Lecture 3

Particle Systems & ODEs

FUNDAMENTALS OF COMPUTER GRAPHICS
Animation & Simulation
Stanford CS248B, Fall 2022
PROFS. KAREN LIU & DOUG JAMES
Announcements

- **HW1 out last Thursday**
  - Due Thursday
  - Last question is related to material from this class (ODE integrators)
- Grade breakdown for course:

<table>
<thead>
<tr>
<th>Material</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW1</td>
<td>6%</td>
</tr>
<tr>
<td>HW2</td>
<td>6%</td>
</tr>
<tr>
<td>HW3</td>
<td>6%</td>
</tr>
<tr>
<td>HW4</td>
<td>6%</td>
</tr>
<tr>
<td>HW5</td>
<td>6%</td>
</tr>
<tr>
<td>P1</td>
<td>12%</td>
</tr>
<tr>
<td>P2</td>
<td>12%</td>
</tr>
<tr>
<td>P3</td>
<td>13%</td>
</tr>
<tr>
<td>P4</td>
<td>13%</td>
</tr>
<tr>
<td>Final exam</td>
<td>20%</td>
</tr>
</tbody>
</table>
Overview

- Particle Systems
- Ordinary Differential Equations (ODEs)
  - Time-stepping schemes (forward Euler, backward Euler, symplectic Euler, midpoint, ...).
  - Stiffness and stability.
Particle Systems
One lousy particle

- Consider a single particle in n dimensions
- Kinematic model of particle motion (vs *dynamics* which involves forces)
- Position, \( \mathbf{p} = \mathbf{p}(t) \in \mathbb{R}^n \)
- Velocity, \( \mathbf{v} = \mathbf{v}(t) \in \mathbb{R}^n \)
- Differential relationship
  \[
  \mathbf{v} = \frac{d\mathbf{p}}{dt} = \dot{\mathbf{p}}
  \]
- Example: 2D ball

\[
\mathbf{p}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \quad \text{and} \quad \mathbf{v}(t) = \begin{pmatrix} v_x(t) \\ v_y(t) \end{pmatrix} = \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \end{pmatrix}
\]
**Particle systems**

- Collection of $N$ particles in $\mathbb{R}^n$
  - Created or destroyed by various means.
- Indexed particle values:
  \[
  p_i = p_i(t) \in \mathbb{R}^n, \quad \text{for} \quad i = 1, \ldots, N.
  \]
  \[
  v_i = v_i(t) \in \mathbb{R}^n, \quad \text{for} \quad i = 1, \ldots, N.
  \]
- Vector form: $v = \dot{p} \in \mathbb{R}^{nN}$ where

\[
\begin{bmatrix}
  p_1 \\
  \vdots \\
  p_i \\
  \vdots \\
  p_N
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
  v_1 \\
  \vdots \\
  v_i \\
  \vdots \\
  v_N
\end{bmatrix}
\]
Zeroth-order dynamics

- Directly specify position, $p(t)$, function
  - Zeroth position derivative

- Examples:
  - Parametric equations, e.g., heart curve
  - Keyframe animation (more later)

- Pros:
  - Full control
  - No simulation or dynamics

- Cons:
  - Tedious & hard to do complex motions
  - No physics, collisions or other interactions

Examples:

$$x = 16 \sin^3(t)$$
$$y = 13 \cos(t)$$
$$5 \cos(2t) - 2 \cos(3t) - \cos(4t)$$

https://mathworld.wolfram.com/HeartCurve.html
https://www.openprocessing.org/sketch/966495
First-order dynamics

- Specify velocity, \( v(t) \)
  - Integrate \( v(t) \) to get \( p(t) \) of particles
  - Solve an ordinary differential equation (ODE)

- Solve an initial value problem (IVP):
  - Solve \( \dot{p} = v \) for \( p(t) \) such that \( p(0) = \bar{p} \).
  - In general, input velocity is given by some gradient function \( v(t) = V(p(t), t) \)

  Solution: \( p(t + \Delta t) = p(t) + \int_{t}^{t+\Delta t} v(\tau) d\tau \)

- Numerical approximation: Forward Euler scheme
  - \( p(t + \Delta t) = p(t) + v(t) \Delta t \)
  - \( O(\Delta t^2) \) truncation error
Application: Particle advection

- Important application of first-order dynamics
  - Advection of massless particles (e.g., soot) in known spatiotemporal velocity fields

- Integrate $\dot{p} = V(p, t)$
  - subject to initial condition, $p(0) = \bar{p}$.

