CS**5643**

02 Systems of particles

Steve Marschner
Cornell University
Spring 2023

Kinematics of a particle

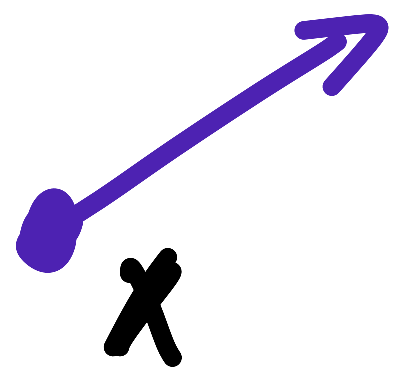
Described completely by its position $\mathbf{x} \in \mathbb{R}^d$ where $d = 2$ or 3

Particle has no extent, no orientation, etc.

- but we can model these properties later by using multiple interacting particles…

This is animation, so the particles are moving!

- We'll use the notation of dots for time derivatives
  so $\dot{\mathbf{x}}$ is the velocity of the particle
- Because velocity is important we also give it
  the name $\mathbf{v}$



$$v = \dot{x} = \frac{dx}{dt}$$

$$x = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \qquad v = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \end{bmatrix} = \begin{bmatrix} v_x(t) \\ v_y(t) \end{bmatrix}$$

Systems of particles

Usually there is more than one particle

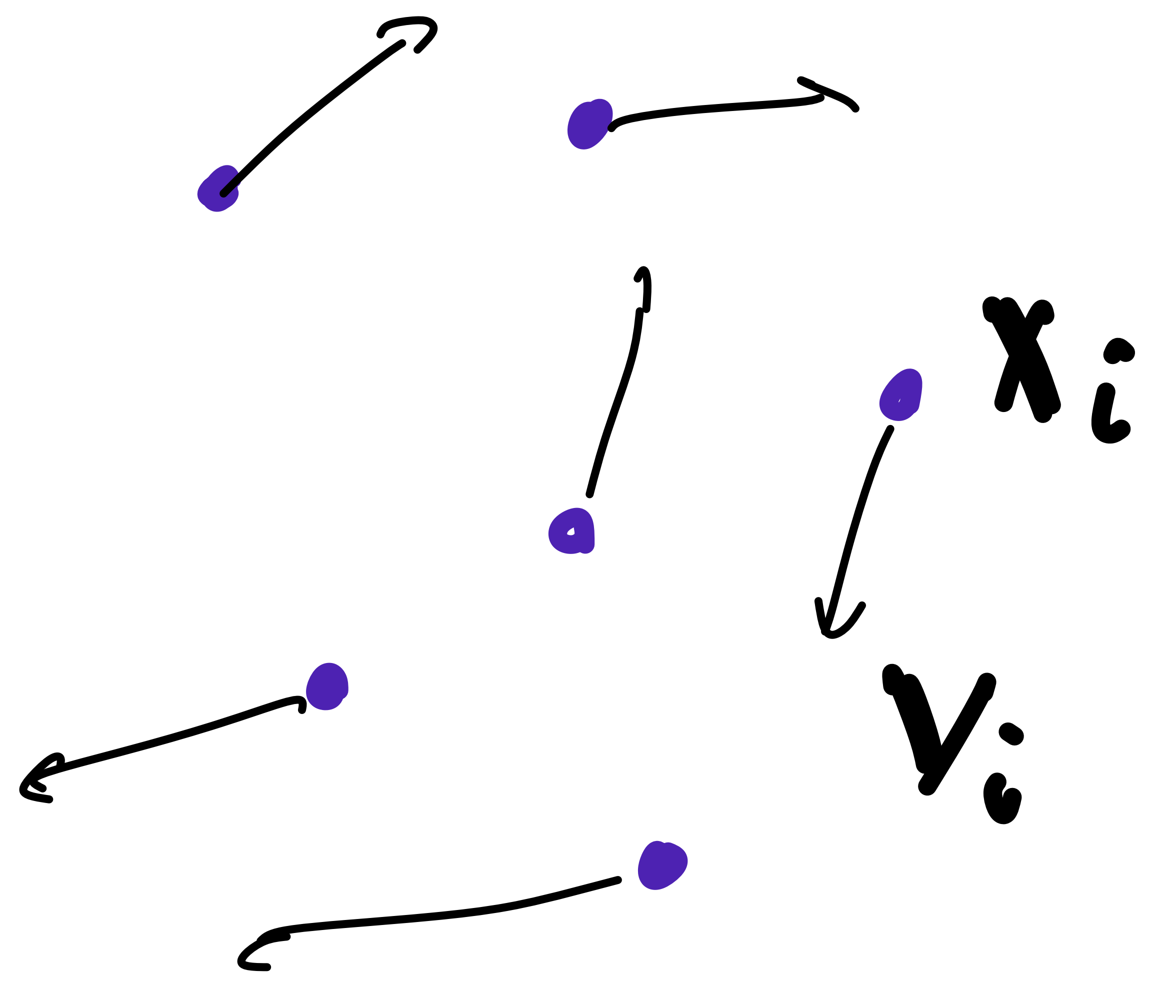
- often thousands or millions!

