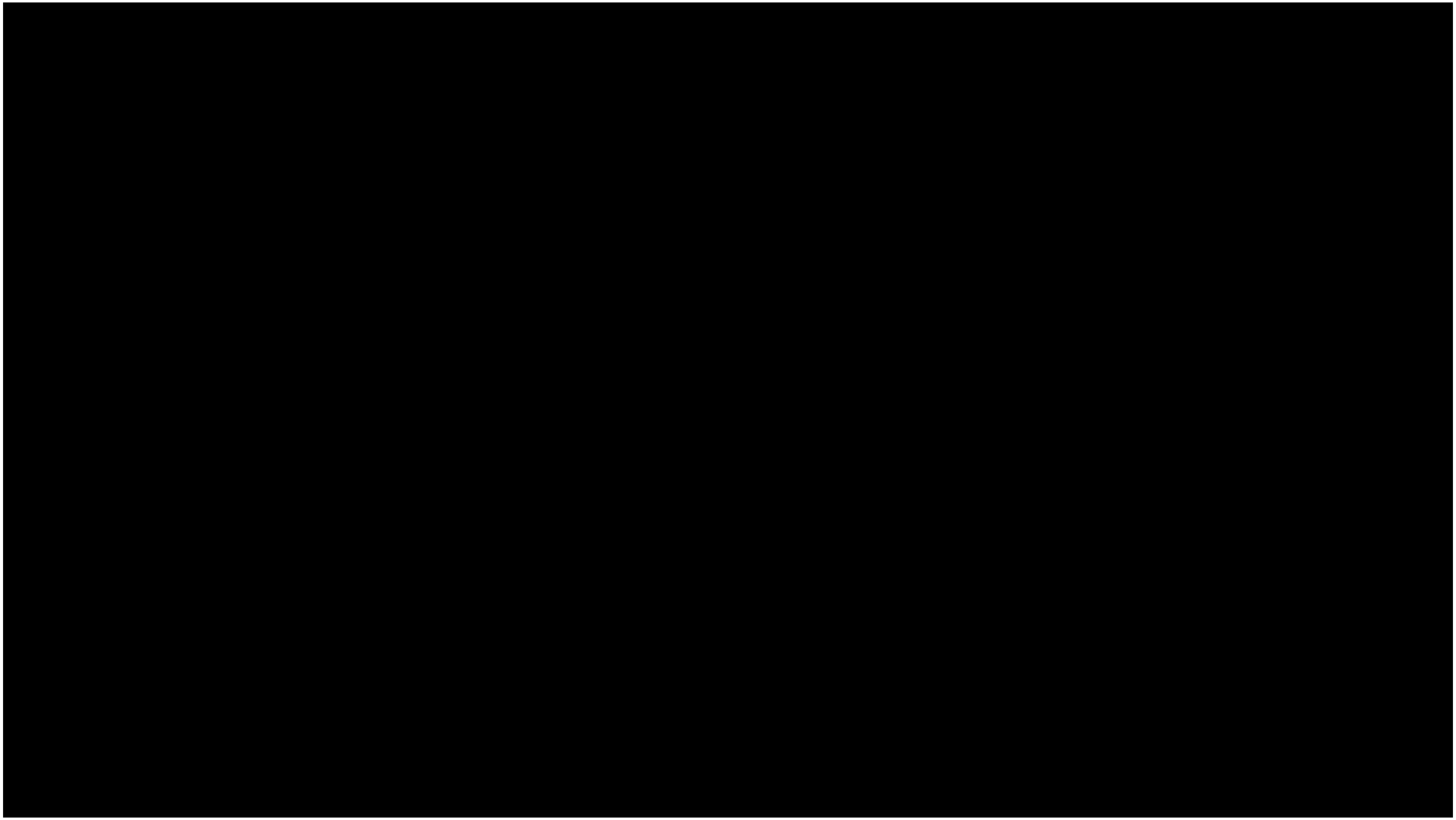


- Many algorithms: analytic formulations (for specific cases), energy-based methods, etc.

Full-body Motion Capture



Full-body Motion Capture – Example 1

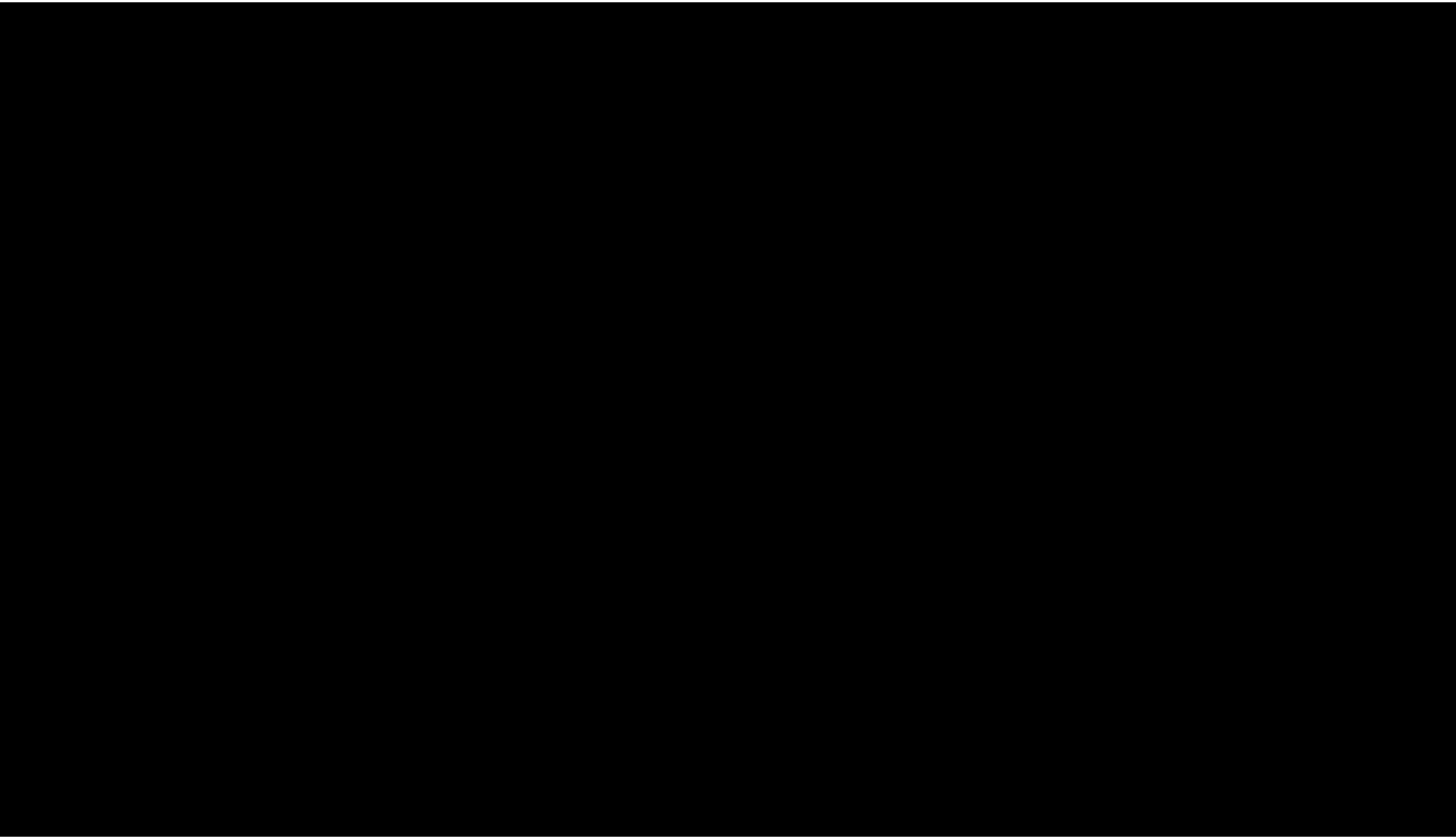


https://www.youtube.com/watch?v=zQPfxcQKr0Q

Motion Capture – Example 2

https://www.youtube.com/watch?v=txoEDIdbUrg

Motion Capture – Example 2



https://www.youtube.com/watch?v=txoEDIdbUrg

A general formulation: optimization-based IK

- Basic idea behind IK algorithm
  - write down distance between (FK-) transformed point $p(t)$ and target $\tilde{p}$ and set up objective
  - Minimize objective with respect to angles using *numerical optimization*

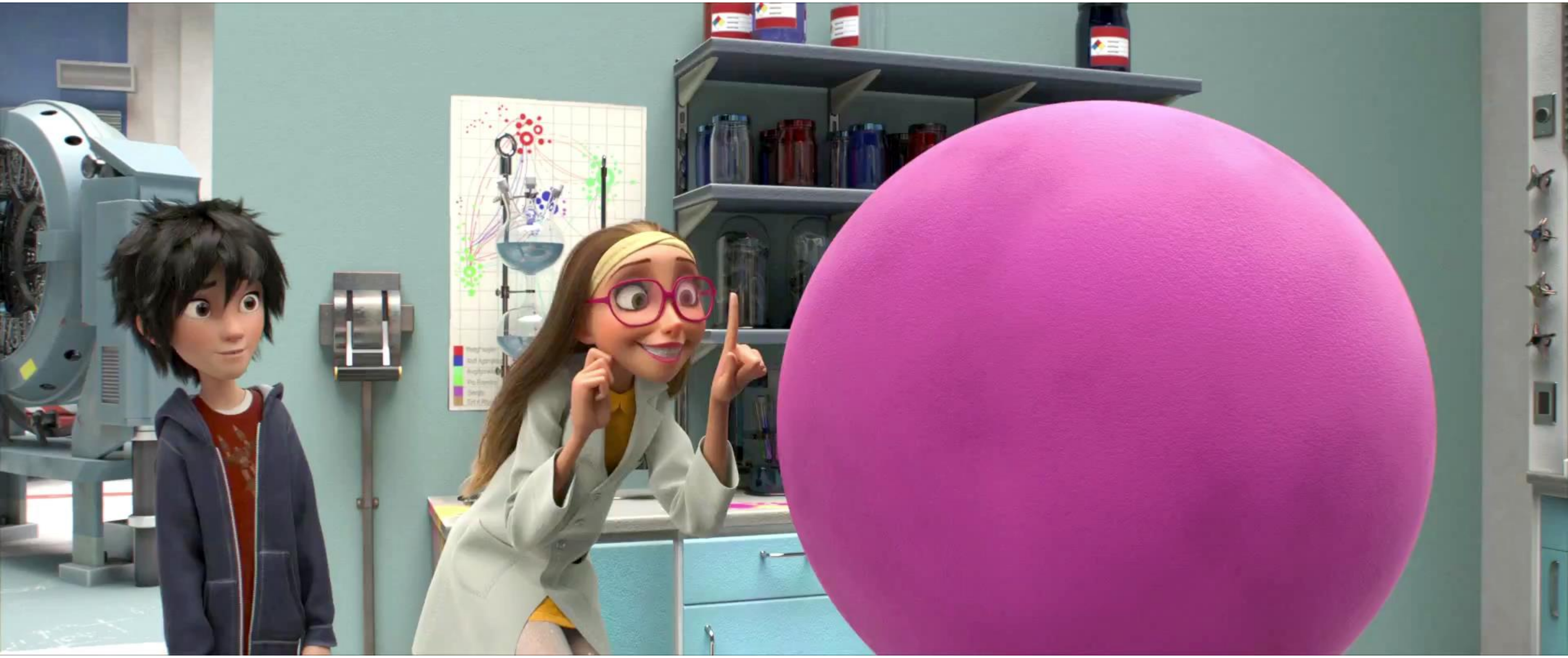
- Objective?

$$\boldsymbol{f_0(\theta)} = \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{p(\theta)} - \widetilde{\boldsymbol{p}})^{\boldsymbol{T}}(\boldsymbol{p(\theta)} - \widetilde{\boldsymbol{p}})$$

- Constraints?
  - We could limit joint angles

Physics-based Animation

Effects in Big Hero Six



Kinematics vs Dynamics

kinematics

/ˌkɪnɪˈmatɪks, ˌkʌɪnɪˈmatɪks/

noun

the branch of mechanics concerned with the motion of objects without reference to the forces which cause the motion.

dynamics

/dʌɪˈnamɪks/

noun

noun: **dynamics**; plural noun: **dynamics**

the branch of mechanics concerned with the motion of bodies under the action of forces.

The Animation Equation

- The *rendering equation*
  - Rasterization and path tracing give approximate solutions to the rendering equation
- What's the *animation equation?*
  - Large spectrum of physical materials and phenomena: solids, fluids, elasticity, plasticity, magnetism, gravity, …
  - Leverage tools from computational physics: dynamical descriptions, numerical integration, etc.

Connection between Force and Motion

Newton's Second Law of Motion

“A change in motion is proportional to the motive force impressed and takes place along the straight line in which that force is impressed.”