- Millions to billions of particles in practice

- Pleasantly parallel integration

https://www.youtube.com/watch?v=f51ScQ162wA&ab_channel=HAGI
Application: Energy minimization

- Consider particular motion under gradient descent

\[
\min_\mathbf{p} E(\mathbf{p})
\]

- Numerical approximation

\[
\mathbf{p}_{i+1} = \mathbf{p}_i - h \nabla E(\mathbf{p}_i)
\]

\[
h \to 0 \quad \frac{d\mathbf{p}}{dt} = -\nabla E(\mathbf{p})
\]

https://www.sciencedaily.com/releases/2012/11/121122152910.htm
Application: Sampling Implicit Surfaces

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

- Particles
  - Constrained to lie on an iso-surface, \( d(p) = 0 \)
  - Repel each other using the energy:
    \[
    E_{ij} = e \frac{-||p_i - p_j||^2}{2\sigma_i^2}
    \]
  - Particle i energy:
    \[
    E_i(p) = \sum_{j=1}^{N} (E_{ij} + E_{ji})
    \]
  - Move particles around using gradient descent
Application: Sampling Implicit Surfaces

Visualization:

Pointshell generation using particle repulsion

(a subset of the CAD scene)

From [Barbic and James 2007]
Second-order dynamics: "f=ma"

- Specify particle acceleration, \( a = \frac{d^2p}{dt^2} \), and integrate twice.

- Newton's equations of motion for a particle system:
  \[
  m_i \frac{dp_i^2}{dt^2} = f_i \quad \text{for} \quad i = 1, \ldots, N.
  \]
  - \( m_i \) is the mass of particle \( i \)
  - \( f_i = f_i(p, v, t) \) are external forces
  - Second-order ODE for \( p(t) \)

- Matrix form
  - \( M \ddot{p} = f \) where \( M = \text{diag}(m_1I_n, \ldots, m_NI_n) \)
Simple particle forces

- **Constant gravity:**
  \[ \mathbf{f}_i = m \mathbf{g} \]

- **Aerodynamic drag:**
  \[ \mathbf{f}_i = -c \mathbf{v}_i, \quad c > 0 \]

- **Zero-rest-length springs:**
  - Pulling particle to anchor point, \( c \):
    \[ \mathbf{f}_i = -k (\mathbf{p}_i - \mathbf{c}) \]
  - Pulling two particles together:
    \[ \mathbf{f}_{i/j} = \pm k (\mathbf{p}_i - \mathbf{p}_j) \]

- **Newton's law of gravitation (c.f. electrostatic forces)**
  \[ \mathbf{f}_{i/j} = \pm G m_i m_j \frac{(\mathbf{p}_i - \mathbf{p}_j)}{||\mathbf{p}_i - \mathbf{p}_j||^3} \]
Ordinary Differential Equations (ODEs)
Basic Theory and Numerical Integration
Ordinary Differential Equations (ODEs)

- Initial value problems (IVPs)
  - Common in animation

- Textbook reference:
  - Solomon, Justin. *Numerical Algorithms.*
Conversion to first-order ODE form

- Can always convert a system of ODEs to first-order form
  - Useful step for applying standard ODE solvers
  - Explicit first-order form looks like $\dot{y} = F(y, t)$

- Convert $M \ddot{p} = f$ by writing

$$
\begin{align*}
\frac{dp}{dt} &= v \\
\frac{dv}{dt} &= M^{-1} f
\end{align*}
\quad \iff 
\begin{align*}
\dot{p} &= v \\
\dot{v} &= M^{-1} f
\end{align*}
$$

- Let $y = \begin{pmatrix} p \\ v \end{pmatrix} \in \mathbb{R}^{2nN}$ then $\dot{y} = F(y, t) = \begin{pmatrix} v(t) \\ M^{-1} f(y, t) \end{pmatrix}$ s.t. $y(0) = \begin{pmatrix} \bar{p} \\ \bar{v} \end{pmatrix}$
Time dependence of gradient function, $F$

\[
\dot{y} = F(y)
\]

