Lec 22 - Animation Cont.

Given the forces / physics / theory, how to simulate actual movements
Euler

Single Particle Simulation

have: (for every single particle) v(t0) & x(t0)
want: x(t1)

- Later, generalize to a multitude of particles

To start, assume motion of particle determined by a velocity vector field that is a function of position and time

速度场：`v(x,t)`

Ordinary Differential Equation (ODE) 常微分方程

计算速度场内粒子的位置需要计算一阶常微分方程：

$$\frac{dx}{dt} = \dot{x} = v(x, t)$$

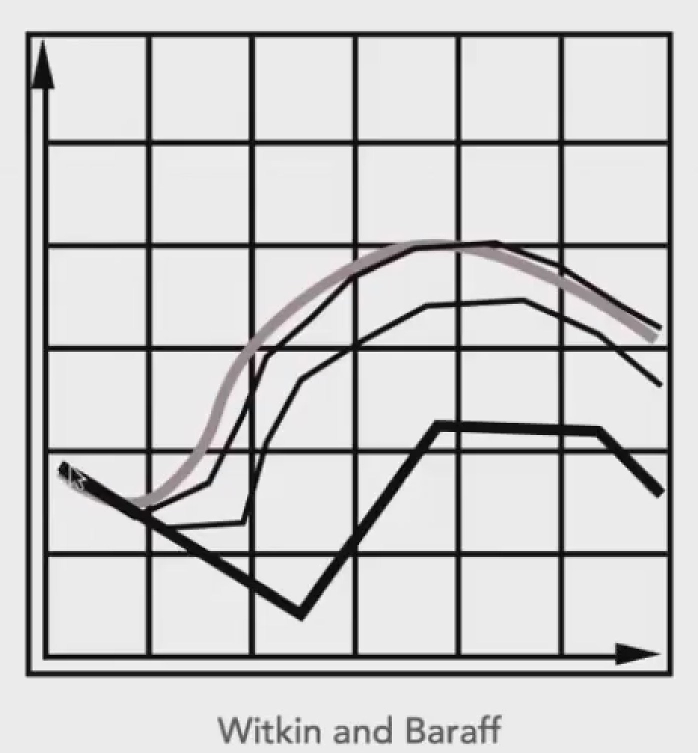
解一阶常微分方程：

- 连续：积分
- 离散：Euler’s Method欧拉方法
  - Explicit Euler method （forward 前向、显式）
    - 始终用前一帧的状态来更新后一帧

$$\begin{aligned} \boldsymbol{x}^{t+\Delta t} &= \boldsymbol{x}^t + \Delta t \dot{\boldsymbol{x}}^t \\ \dot{\boldsymbol{x}}^{t+\Delta t} &= \dot{\boldsymbol{x}}^t + \Delta t \ddot{\boldsymbol{x}}^t \end{aligned}$$

    - Simple iterative method
    - Commonly used
    - Very inaccurate 很不准确
      - With numerical integration, errors accumulate
      - Euler integration is particularly bad
    - Most often goes unstable 不稳定

- 容易出现正反馈，离正确结果越来越远



Errors and Instability

Solving by numerical integration with finite differences leads to two problems:数值方法解微分方程都会面临

Errors 误差 不是特别大的问题

- Errors at each time step accumulate. Accuracy decreases as simulation proceeds
- Accuracy may not be critical in graphics applications

Instability **不稳定性 很要命**！

- Errors can compound, causing the simulation to **diverge** even when the underlying system does not 收敛很重要！
- Lack of stability is a fundamental problem in simulation, and cannot be ignored

How to determine / quantize “stability”?