—Sir Isaac Newton, 1687

The Animation Equation

mass

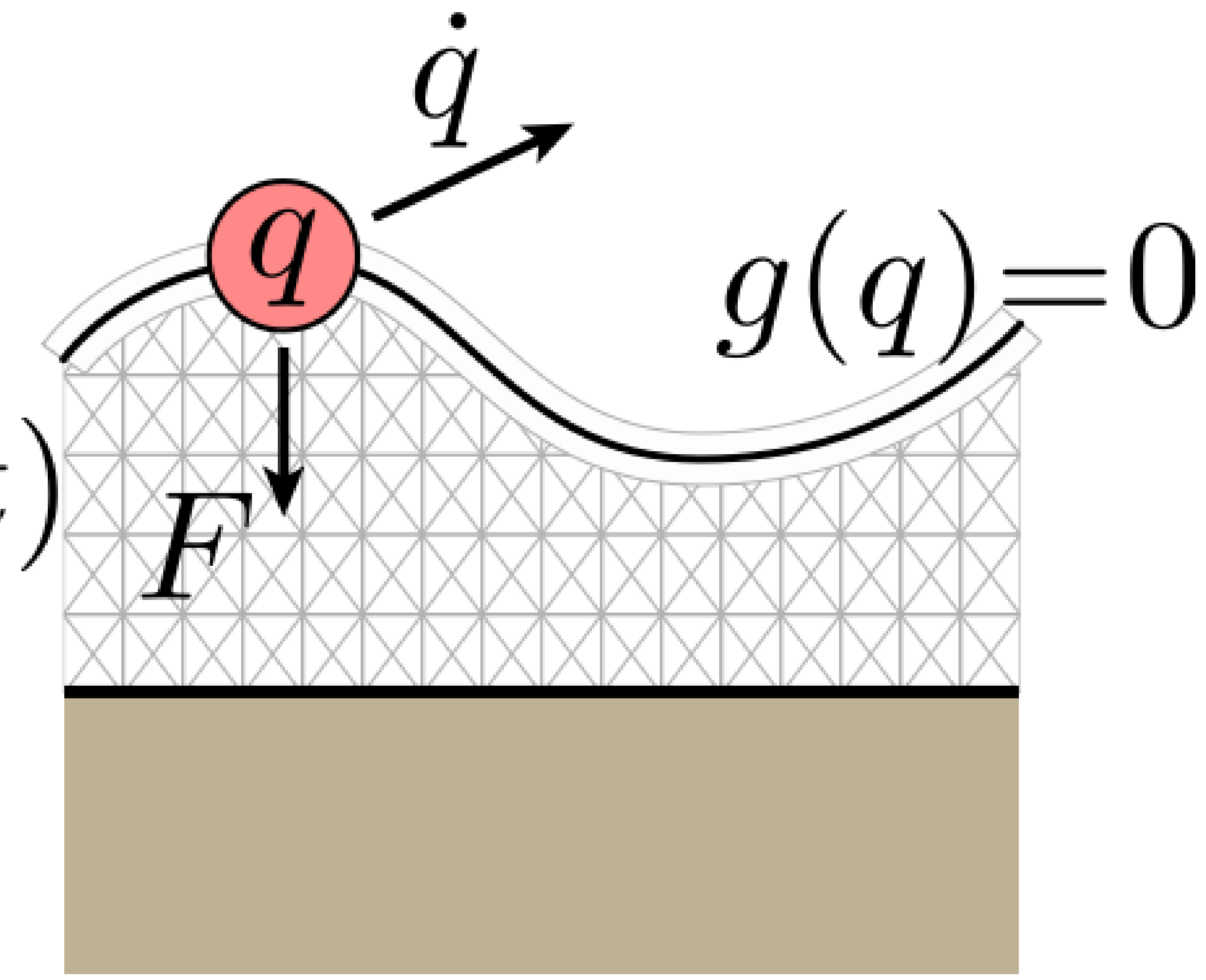
acceleration

$$F = ma$$

force

Often called the *Equations of Motion (EoM)*

The Animation Equation



There is more to be said

- Every system has a *configuration* $q(t)$
- It also has a *velocity* $\dot{q} := \frac{d}{dt}q$
- It has some kind of *mass* M
- There are *forces* F acting on the system (e.g., gravity)
- And also potentially some *constraints* $g(q, \dot{q}, t) = 0$

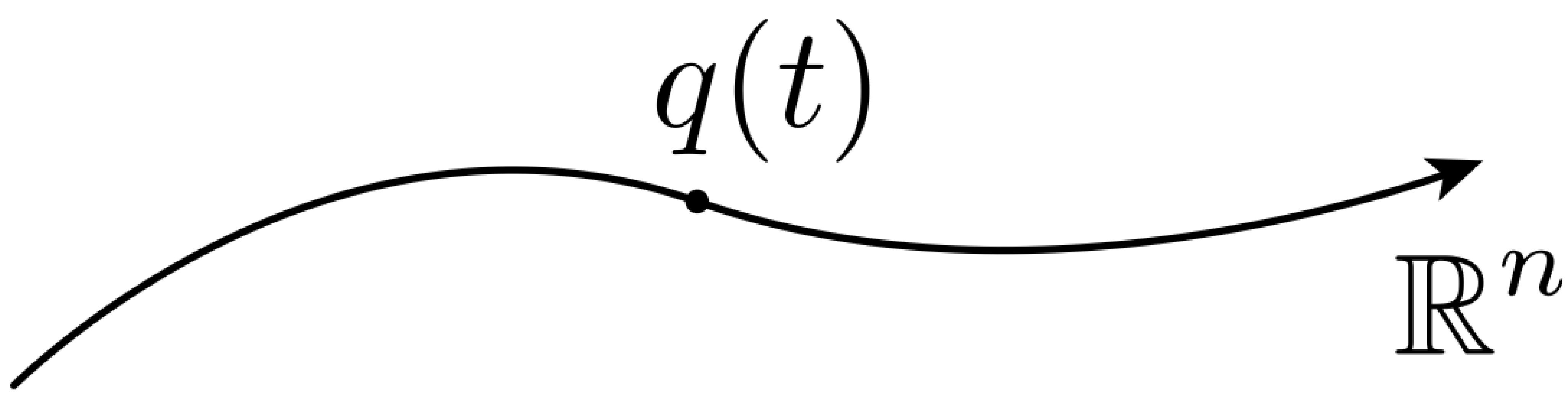
Can write Newton's 2nd law as $\ddot{q} = F/m$

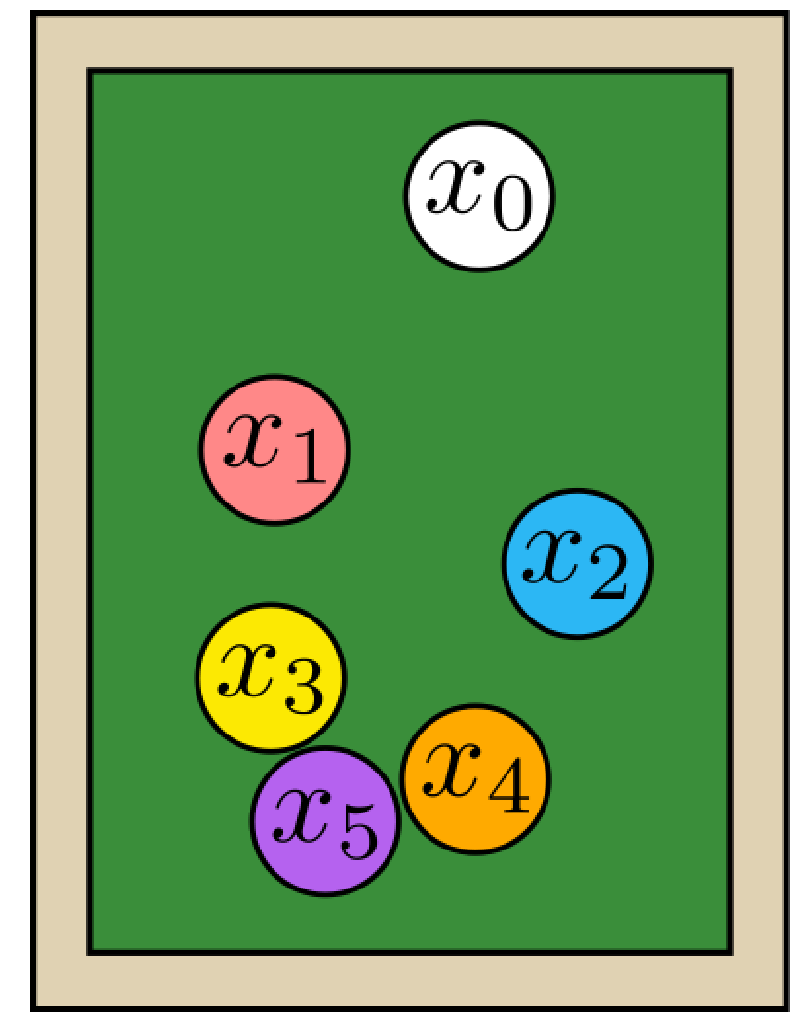
- acceleration is 2nd time derivative of configuration
- ultimately, we want to solve for the configuration $q(t)$

Generalized Coordinates

- Often describing systems with many moving parts
- E.g., a collection of billiard balls, each with position x_i
- Collect them all into a single vector of *generalized coordinates:*

$$q = (x_0, x_1, \ldots, x_n)$$



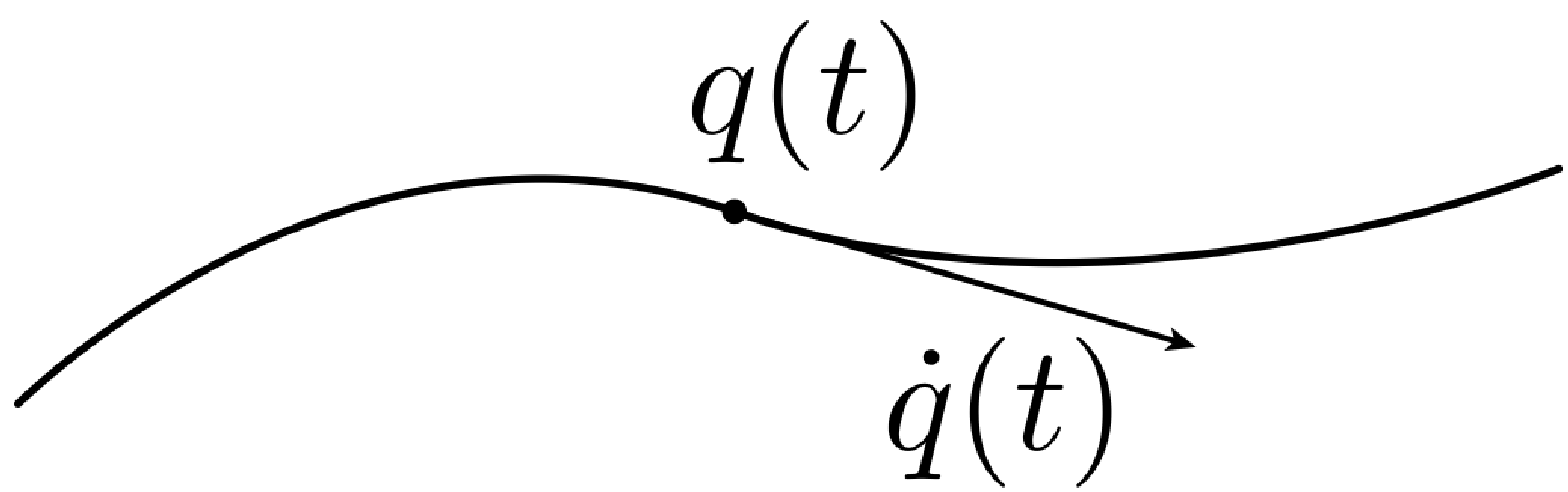


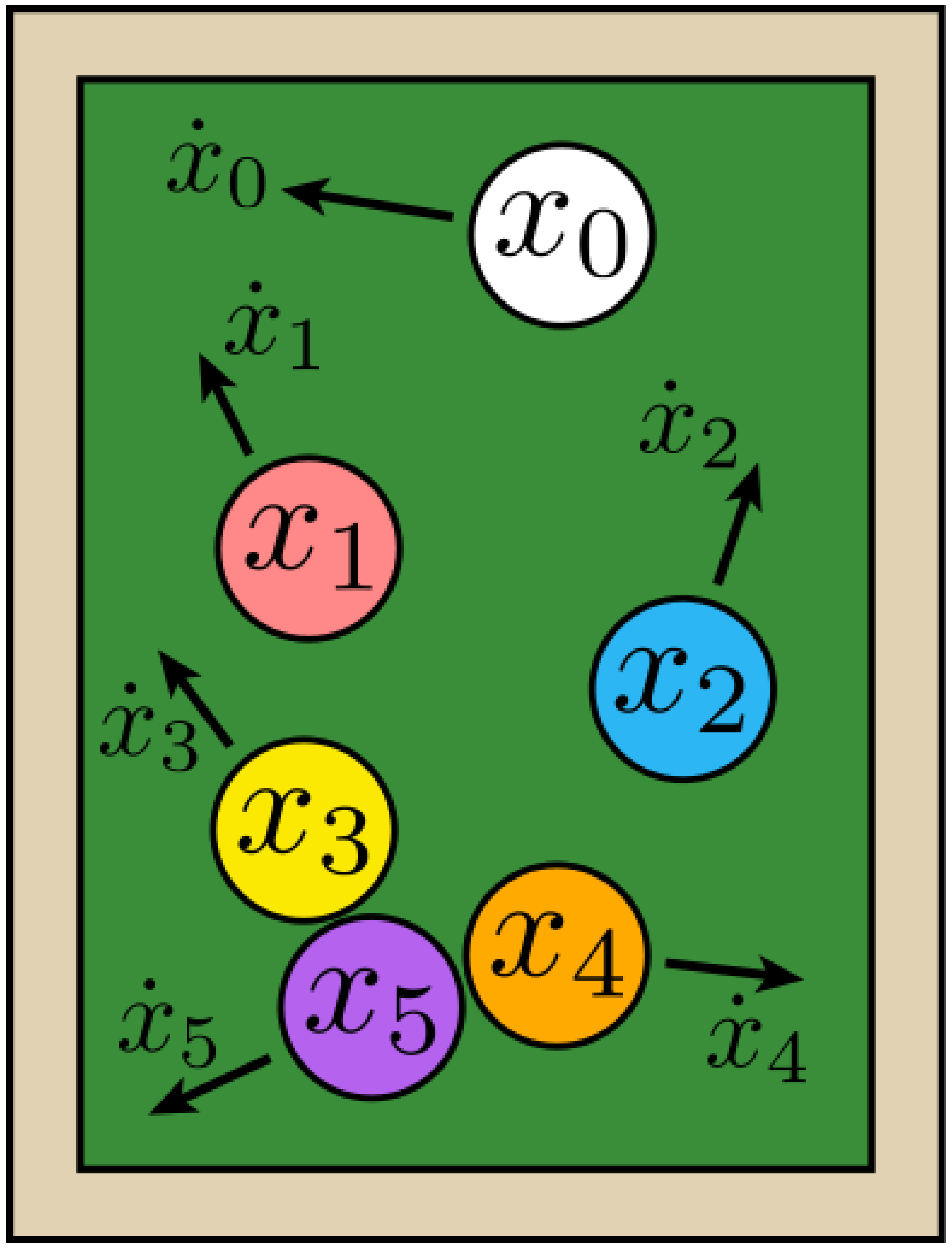
- Can think of q as a *single point* moving along a trajectory in R^n
- Naturally maps to computer implementation:
  - variables are "stacked" into one large vector
  - handed to numerical method for solving equations of motion