In this case we let $\mathbf{x}, \mathbf{v} \in \mathbb{R}^{dN}$ represent the positions and velocities of all the particles

- sometimes N changes as particles are created or deleted (caution: bookkeeping!)

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ x_1 \\ y_1 \\ z_1 \\ \vdots \\ x_N \\ y_N \\ z_N \end{bmatrix}$$

$$\mathbf{v} = \dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{bmatrix}$$



Defining particle dynamics

How to make particles go where we want?

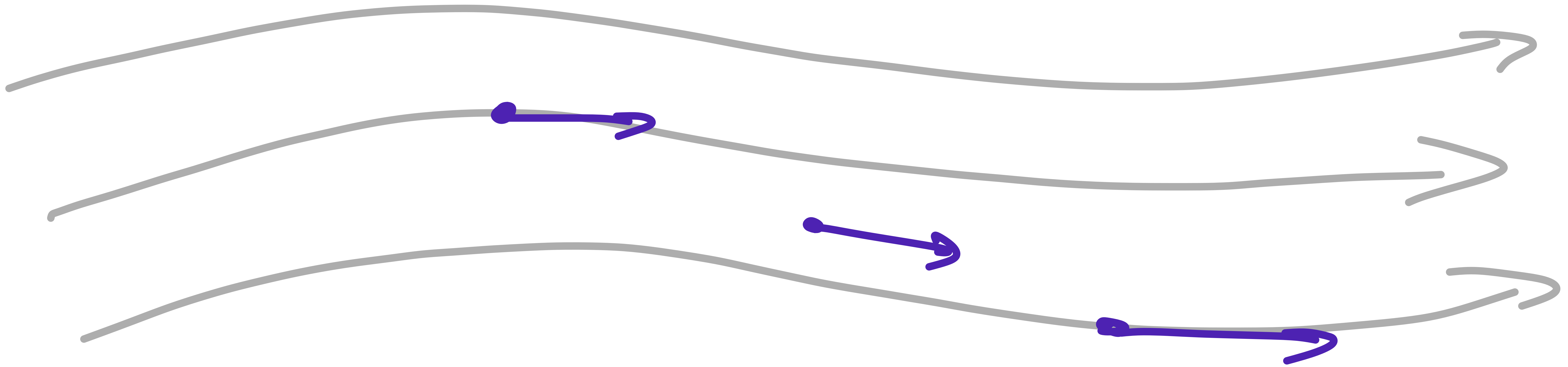
0: Script their position directly

- control $\mathbf{x}(t)$ directly
- need a function that takes a time and returns a position
- this implicitly sets $\mathbf{v}$ because it's the derivative of $\mathbf{x}$: $\mathbf{v}(t) = \dot{\mathbf{x}}(t)$
- often used to control a particle that interacts with others
  - e.g. hold up one corner of a sheet
  - e.g. let the user control one particle to push others away
  - e.g. attach one end of a rope to some game element

Defining particle dynamics

1: Set particle velocity as a function of position

- control $\mathbf{v}(t) = \mathbf{v}(t, \mathbf{x}(t))$
- need initial position, then particle path is the solution to an initial value problem
    - general form $y'(t) = f(t, y(t))$ — a first-order ordinary differential equation (ODE)
- need a function that takes a position and returns a velocity (don't need to store particle velocity)
- can do the simulation with various methods for solving ODEs