Time-independent

\[
\dot{y} = F(y, t)
\]

Time-dependent
Autonomous & non-autonomous ODEs

- Minor issue: We have a non-autonomous ODE (explicit time dependence)
  \[ \dot{y} = F(y, t) \]
  but some solvers assume autonomous ODE form (no explicit time dependence)
  \[ \dot{z} = G(z) \]

- There's an easy fix to convert a non-autonomous ODE, \( \dot{y} = F(y, t) \), to autonomous form

- Introduce a time-like variable \( \tau = \tau(t) \) with \( \dot{\tau} = 1 \), and \( \tau(0) = 0 \) (solution: \( \tau = t \)).

- Form an augmented system, and replace all RHS references to \( t \) with \( \tau \) :
  \[
  \frac{d}{dt} \begin{pmatrix} y \\ \tau \end{pmatrix} = \begin{pmatrix} F(y, \tau) \\ 1 \end{pmatrix} \quad \iff \quad \dot{z} = G(z)
  \]

  where \( z(t) = \begin{pmatrix} y(t) \\ \tau(t) \end{pmatrix} \) and \( z(0) = \begin{pmatrix} \bar{y} \\ 0 \end{pmatrix} \).
Existence and Uniqueness of Solutions to $y' = F(y)$

**Theorem: Local existence and uniqueness**

Suppose $F$ is continuous and Lipschitz, that is,
$$
\| F[\vec{y}] - F[\vec{x}] \|_2 \leq L \| \vec{y} - \vec{x} \|_2
$$
for some fixed $L \geq 0$. Then, the ODE $f'(t) = F[f(t)]$ admits exactly one solution for all $t \geq 0$ regardless of initial conditions.

- **Issue:** Many problems in animation may not be sufficiently continuous due to non-smooth contact, e.g., instantaneous velocity changes.
Linear Model Equations

- Common to study stability and behavior of time-stepping schemes on model equations
- Consider the 1D ODE: $\dot{y} = F(y)$
  - Linearizing problem:
    - $F(y) = F(0) + F'(0) y = a y + b = a \left( y + \frac{b}{a} \right) = a \bar{y}$
    - Since $\frac{d}{dt} y' = \frac{d}{dt} \bar{y}'$ it is sufficient to consider...

Model Problem:

$\dot{y} = a y \implies y(t) = C e^{at}$
Stability: Visualization

\[ \dot{y} = a \cdot y \]

- **Unstable**: Solutions separate when \( a > 0 \)
- **Stable**: Solutions get closer when \( a < 0 \)
- **Stable**: Solutions remain unchanged when \( a = 0 \)
Stability and error in initial conditions

An *unstable* ODE magnifies mistakes in the initial conditions $y(0)$. 

![Graph showing stability and instability of ODEs](image)
Time-stepping schemes

- Consider $\dot{y} = F(y)$
- Given $y_k$ at time $t_k$ generate $y_{k+1}$ at time $t_{k+1} = t_k + h$
- Things to consider
  - Accuracy
    - Local truncation error
    - Global truncation error
  - Stability
    - Analyze model problem $y' = ay$ ($a < 0$)
    - Stable when $|y_{k+1}| \leq |y_k|$
    - Unconditional vs conditional stability
      - Time-step restrictions
Forward Euler (a.k.a. Explicit Euler)

\[ y_{k+1} = y_k + h \, F(y_k) \]

- Explicit method
- \( O(h^2) \) localized truncation error
- \( O(h) \) global truncation error; “first order accurate”

Note: Take \( O\left(\frac{1}{h}\right) \) steps to advance \( O(1) \) time.
Forward Euler: Stability on model problem, $\dot{y} = ay$

Stable ($a = -0.4$)

Unstable ($a = -2.3$)
Forward Euler: Stability on model problem, $\dot{y} = ay$

$y' = ay \longrightarrow y_{k+1} = (1 + ah)y_k$

For $a < 0$, stable when $h < \frac{2}{|a|}$.