Generalized Velocity

- Just the time derivative of the generalized coordinates

$$\dot{q} = (\dot{x}_0, \dot{x}_1, \ldots, \dot{x}_n)$$





Ordinary Differential Equations

- Many dynamical systems can be described via an *ordinary differential equation (ODE)* :

change in configuration over time

velocity function

$$\frac{d}{dt}q = f(q, \dot{q}, t)$$

- Example:

$$\frac{d}{dt}u(t) = au$$

"rate of growth is proportional to value"

- Solution? $u(t) = be^{at}$
- Describes exponential decay (a < 0), or exponential growth (a > 0)
- "Ordinary" means "involves derivatives w.r.t. only one variable"
- We'll talk about multiple partial derivatives (PDEs) in another lecture...

Dynamics via ODEs

- Newton's 2nd law is an ODE as well

$$\ddot{q} = F/m$$

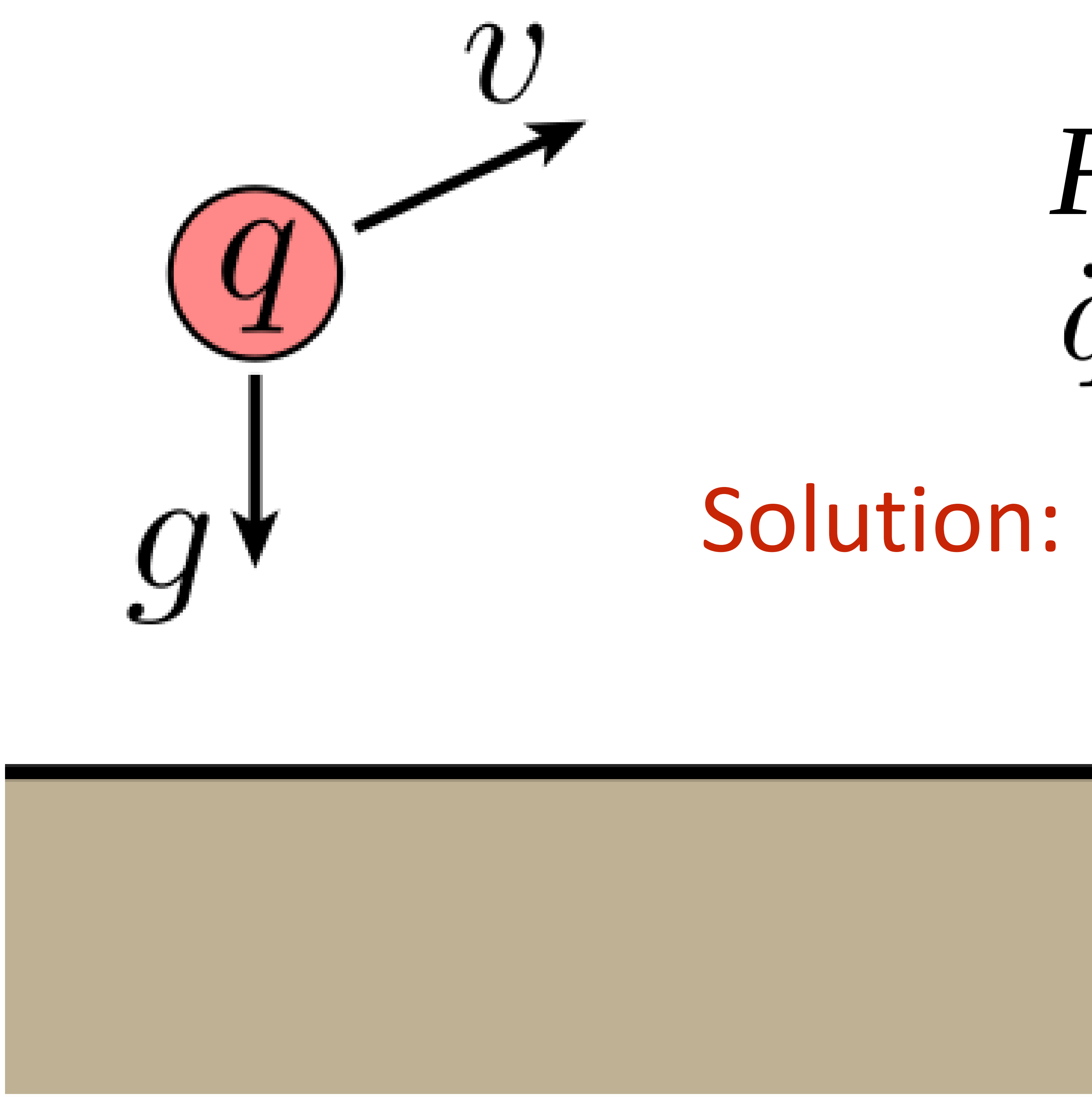
- *Second order* ODE since we differentiate *twice* w.r.t. time
- Can also write as a *system* of two *first order* ODEs, by introducing new variables for velocity:

$$\begin{aligned} \dot{q} &= v \\ \dot{v} &= F/m \end{aligned} \qquad \frac{d}{dt}\begin{bmatrix} q \\ v \end{bmatrix} = \begin{bmatrix} v \\ F/m \end{bmatrix}$$

- This splitting makes it easy to talk about solving these equations numerically (among other things)

Simple Example: Throwing a Rock

- Consider a rock* of mass *m* tossed under force of gravity *g*
- Easy to write dynamical equations, since the only active force is gravity → constant acceleration:



$$F = mg \qquad \dot{q} = v$$
$$\ddot{q} = g \qquad \dot{v} = g$$

or

Solution: $v(t) = v_0 + \frac{t}{m}F$

$$q(t) = q_0 + tv_0 + \frac{t^2}{2m}F$$

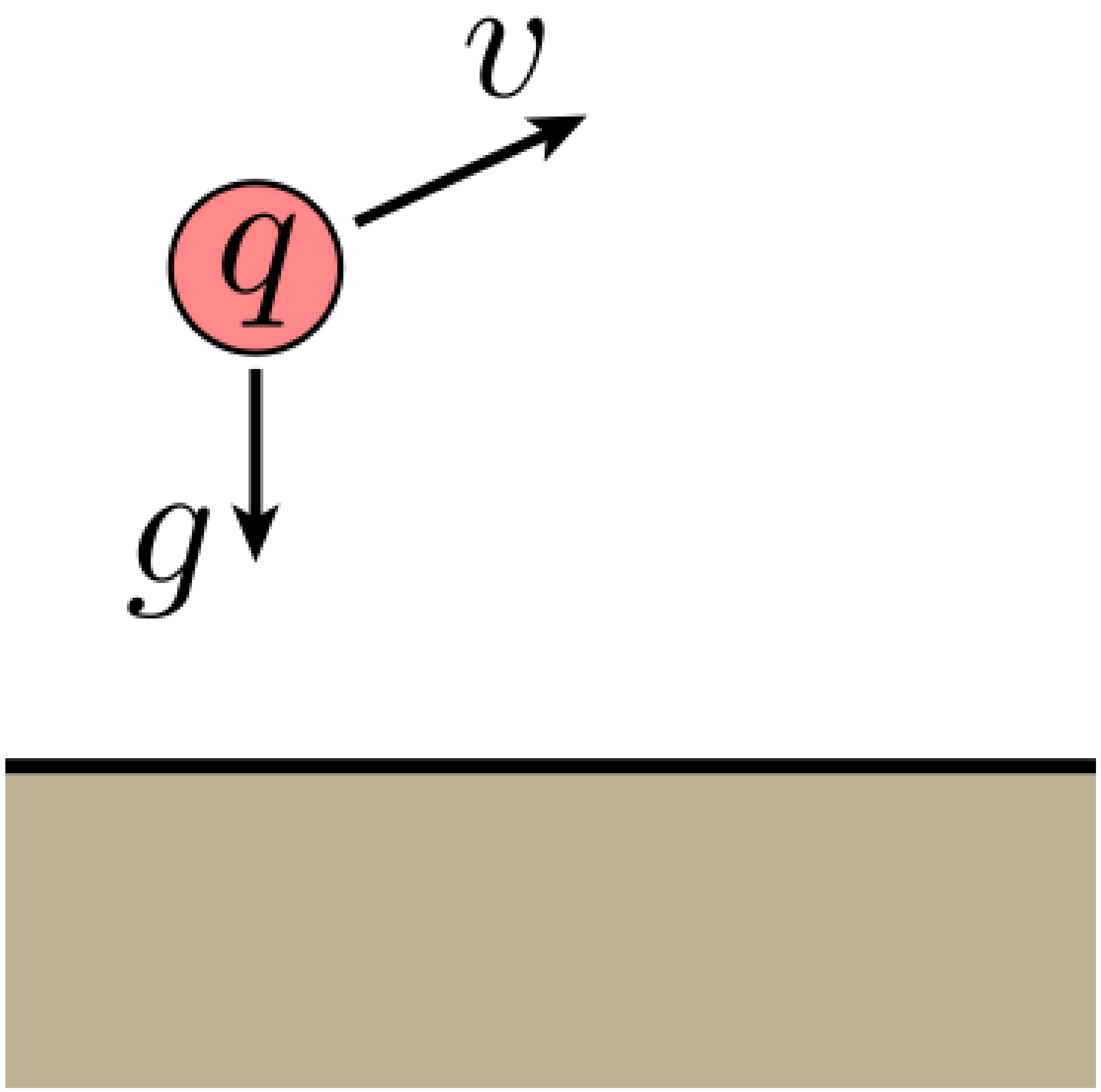
(What do we need a computer for?!)

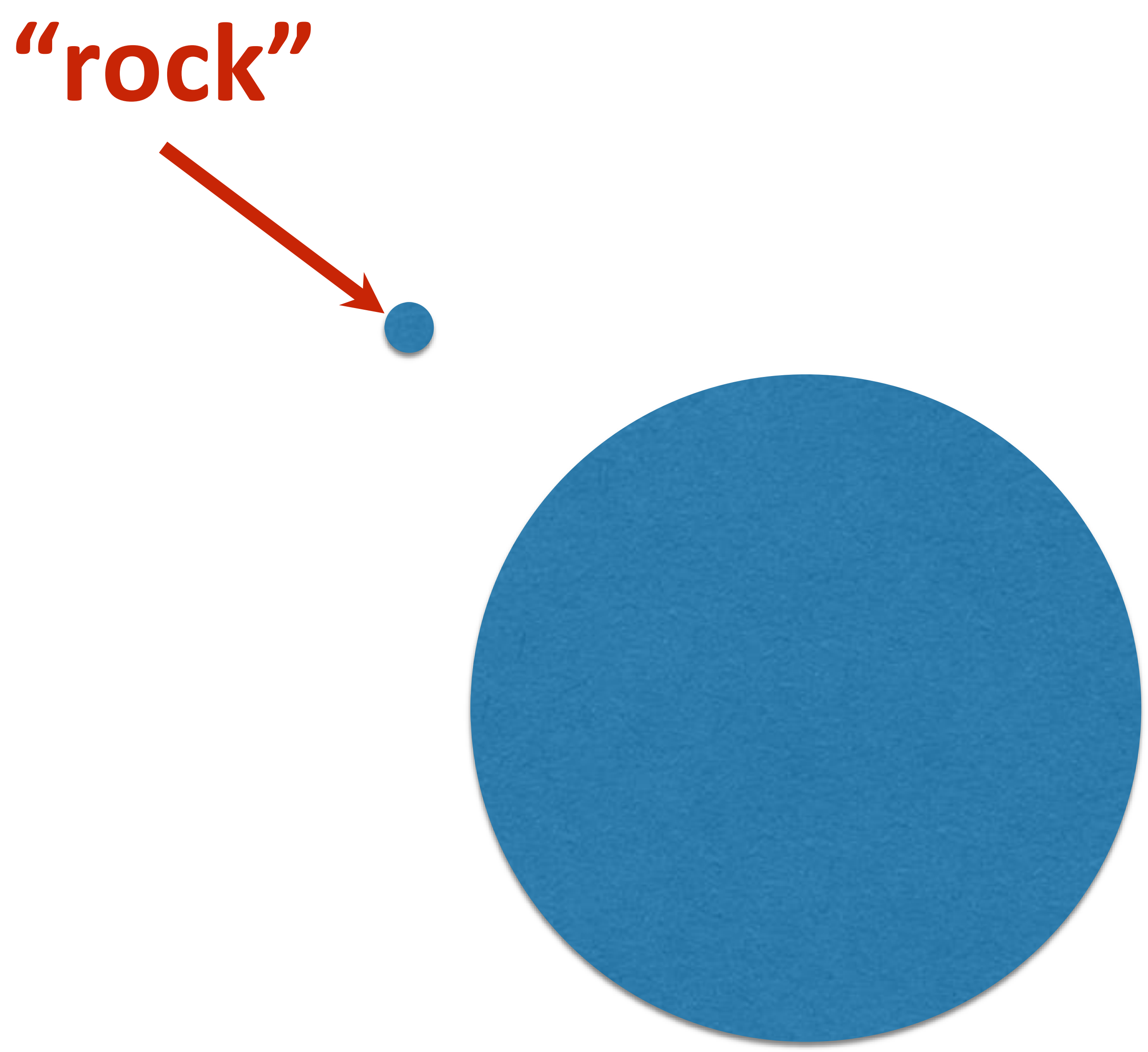
*This rock is spherical and has uniform density.