First-order dynamics: advection

particles are advected by the velocity field of a (fake) fluid flow

- no notion of particle mass or inertia here (e.g. dust or smoke particles, or fluffy snowflakes)
- for elaborate examples flow fields can come from fluid simulations
- more commonly flow fields are generated procedurally

generating plausible flow fields

- randomly generated vector fields tend to have sources and sinks
- particles will soon accumulate at the sinks — not a nice swirling motion!
- key is generating divergence free velocity fields: $\nabla \cdot \mathbf{v} = 0$
- handy fact: if your velocity field is the curl of something, it has zero divergence
- so cook up any potential $\boldsymbol{\phi}(\mathbf{x})$ you like, then define $\mathbf{v} = \nabla \times \boldsymbol{\phi}$
- demo in a bit!

TUTSBYKAI

TutsByKai https://www.youtube.com/watch?v=E8NnENBLRf8

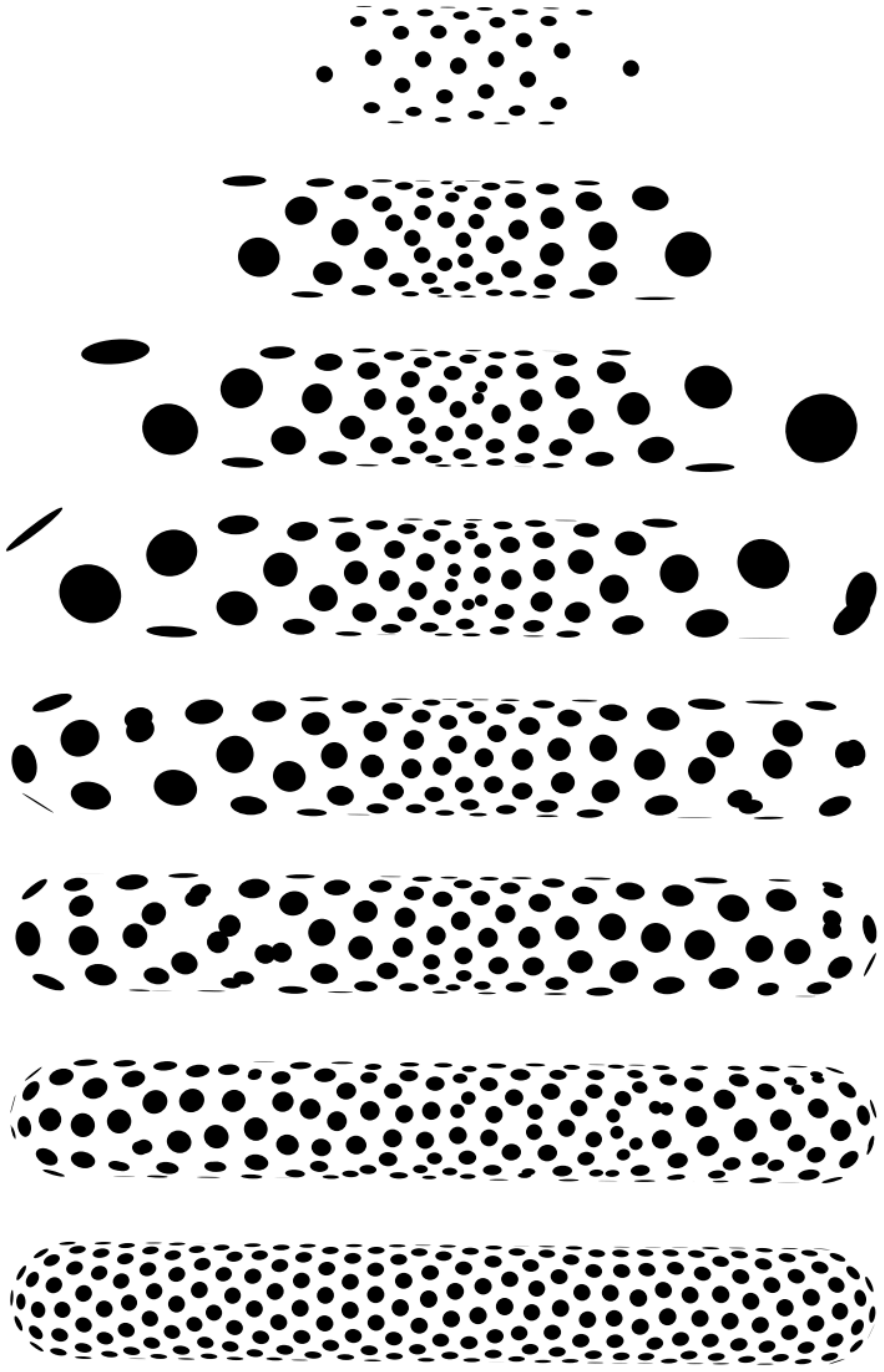
First-order dynamics: optimization

Sometimes the point of particle evolution is the end state

Use particle systems to achieve uniform spacing or other desired arrangements

In this case we can use gradient flow on a loss function:

- $\dot{\mathbf{x}}(t) = -\nabla L(\mathbf{x}(t))$
- that way the value of $L(\mathbf{x}(t))$ always decreases with time, until $\mathbf{x}$ reaches a stationary point
- note that L might involve interactions between particles so that $\dot{\mathbf{x}}_i$ might depend on various $\mathbf{x}_j$



Witkin and Heckbert, "Using Particles to Sample and Control Implicit Surfaces," SIGGRAPH 94.

Defining particle dynamics

2: Set particle acceleration as a function of position (and maybe velocity)

- control $\dot{\mathbf{v}}(t) = \mathbf{a}(t, \mathbf{x}(t), \mathbf{v}(t))$
- need initial position *and velocity* and then future path is determined
- need a function that takes system state and computes the acceleration

Newtonian mechanics leads to a second order formulation like this