• We use the local truncation error (every step) / total accumulated error (overall)

• Absolute values do not matter, but the orders w.r.t. step 研究误差和步长的关系（几次方？）

• Implicit Euler has order 1, which means that

- Local truncation error: O(h 2) and
- Global truncation error: O(h) (h is the step, i.e. Δt)

• Understanding of O(h)

- If we halve h, we can expect the error to halve as well

一些对抗**不稳定性的**方法：

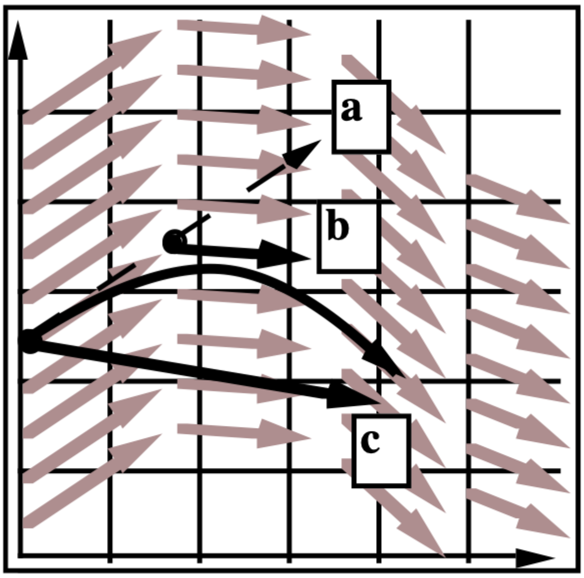
Midpoint method / Modified Euler (中点法)

- Average velocities at start and endpoint

Midpoint method

- Compute Euler step (a)
- Compute derivative at midpoint of Euler step (b)
- Update position using midpoint derivative (c)

$$x_{\mathrm{mid}} = x(t) + \Delta t/2 \cdot v(x(t), t)$$
$$x(t + \Delta t) = x(t) + \Delta t \cdot v(x_{\mathrm{mid}}, t)$$



Witkin and Baraff

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \frac{\Delta t}{2}\left(\dot{\boldsymbol{x}}^t + \dot{\boldsymbol{x}}^{t+\Delta t}\right)$$
$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^t + \Delta t \ddot{\boldsymbol{x}}^t$$
$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t \dot{\boldsymbol{x}}^t + \frac{(\Delta t)^2}{2}\ddot{\boldsymbol{x}}^t$$

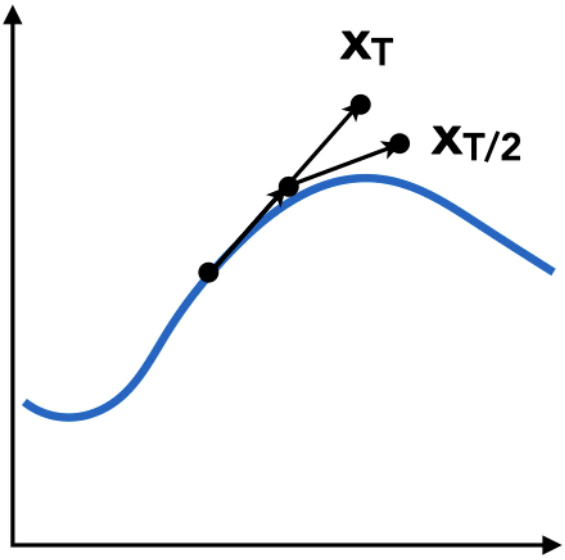
- 多了一个二次项

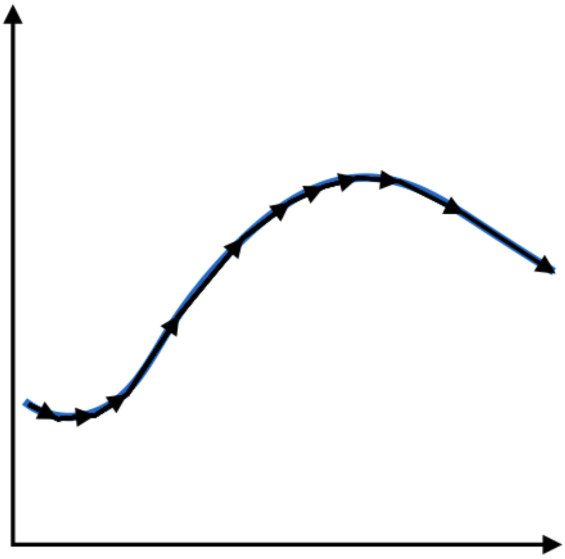
Adaptive step size

- Compare one step and two half-steps, recursively, until error is acceptable
- Repeat until error is below threshold:
    - Compute x_T an Euler step, size T
    - Compute x_T/2 two Euler steps, size T/2
    - Compute error || x T – x T/2 ||
    - If (error > threshold) reduce step size and try again

      如果两者相差很远，再缩小步长
- Technique for choosing step size based on error estimate
- Very practical technique

- But may need very small steps!