Note: this is an initial value problem!

Simple Example: the two-body problem

- Let’s take a closer look…

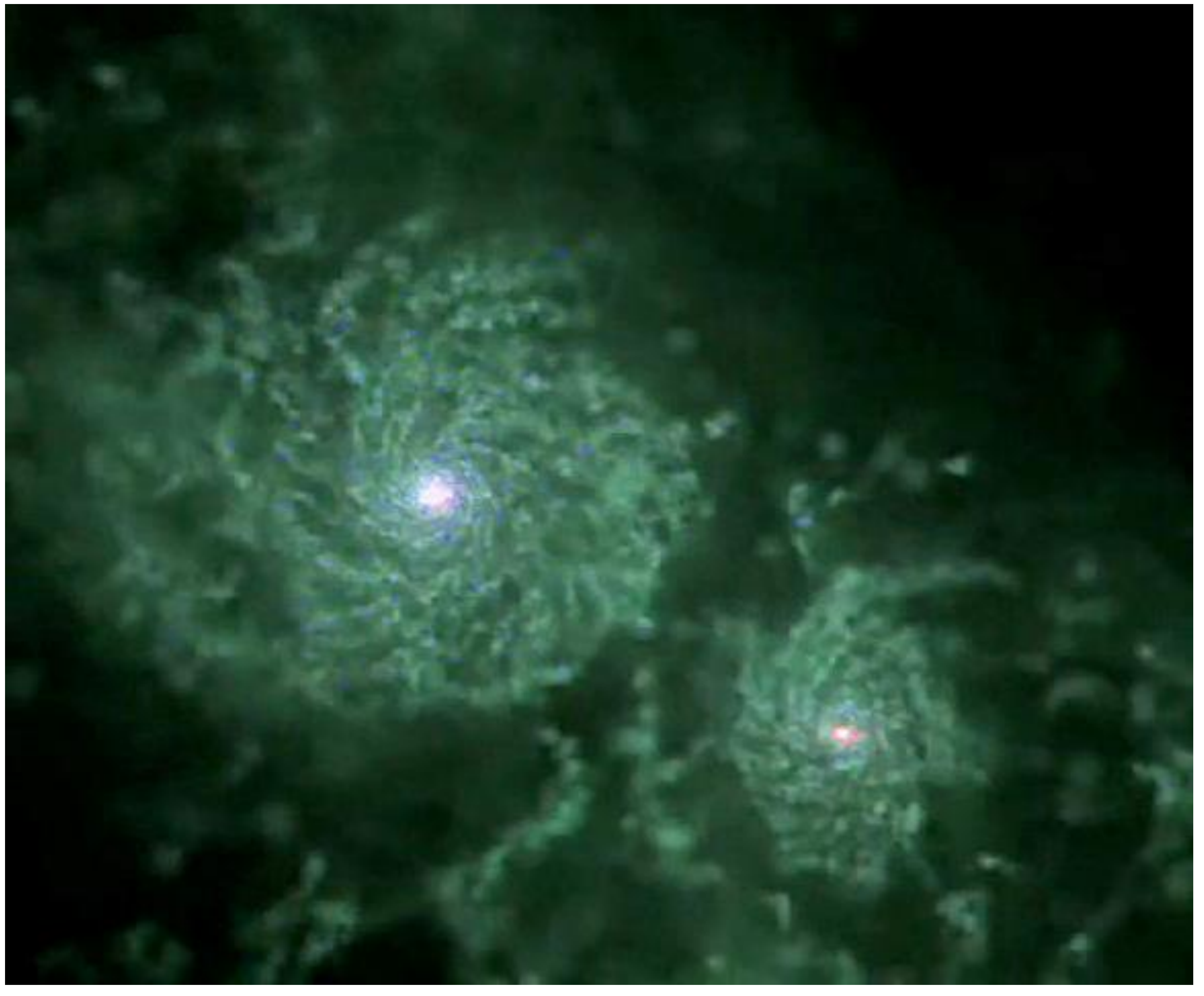




$$F_{gravity} = -GmM_0 \frac{x - x_0}{\left| x - x_0 \right|^3}$$

Not-So-Simple Example: *n*-Body Problem

- Consider the Earth, moon, and sun—where do they go?
- As soon as $n \geq 3$, no closed form (chaotic solutions)
- What if we want to simulate entire *galaxies?*

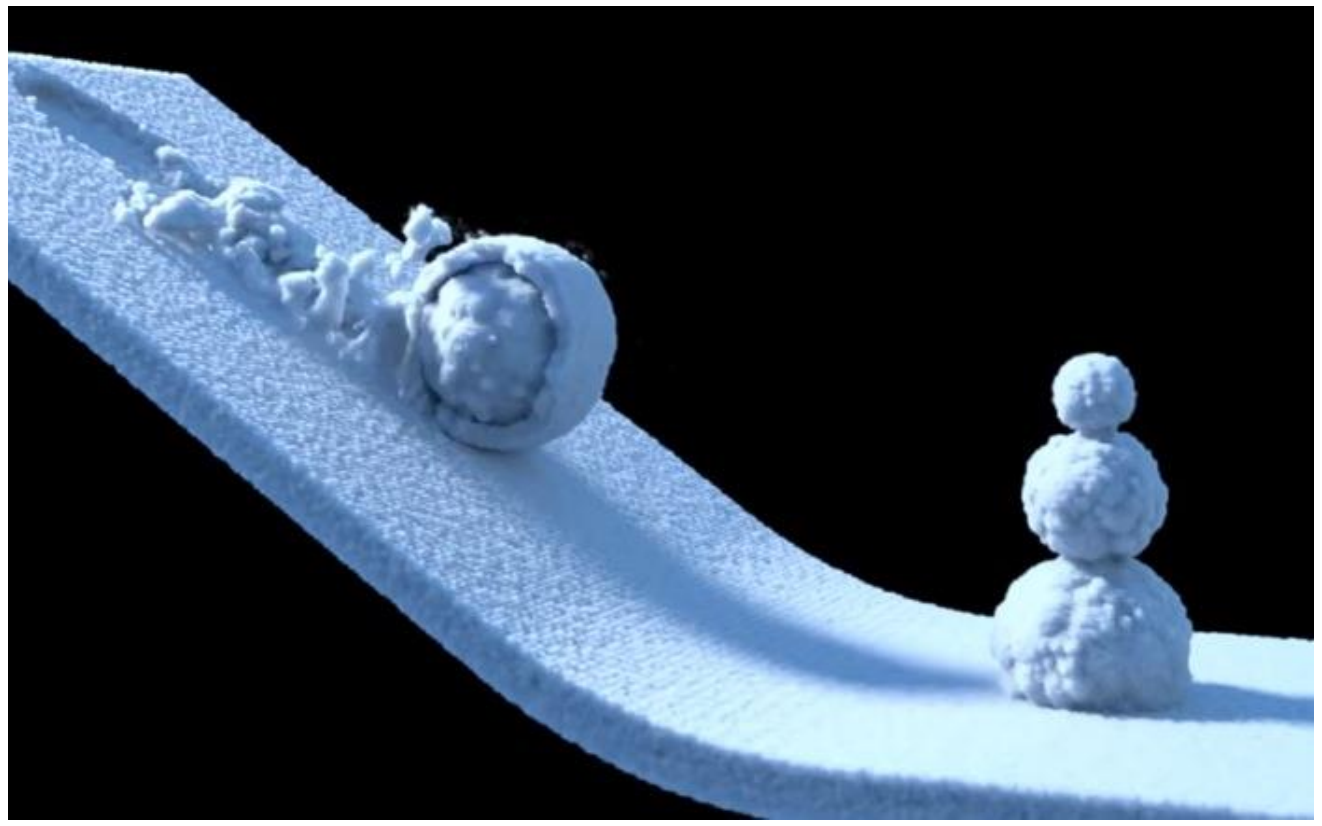


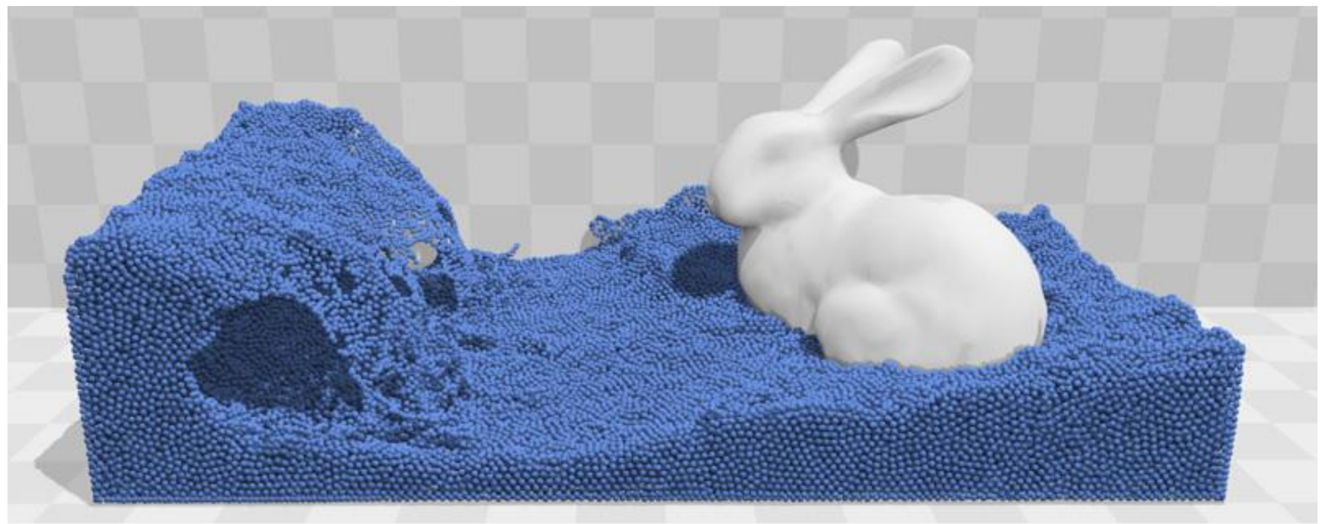
Credit: Governato et al / NASA

In computer animation, we *want* to simulate phenomena that are quite complex!

Particle Systems

- Model complex phenomena as large collection of particles
- Each particle has a behavior described by (physical or *non*-physical) forces
- Very common in graphics/games
  - easy to understand
  - simple equation for each particle
  - easy to scale up/down
- May need *many* particles to capture certain phenomena (e.g., fluids)
  - may require fast hierarchical data structure (kd-tree, BVH, ...)
  - sometimes better to use continuum model!