Forward Euler has a well-known time-step restriction for stability.
Backward Euler (a.k.a. Implicit Euler)

\[ y_{k+1} = y_k + h \, F(y_{k+1}) \]

- Implicit method
- \( O(h^2) \) localized truncation error
- \( O(h) \) global truncation error; “first order accurate”
Backward Euler: Stability on model problem, \( \dot{y} = ay \)
Backward Euler: Stability on model problem, \( \dot{y} = ay \)

\[
y' = ay \quad \Rightarrow \quad y_{k+1} = \frac{1}{1 - ah} y_k
\]

*Unconditionally* stable!

But this has nothing to do with accuracy.

Good for *stiff* equations.
Numerical Stiffness for IVP ODEs

An IVP is said to be numerically “stiff” if stability requirements dictate a much smaller time step size than is needed to satisfy the approximation requirements alone.

[Ascher & Petzold 1998]
Midpoint Method

\[
y_{k+\frac{1}{2}} = y_k + \frac{h}{2} F(y_k) \quad \text{Forward Euler half-step}
\]

\[
y_{k+1} = y_k + h F(y_{k+\frac{1}{2}}) \quad \text{Full-step w/ mid-point gradient}
\]

- Explicit method
- Two function evaluations, but ...
- \(O(h^3)\) localized truncation error
- \(O(h^2)\) global truncation error
  - "second-order accurate"
Midpoint Method: Stability on model problem, $\dot{y} = ay$

$$y_{k+1} = y_k + h \left( ay_{k+\frac{1}{2}} \right) = y_k + ha \left( y_k + \frac{h}{2} ay_k \right) = \left( 1 + ha + \frac{h^2a^2}{2} \right) y_k$$

- Stable when $|y_{k+1}| \leq |y_k|$ or $\left| 1 + ha + \frac{h^2a^2}{2} \right| \leq 1$
- Requires $h \leq \frac{2}{|a|}$ time-step restriction (same as forward Euler).
Symplectic Euler Method (a.k.a. semi-implicit Euler)

- So far time-stepping schemes work with $y = \begin{pmatrix} p \\ v \end{pmatrix}$, and treat $p$ and $v$ similarly.

- Downside: Big forces update $v$ immediately, but $p$ only sees them on the next timestep.
  - Bad for collisions.

- Idea: Update $v$ first, then use it to update $p$

- Aside: Could also update $p$ first, then use it to update $v$
  - So-called adjoint version
  - Valid approach, but it delays force integration
Symplectic Euler Method (a.k.a. semi-implicit Euler)

\[
\begin{align*}
    \mathbf{v}_{k+1} &= \mathbf{v}_k + h \mathbf{a}(\mathbf{p}_k, \mathbf{v}_k) & \text{Update velocity} \\
    \mathbf{p}_{k+1} &= \mathbf{p}_k + h \mathbf{v}_{k+1} & \text{Update position}
\end{align*}
\]

- Semi-implicit method
- One function evaluation
- $O(h^2)$ localized truncation error
- $O(h)$ global truncation error
  - "first-order accurate"
- Symplectic structure
  - preserves area in position-momentum phase space
Symplectic Euler Method: Stability on model problem, $\dot{y} = ay$
Harmonic oscillator model problem

\[ \frac{d}{dt} \begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} v \\ -x \end{pmatrix} \iff \dot{r} = r_{\perp} \]

- Harmonic oscillator equation

\[ \ddot{x} + \omega^2 x = 0 \]

with unit natural frequency, \( \omega = 1 \).
Stability on harmonic oscillator model problem

Forward Euler
Stability on harmonic oscillator model problem

Backward Euler
Example: Particle advection on vortex

https://www.openprocessing.org/sketch/966498
Stability on harmonic oscillator model problem

Forward vs Symplectic Euler

https://www.av8n.com/physics/symplectic-integrator.htm
Symplectic Integrators

**Figure 1:** Phase Space of a Harmonic Oscillator

**Figure 3:** Non-Symplectic Integration: Harmonic Oscillator

**Figure 2:** Symplectic Integration: Harmonic Oscillator

[https://www.av8n.com/physics/symplectic-integrator.htm](https://www.av8n.com/physics/symplectic-integrator.htm)