- for each particle, $\mathbf{f}_i(t, \mathbf{x}_i(t), \mathbf{v}_i(t), \ldots) = m_i \ddot{\mathbf{x}}(t)$ (ellipsis because forces could depend on other particles)
- for the whole system $\mathbf{f}(t, \mathbf{x}(t), \mathbf{v}(t)) = M \ddot{\mathbf{x}}(t)$ (and for a particle system M is diagonal)

Forces for particles: unary

gravity (constant)

- $f_i = m_i \mathbf{g}$ (doesn't depend on anything at all…)

drag (depends on own velocity)

- $f_i = -k\mathbf{v}_i$ (viscous drag, a reasonable model for slow moving objects)
- $f_i = -kv_i\mathbf{v}_i = -kv_i^2\hat{\mathbf{v}}_i$, where $v = |\mathbf{v}|$ (aerodynamic drag, a reasonable model for fast objects)

anchored spring with zero rest length (depends on own position)

- $f_i = -k(\mathbf{x} - \mathbf{p})$ (spring anchored at $\mathbf{p}$)

gravitational field (depends on own position)

- $f_i = -\dfrac{GMm_i}{|\mathbf{x}|^2}\hat{\mathbf{x}} = -\dfrac{GMm_i}{|\mathbf{x}|^3}\mathbf{x}$ (for mass M fixed at origin) $f_i = -GMm_i\dfrac{\mathbf{x} - \mathbf{p}}{|\mathbf{x} - \mathbf{p}|^3}$ (M at $\mathbf{p}$)

Forces for particles: binary

spring with zero rest length

- $f_i = -k(\mathbf{x}_i - \mathbf{x}_j)$
- $f_j = -k(\mathbf{x}_j - \mathbf{x}_i) = -f_i$ (forces on i and j must be opposite, otherwise there is an external force)

gravitational force between two bodies (same form for elecrostatic)

- $f_i = -\dfrac{Gm_im_j}{|\mathbf{x}_i - \mathbf{x}_j|^2}\dfrac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|} = -Gm_im_j\dfrac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|^3}$

Fun with demos!

ballistic particles

- Newtonian particle system (second order)
- initialize with random velocity
- gravity, drag forces

springy particles

- Newtonian particle system still
- initialize with random position and velocity
- spring and gravitational forces

particle advection

- first order advection
- random vector fields using Perlin noise and curl noise
- line integral convolution for field visualization