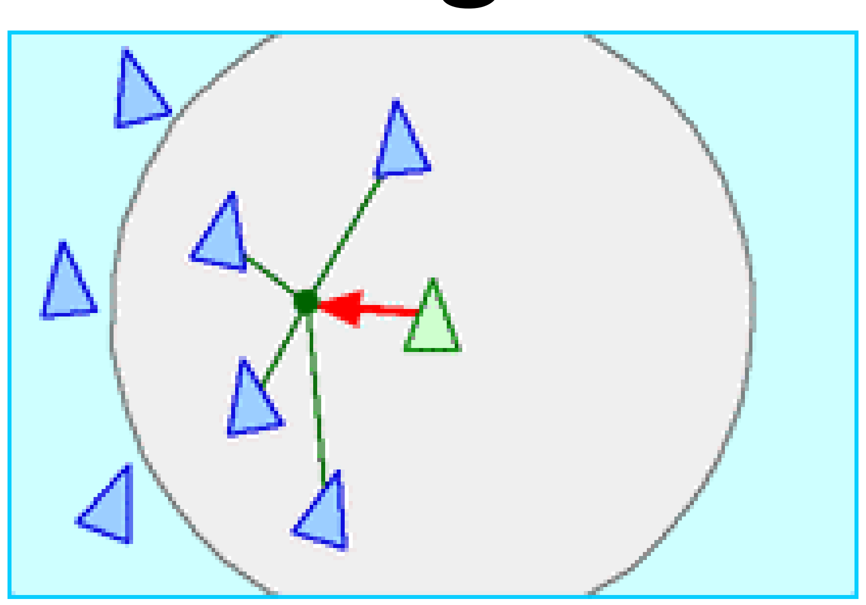




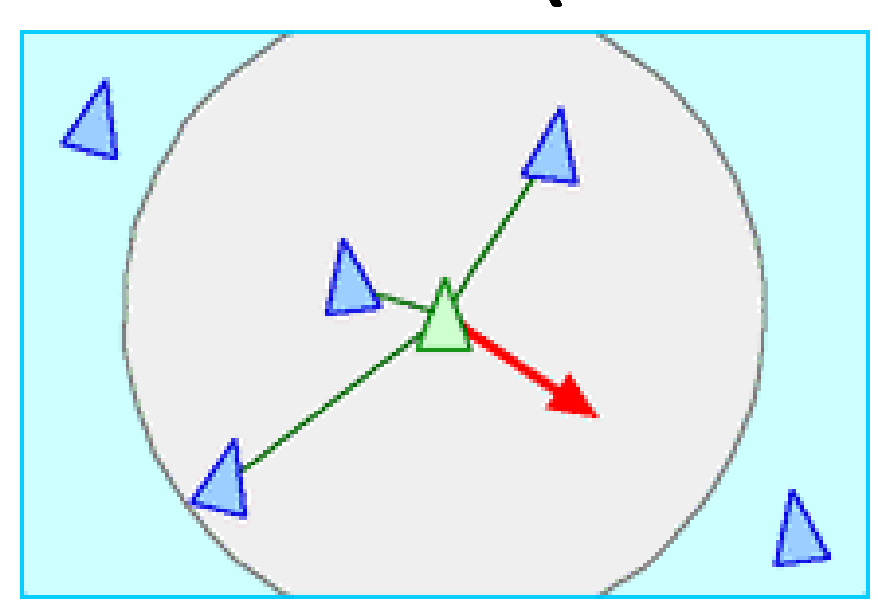
Example: Flocking

Simulated Flocking as an ODE

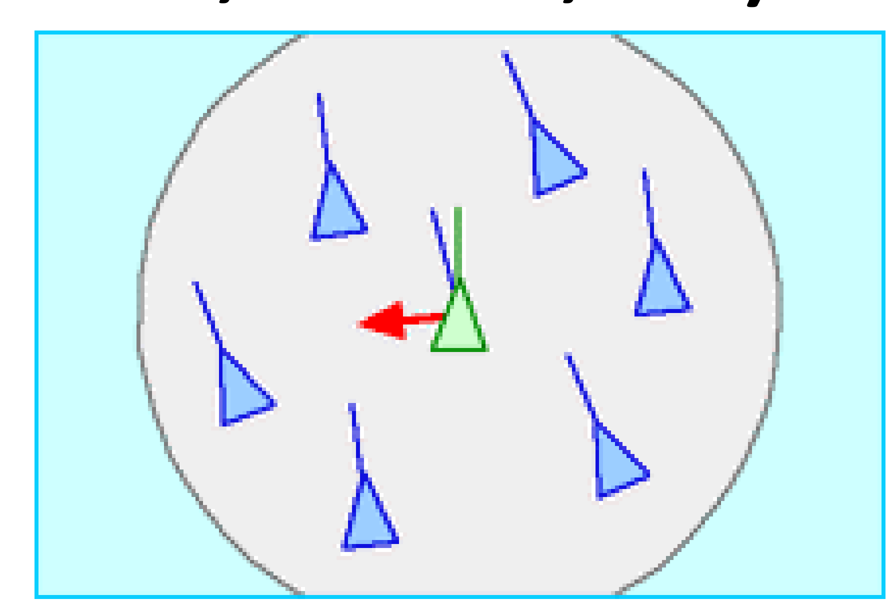
- Each bird is a particle
- Subject to very simple forces:
  - *attraction* to center of neighbors
  - *repulsion* from individual neighbors
  - *alignment* toward average trajectory of neighbors
- Solve large system of ODEs (numerically!)
- Emergent complex behavior (also seen in fish, bees, ...)



attraction



repulsion



alignment

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

Example: Crowds

Where are the bottlenecks in a building plan?

Example: Granular Materials

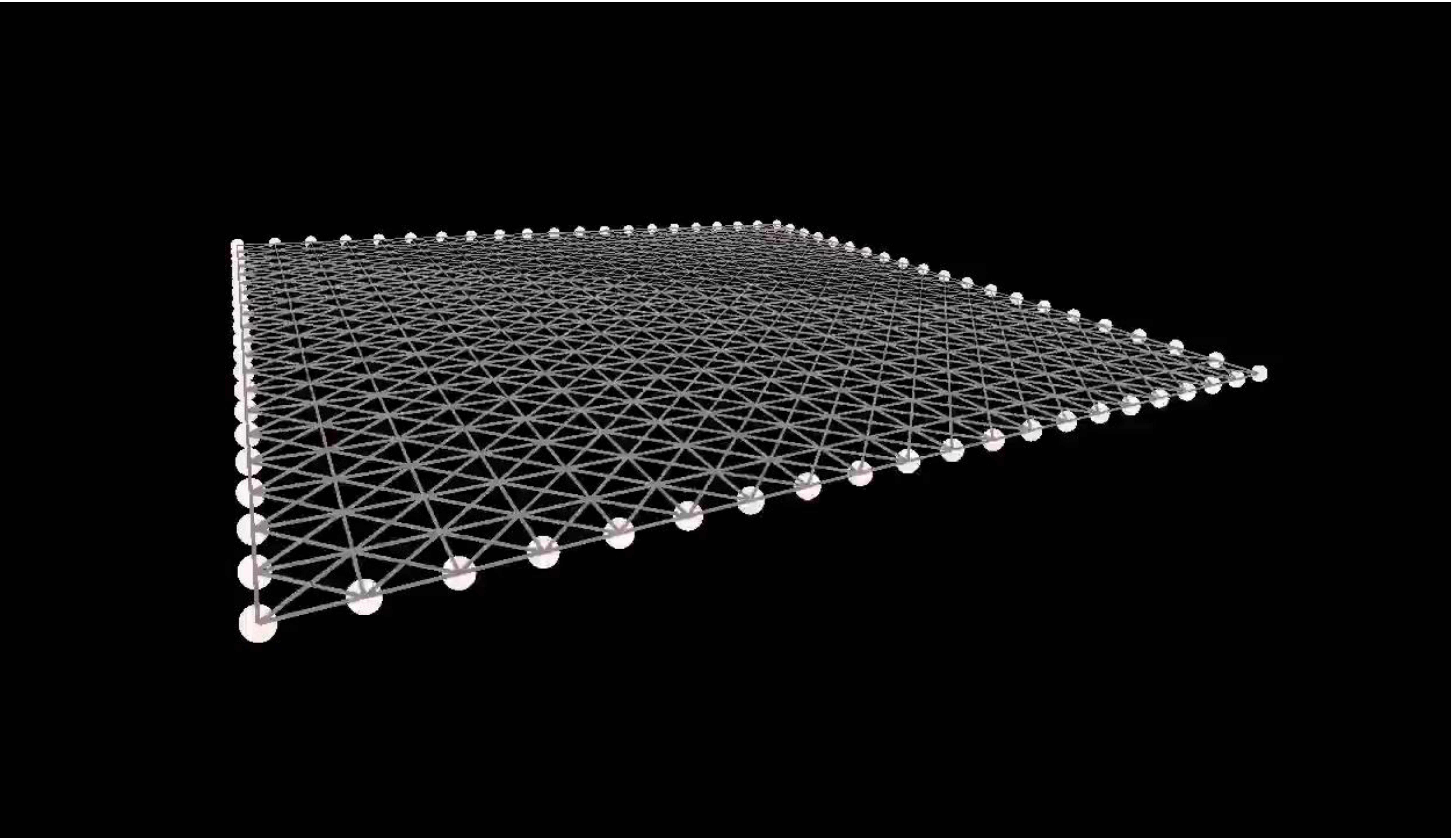
Bell et al, “Particle-Based Simulation of Granular Materials”

Example: Particle-Based Fluids

Movie *Battleship*

(Fluid: particles or continuum?)

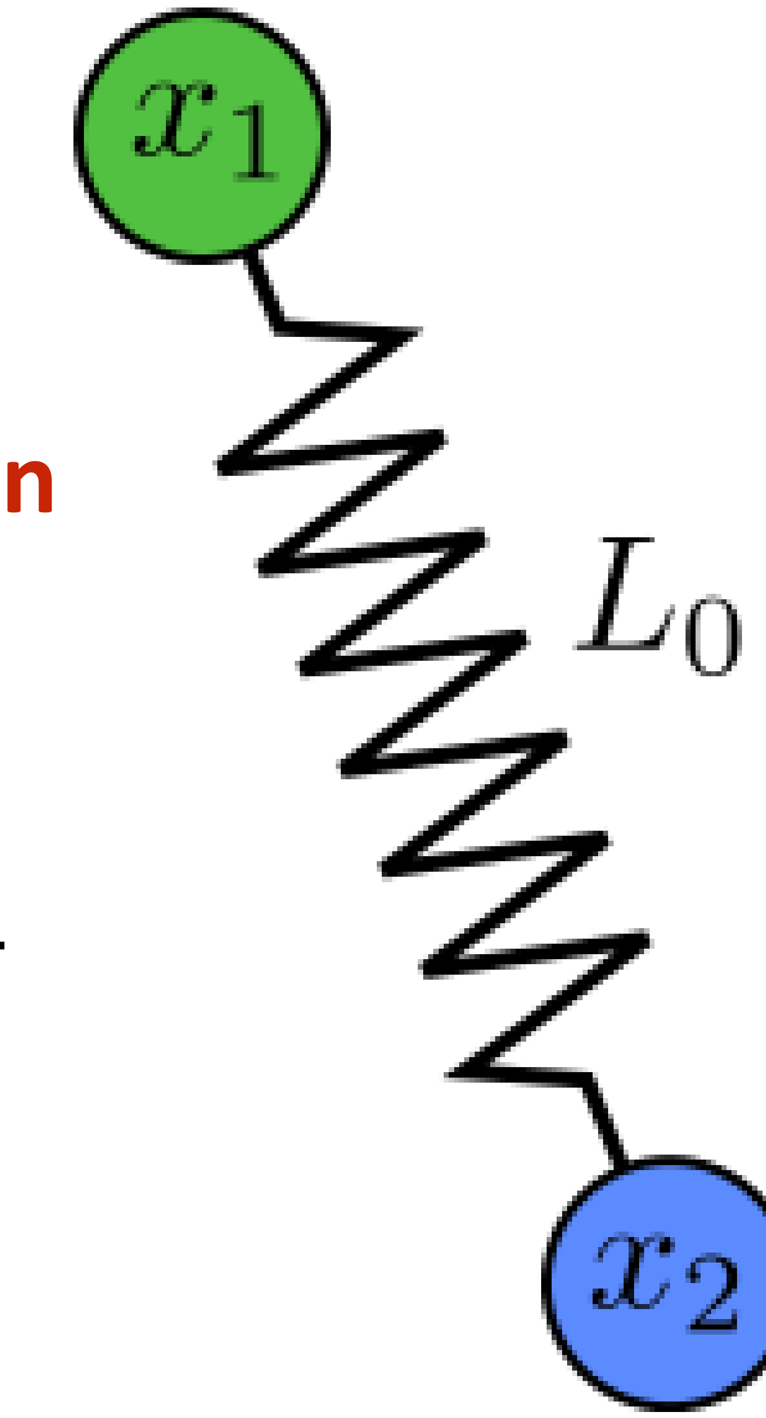
Example: Mass Spring System



Example: Mass-Spring System

- Connect particles x_1, x_2 by a spring of length L_0
- Spring force is given by Hooke's law:

  (*ceiiinosssttuu, or Ut tensio, sic vis.*)



current length direction

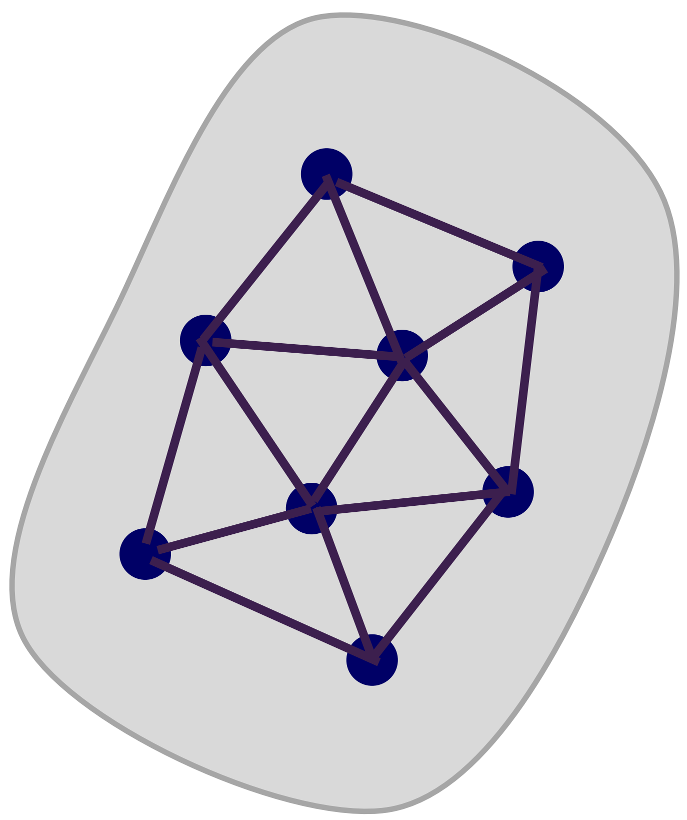
stiffness

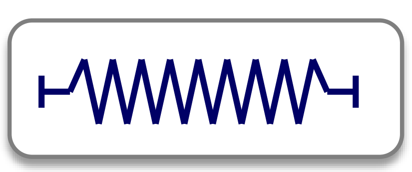
$$F_{spring} = -K\left(\frac{|x_1 - x_2|}{L_0} - 1\right)\frac{x_1 - x_2}{|x_1 - x_2|}$$

rest length

- Very common in graphics/games
  - easy to understand
  - simple force equation
  - Easy to combine springs + particles to model complex phenomena!