Implicit methods (backward 隐式、后向)

- Use the velocity at the next time step (hard)

$$
\begin{aligned}
\boldsymbol{x}^{t+\Delta t} &= \boldsymbol{x}^{t} + \Delta t \dot{\boldsymbol{x}}^{t+\Delta t} \\
\dot{\boldsymbol{x}}^{t+\Delta t} &= \dot{\boldsymbol{x}}^{t} + \Delta t \ddot{\boldsymbol{x}}^{t+\Delta t}
\end{aligned}
$$

- 是一个方程组，有三个未知数，只能假设一个已知（猜）
- Use root-finding algorithm, e.g. Newton's method
- Offers much better stability
- Implicit Euler has order 1, which means that
  - Local truncation error: O(h^2) and
  - Global truncation error: O(h) (h is the step, i.e. Δt)

  O(h)的理解

  - If we halve h, we can expect the error to halve as well
  - 阶数越高越好，减小步长的情况下降低更快

Runge-Kutta Families (龙格库塔方法)

A family of advanced methods for solving ODEs(常微分)

Especially good at dealing with non-linearity

It's order-four version is the most widely used, a.k.a. **RK4**

Initial condition:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

RK4 solution:

$$y_{n+1} = y_n + \frac{1}{6} h \left(k_1 + 2k_2 + 2k_3 + k_4\right),$$
$$t_{n+1} = t_n + h$$

where

$$k_1 = f(t_n, y_n),$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),$$
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right),$$
$$k_4 = f\left(t_n + h, y_n + hk_3\right).$$

更多：**Numerical Analysis** 对图形学有用 （其他的：信号处理）

Position-based / Verlet integration

- Constrain positions and velocities of particles after time step

假设弹簧劲度系数无限大-->恢复

Idea:

- After modified Euler forward-step, constrain positions of particles to prevent divergent, unstable behavior
- Use constrained positions to calculate velocity
- Both of these ideas will dissipate energy, stabilize

Pros / cons

- Fast and simple
- Not physically based, dissipates energy (error)

Rigid body simulation

不会形变

Simple case

- Similar to simulating a particle

- Just consider a bit more properties 拓展

$$\frac{d}{dt}\begin{pmatrix} \mathrm{X} \\ \theta \\ \dot{\mathrm{X}} \\ \omega \end{pmatrix} = \begin{pmatrix} \dot{\mathrm{X}} \\ \omega \\ \mathrm{F}/M \\ \Gamma/I \end{pmatrix}$$

X : positions
θ : rotation angle
ω : angular velocity
F : forces
Γ : torque
I : momentum of inertia

Fluid simulation

A Simple Position-Based Method

模拟只需要输出物体的位置，其他的就是渲染的问题了

Key idea

- Assuming water is composed of small rigid-body spheres 水是由小球组成的
- Assuming the water cannot be compressed (i.e. const. density) 不可压缩
- So, as long as the density changes somewhere, it should be "corrected" via changing the positions of particles 密度有变化，就要想着改回去，移动小球位置
- You need to know the gradient of the density anywhere w.r.t. each particle's position

  一个小球位置的变化对其周围密度的影响
- Update? Just gradient descent! 梯度下降

非物理

流体模拟中两种不同的思路：Eulerian vs. Lagrangian

Lagrangian：质点法，以每个元素为单位模拟

Eulerian：网格法，以空间为单位分割模拟

Material Point Method (MPM)

Hybrid, combining Eulerian and Lagrangian views

- Lagrangian: consider particles carrying material properties 物质由粒子组成，粒子有属性
- Eulerian: use a grid to do numerical integration 网格上计算如何运动
- Interaction: particles transfer properties to the grid, grid performs update, then interpolate back to particles 网格属性写回网格内粒子上