Example: Mass-Spring System





Spatial discretization: sample object with mass points

- Total mass of object: M
- Number of mass points: p
- Mass of each point: $m=M/p$ (*uniform distribution*)

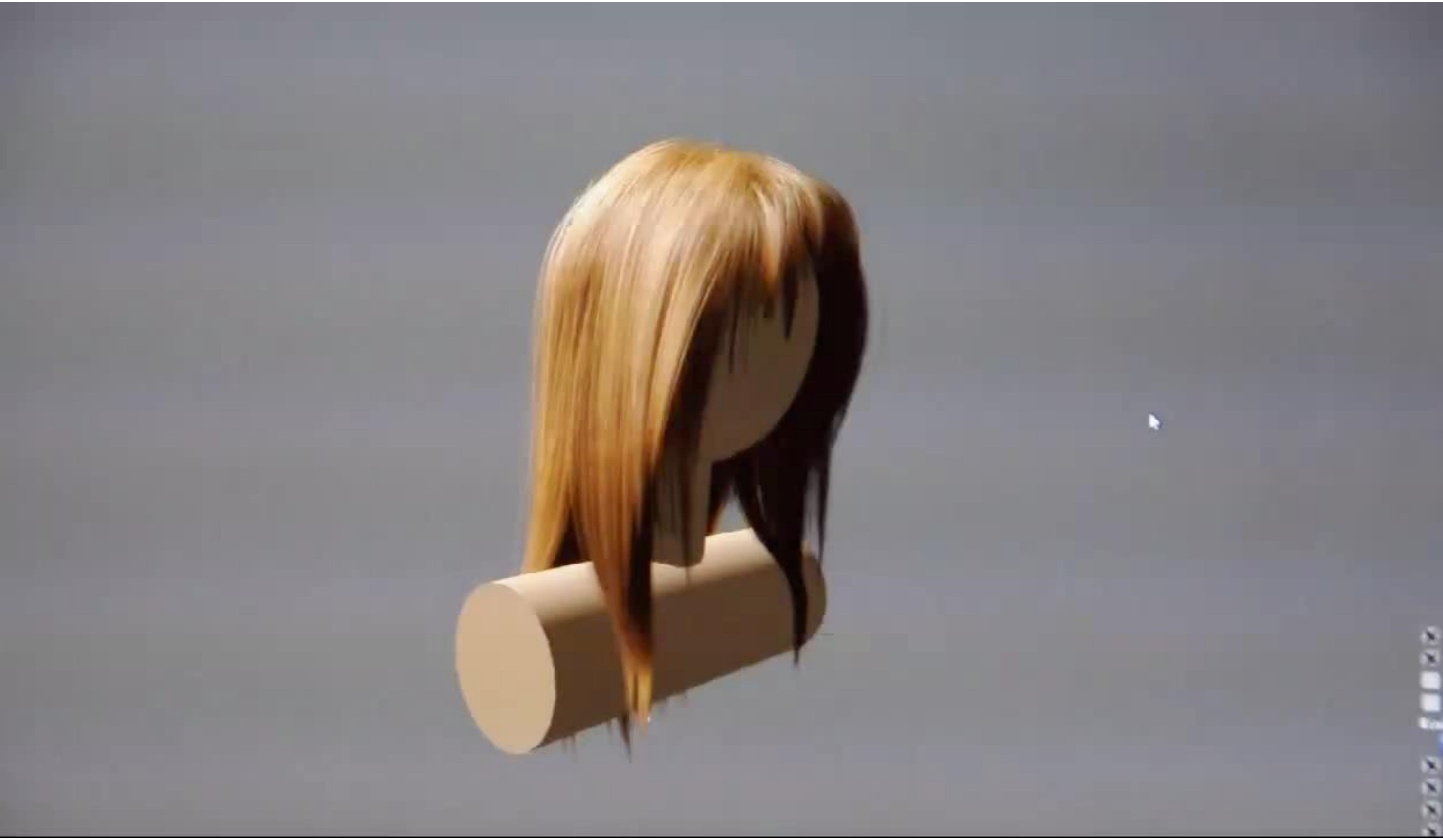
Each point is a particle, just like before. It has

- Mass
- Position
- Velocity

Connect particles with springs, evaluate force due to each spring, add gravity, etc and integrate.

Example: Mass Spring + Character

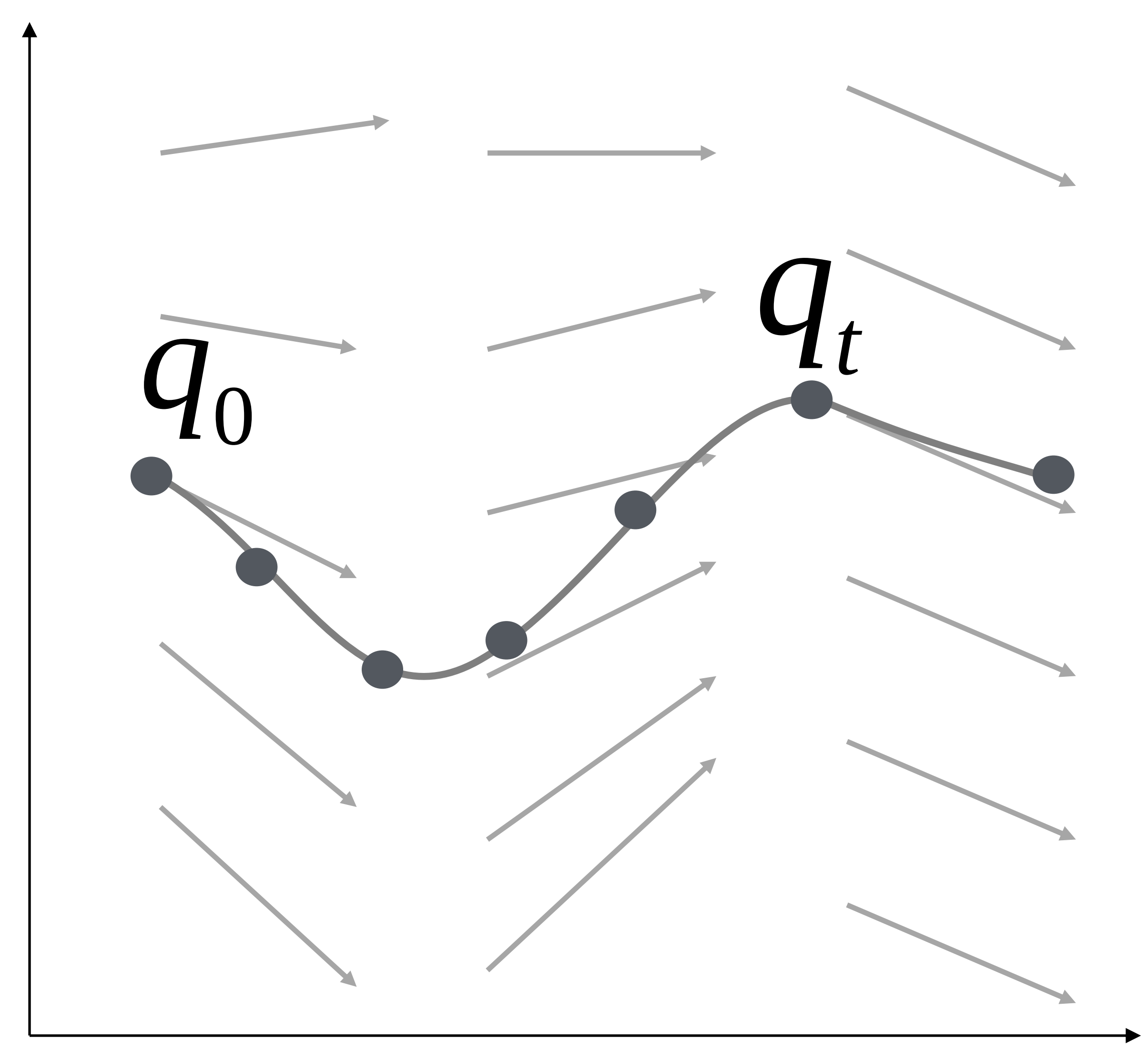
Example: Hair



How do we solve these ODEs numerically?

Numerical Integration

- Given initial conditions $q(0), \dot{q}(0)$, find function $q(t)$
- Replace time-continuous function $q(t)$ with discrete samples q_i at time t_i



Numerical Integration

- How do you compute time-discretized samples q_i?

  → by solving the ODE *numerically*

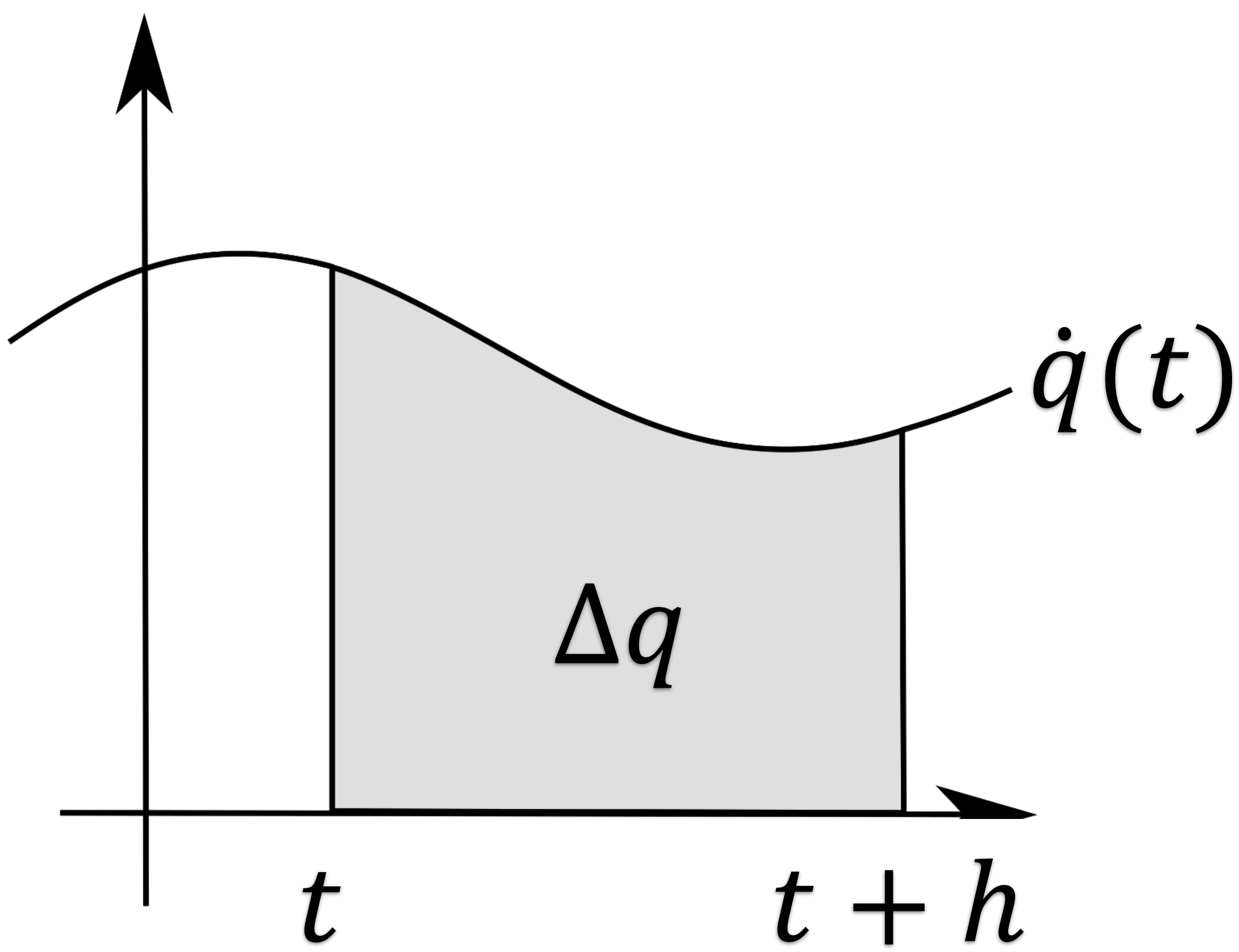
- Solving ODEs numerically → numerical time integration

$$q(t+h) = q(t) + \int_t^{t+h} \dot{q}(t)\, dt$$

- How do we solve this integral numerically?

  → by using numerical integration rules

Numerical Integration



Continuous problem:

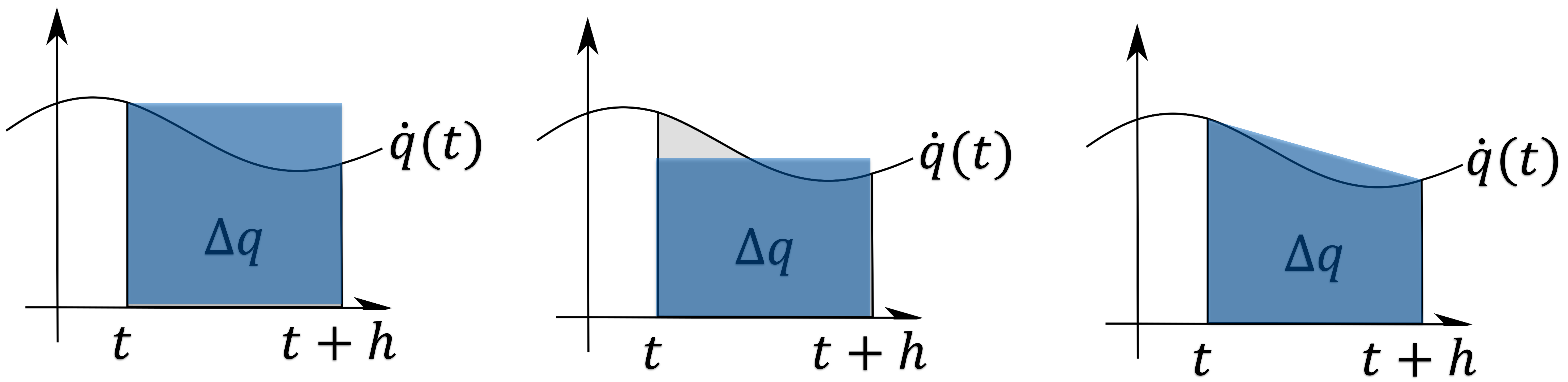
$$q(t+h) = q(t) + \int_t^{t+h} \dot{q}(t)\, dt$$

Discrete approximation:

$$q_{i+1} = q_i + \Delta q_i$$

$$\Delta q_i \approx \int_t^{t+h} \dot{q}(t)\, dt$$

Numerical Integration Rules

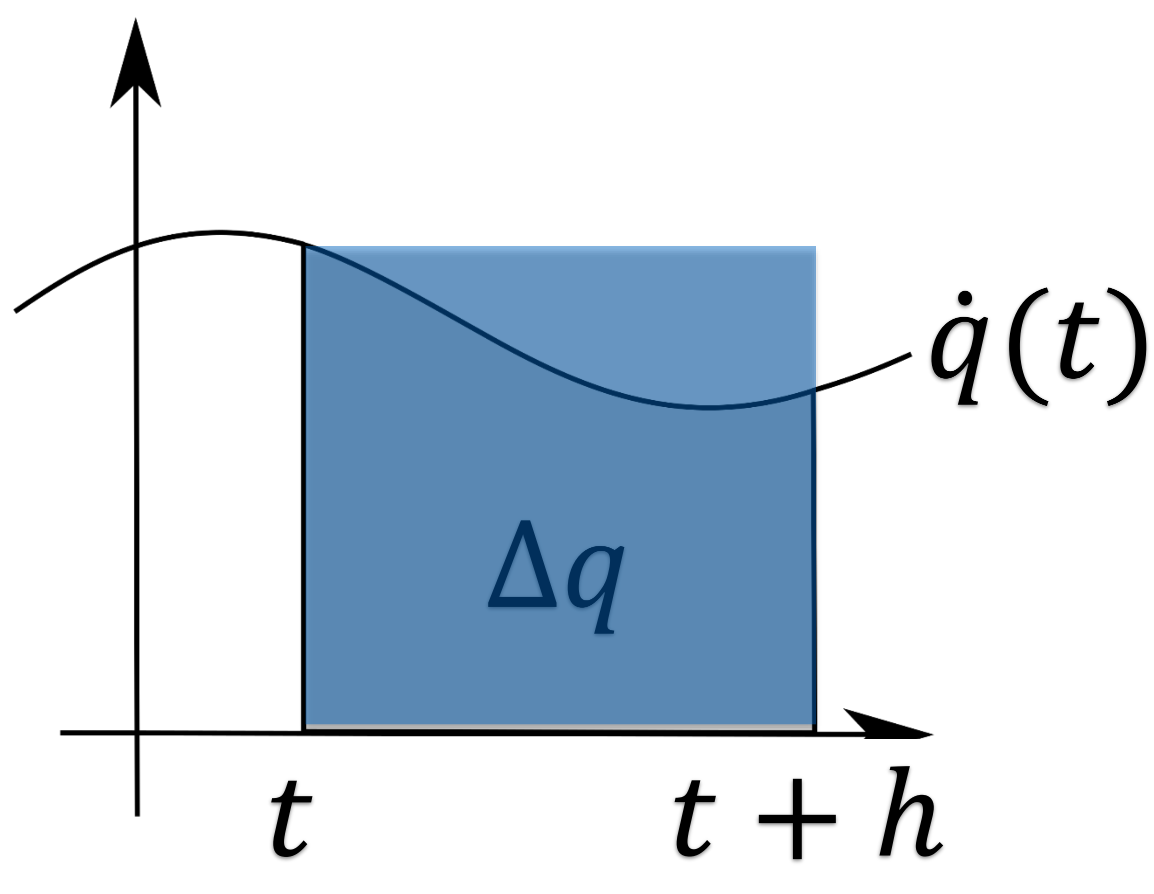


Rectangle rule	**Midpoint rule**	**Trapezoid rule**
$\Delta q_i \approx \dot{q}(t) \cdot h$	$\Delta q_i \approx \dot{q}(t + h/2) \cdot h$	$\Delta q_i \approx \frac{\dot{q}(t) + \dot{q}(t+h)}{2}$

Configuration update (rectangle rule):

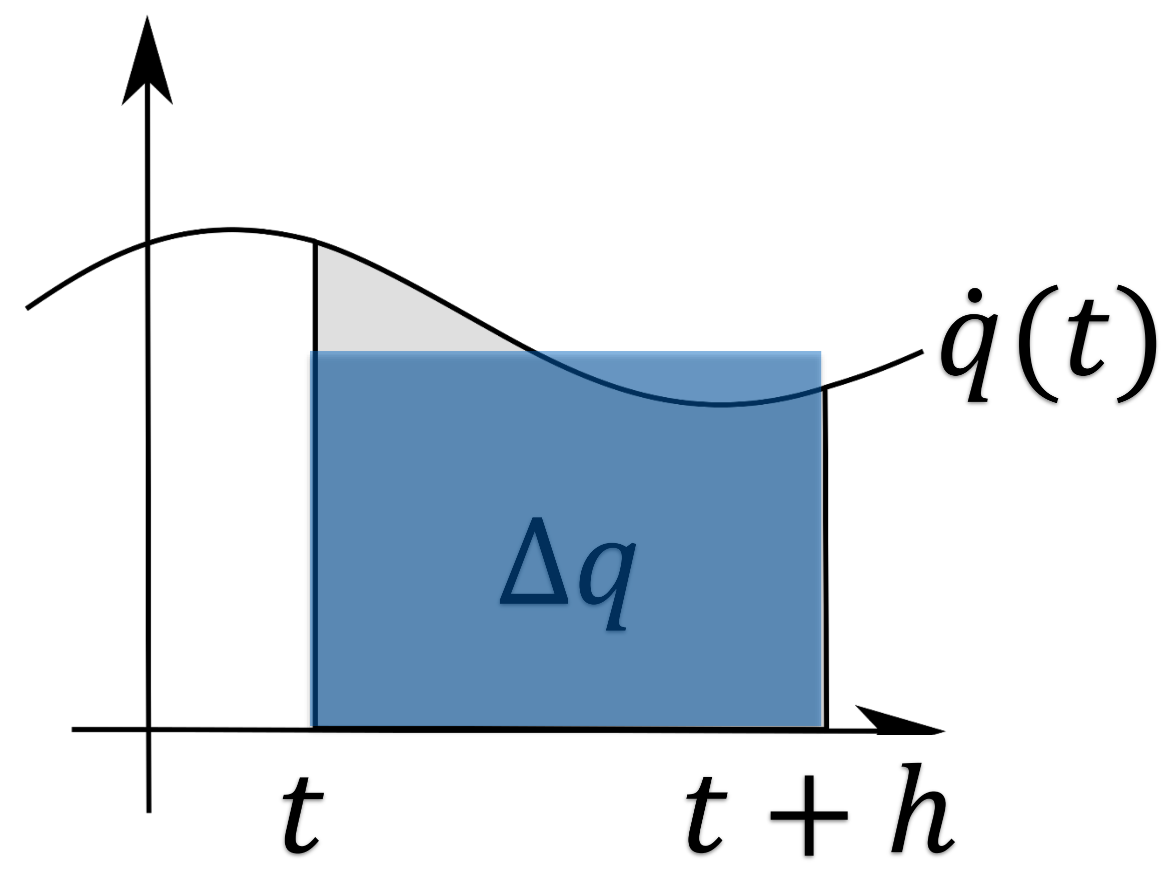
$$q_{i+1} = q_i + h \cdot \dot{q}_i$$

Numerical Integration Rules



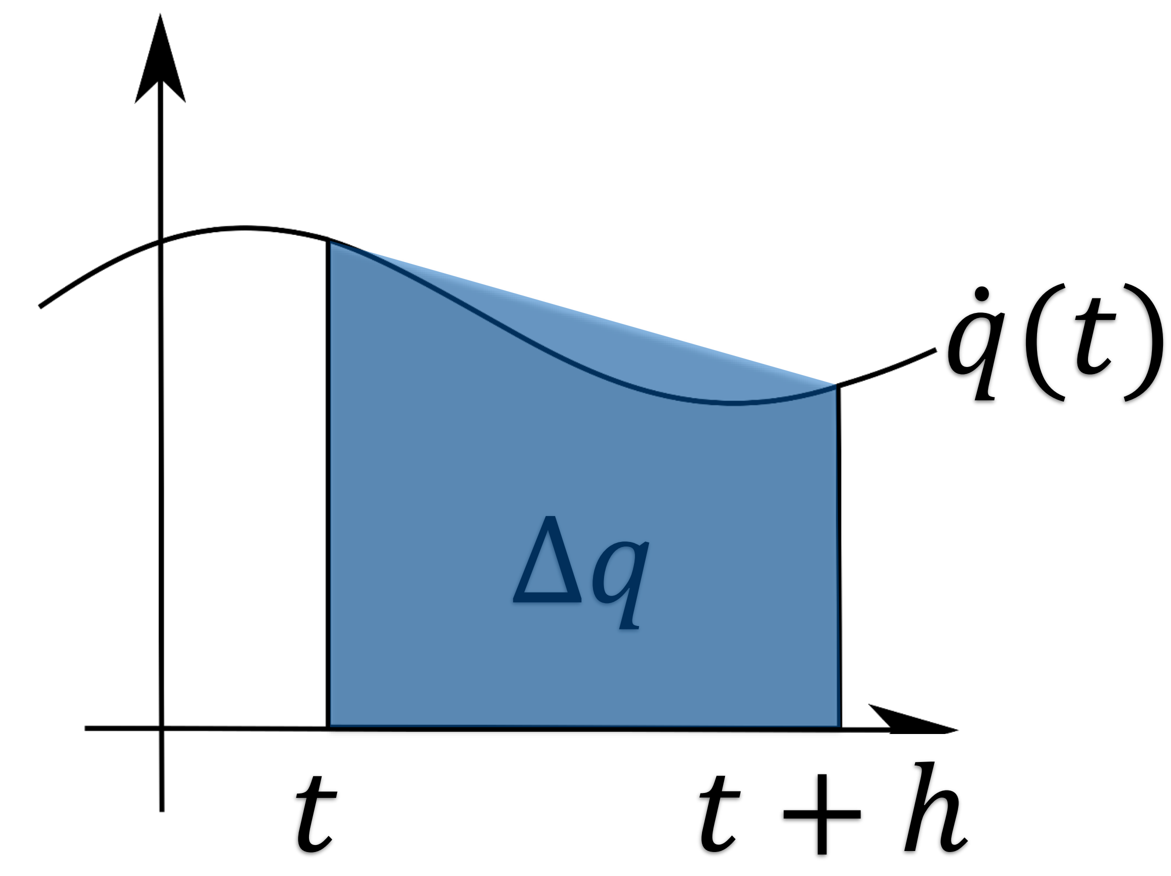
Rectangle rule

$$\Delta q_i \approx \dot{q}(t) \cdot h$$



Midpoint rule

$$\Delta q_i \approx \dot{q}(t + h/2) \cdot h$$



Trapezoid rule

$$\Delta q_i \approx \frac{\dot{q}(t) + \dot{q}(t + h)}{2}$$

- Integration schemes differ in terms of
  - accuracy/approximation order
  - number/location of function evaluations
  - ...

A Different Point of View

- Taylor series expansion

$$q(t+h) = q(t) + h\frac{dq(t)}{dt} + \frac{h^2}{2}\frac{d^2q(t)}{dt^2} + \cdots = \sum_n^\infty \frac{1}{n!}\frac{d^nq(t)}{dt^n}$$

- Truncation yields arbitrary-order approximation

$$q(t+h) = q(t) + h\frac{dq(t)}{dt} + O(h^2)$$

- Gives rise to discrete update rule

$$q_{i+1} = q_i + h \cdot \dot{q}_i \qquad *$$

- Also known as *Explicit* or *Forward Euler* scheme

*Note that this expression is identical to the one obtained by applying the *rectangle rule* from previous slide

Forward Euler

- Simplest scheme: evaluate derivative at current configuration
- New state can then be written *explicitly* in terms of known data:

current configuration

new configuration **velocity at current time**

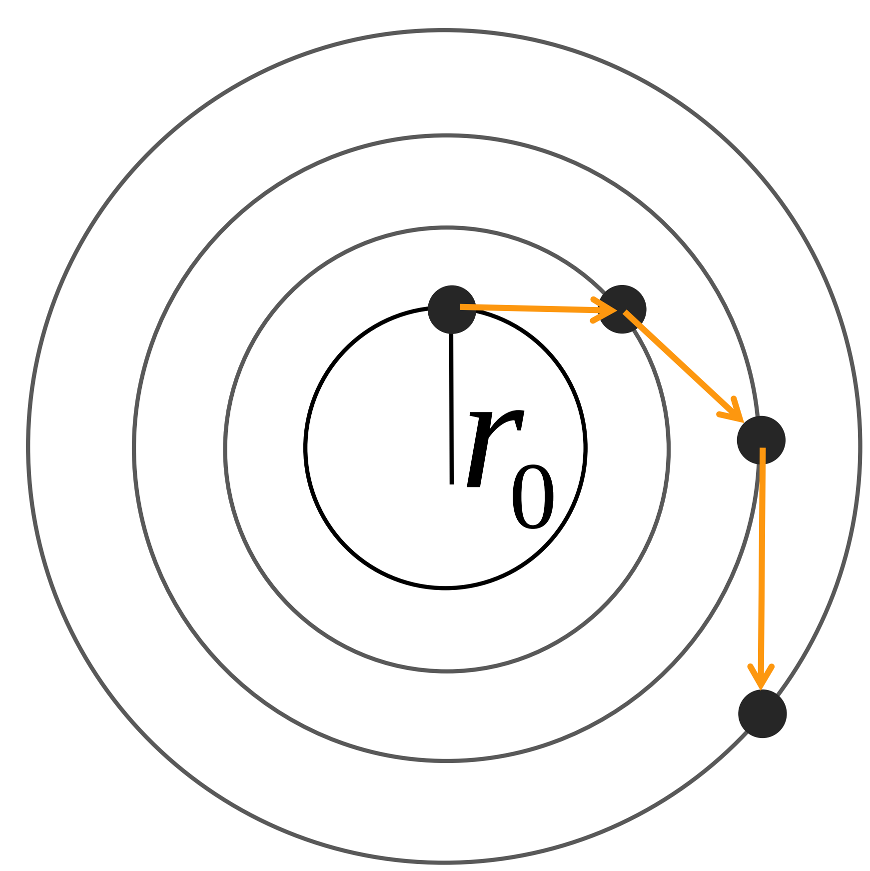
$$q_{i+1} = q_i + h \cdot \dot{q}_i$$

velocity at current time

$$\dot{q}_{i+1} = q_i + h \cdot \ddot{q}_i = q_i + hM^{-1}F(q_i, \dot{q}_i)$$

- Simple and intuitive: walk a bit in the direction of the derivative
- Unfortunately, not very *stable* ☹
- Consider the following 2D, first order ODE (what does it do?):

$$\dot{q} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} q$$



Forward Euler - Stability Analysis

- Consider the behavior of forward Euler for simple linear ODE (e.g. temperature of an object):

$$\dot{u} = -au, \quad a > 0$$

- Exact solution is $u(t) = u(0)e^{-at}$, so $u_k \to 0$ as $k \to \infty$
- Forward Euler approximation is

$$\begin{aligned} u_{k+1} &= u_k - \tau a u_k \\ &= (1 - \tau a) u_k \end{aligned}$$

- Which means after n steps, we have

$$u_n = (1 - \tau a)^n u_0$$

- **Decays only if $|1-\tau a| < 1$, or equivalently, if $\tau < 2/a$**
- In practice: may need *very small* time steps ("stiff ODE")

Backward Euler

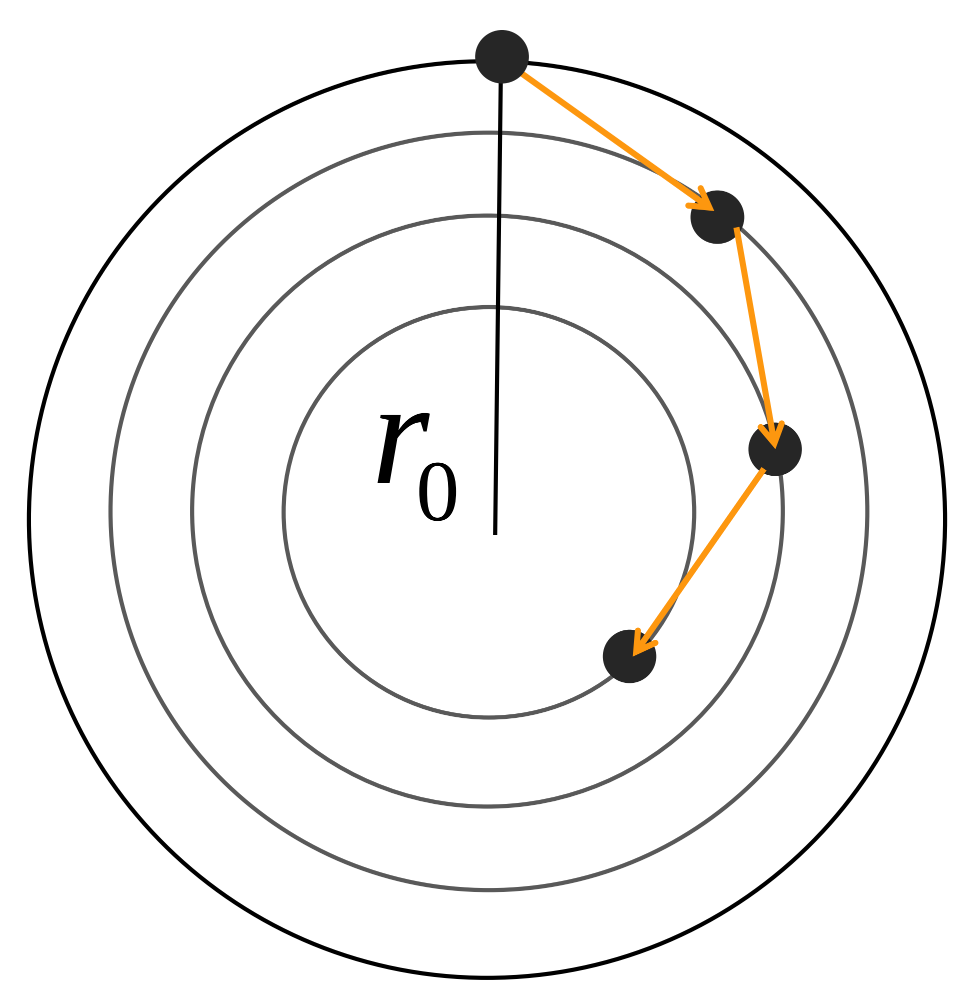
- Let's try something else: evaluate velocity at *new* configuration
- New configuration is then *implicit*, and we must solve for it:

new configuration　　current configuration　　velocity at next time

$$q_{k+1} = q_k + \tau f(q_{k+1})$$

- f is generally nonlinear, solve nonlinear equations

$$\dot{q} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} q$$



Backward Euler - Stability Analysis

- Again consider the simple linear ODE:

$$\dot{u} = -au, \quad a > 0$$

- Remember: u_k should *decay*
- Backward Euler approximation is

$$\begin{aligned} u_{k+1} &= u_k - \tau a u_{k+1} \\ (1 + \tau a) u_{k+1} &= u_k \\ u_{k+1} &= (1 + \tau a)^{-1} u_k \end{aligned}$$

- Which means after n steps, we have

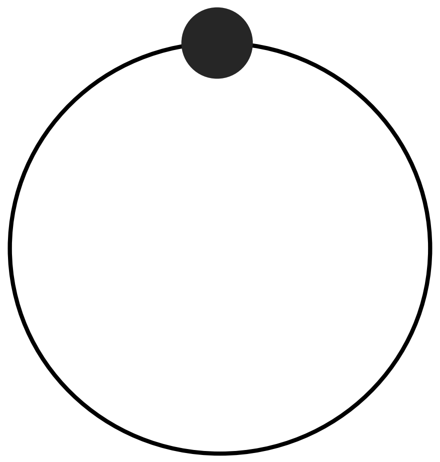
$$u_n = \left(\frac{1}{1+\tau a}\right)^n u_0$$

- **Decays if |1+τa| > 1, which is always true!**
- **⇒ Backward Euler is *unconditionally stable* for linear ODEs**

Symplectic Euler

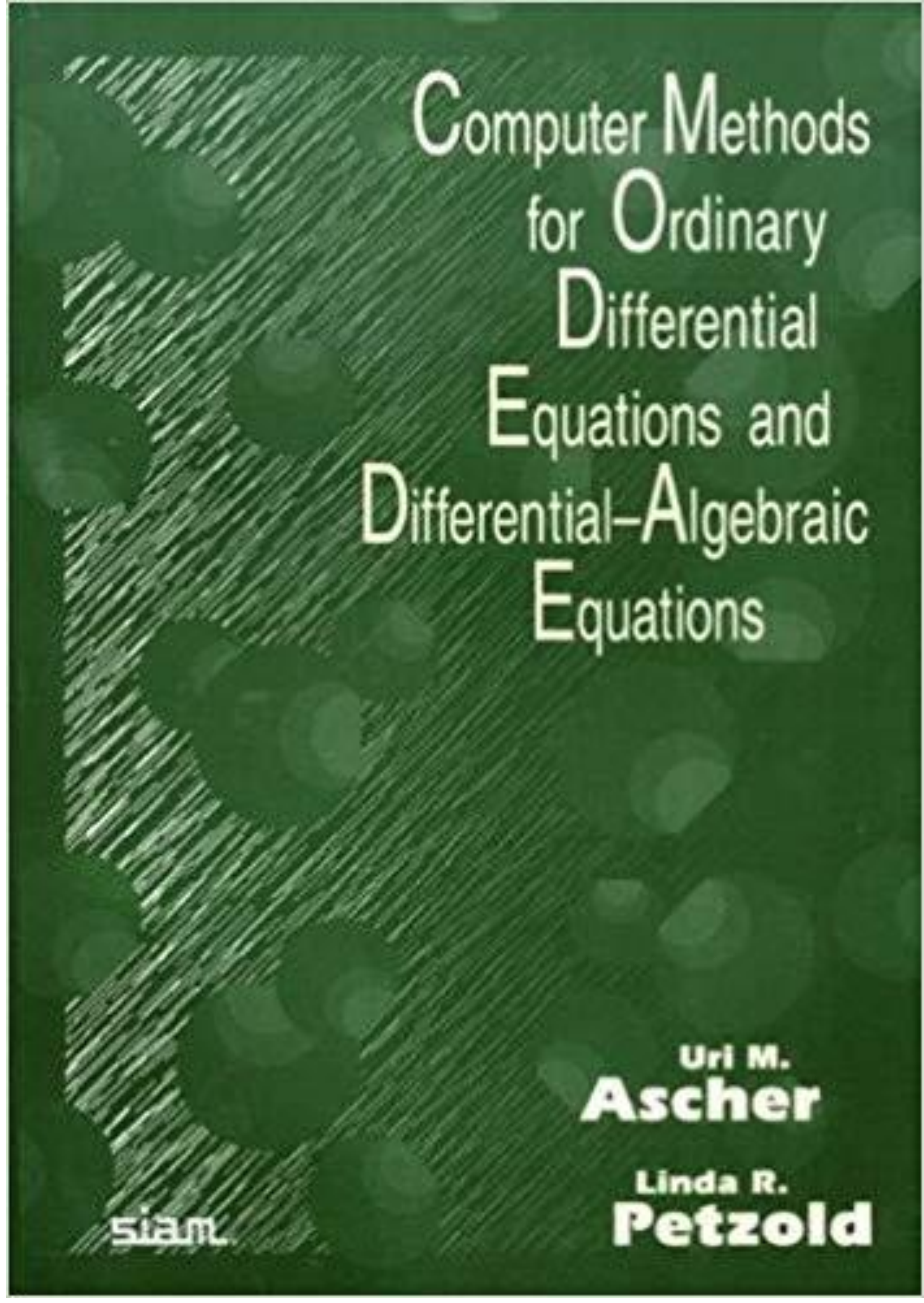
- Backward Euler was stable, but we also saw (empirically) that it exhibits *numerical damping* (damping not found in original eqn.)
- Nice alternative is *Symplectic Euler* (for 2nd order ODEs)
  - update velocity using current configuration
  - update configuration using *new* velocity

  - Easy to implement
  - Energy is conserved *almost exactly*, forever. Is that desirable?



(Proof? The analysis is not so easy...)

Numerical Integrators for ODEs



- Barely scratched the surface here
- *Many* different integrators
- Why? Because many notions of "good":
  - stability
  - accuracy
  - consistency/convergence
  - conservation, symmetry, ...
  - computational efficiency (!)
- No one "best" integrator—*pick the right tool for the job!*
- Could do (at least) an entire course on time integration...