Differential Equations & Particle Systems

Thanks to Trueille, Popovic, Baraff, Witkin
Physics-based Animation

http://physbam.stanford.edu/~fedkiw/animations/large_pile.avi
Passive—no muscles or motors

Initial conditions

User → Model

State

Model → Numerical integrator

Graphics

Active—internal sources of energy

Desired behavior

User → Control

Forces and torques

Control → Model

State

Model → Numerical integrator

Graphics

Particle systems
Leaves
Water
Smoke
Clothing

Running human
Trotting dog
Swimming fish
Dynamics

- Generate motion by specifying mass and force, apply physical laws (e.g., Newton’s laws)
  - particles
  - soft objects
  - rigid bodies
- Simulates physical phenomena
  - gravity
  - momentum (inertia)
  - collisions
  - friction
  - fluid flow (drag, turbulence, ...)
  - solidity, flexibility, elasticity
  - fracture

Maya Dynamics
Describing Physics

source: http://people.rit.edu/andpph/exhibit-8.html
What variables do we need?

<table>
<thead>
<tr>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Radius</td>
<td>• Position</td>
</tr>
<tr>
<td>• Mass</td>
<td>• Velocity</td>
</tr>
<tr>
<td>• Racquet Info</td>
<td>• Rotation?</td>
</tr>
</tbody>
</table>

http://people.rit.edu/andpph/exhibit-8.html
What Happens Next?

- Position
- Velocity

\[ x = \begin{bmatrix} x & y & z \\ \dot{x} & \dot{y} & \dot{z} \end{bmatrix} \]

Discrete Time: \( x_{t+1} = f(x_t) \)

Continuous Time: \( \dot{x} = f(x) \)

http://people.rit.edu/andpph/exhibit-8.html
Differential Equations

\[ \dot{x} = f(x) \]
Differential Equation Basics

Andrew Witkin

PIXAR Animation Studios
A Canonical Differential Equation

\[ \dot{x} = f(x, t) \]

- \( x(t) \): a moving point.
- \( f(x,t) \): \( x \)'s velocity.
Vector Field

The differential equation

$$\dot{x} = f(x, t)$$

defines a vector field over $x$. 
Integral Curves

Start Here

Pick any starting point, and follow the vectors.
Initial Value Problems

Given the starting point, follow the integral curve.
Euler’s Method

- Simplest numerical solution method
- Discrete time steps
- Bigger steps, bigger errors.

\[ x(t + \Delta t) = x(t) + \Delta t f(x, t) \]
Two Problems

• Accuracy
• Instability
Accuracy

Consider the equation:

\[ \dot{x} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x \]

What do the integral curves look like?
Problem I: Inaccuracy

Error turns $x(t)$ from a circle into the spiral of your choice.
Problem 2: Instability

• Consider the following system:

\[
\begin{align*}
\dot{x} &= -x \\
x(0) &= 1
\end{align*}
\]
Problem 2: Instability to Neptune!
Accuracy of Euler Method

\[ \dot{x} = f(x) \]

Consider Taylor Expansion about \( x(t) \)...

\[
x(t + h) = x(t) + h \cdot f(x(t)) + O(h^2)
\]

Euler’s method has error \( O(h^2) \)… first order.

How can we get to \( O(h^3) \) error?
Euler’s method has a speed limit

\[ x = -kx \quad \text{and} \quad \Delta x = -hkx \]

\[
\begin{array}{cccc}
0 & x=0 & x=0 & x=0 \\
6h & h = .5\left(\frac{1}{k}\right) & h = 1\left(\frac{1}{k}\right) & h = 1.5\left(\frac{1}{k}\right) \\
5h & h = 2\left(\frac{1}{k}\right) & h = 3\left(\frac{1}{k}\right) \\
4h & \\
3h & \\
2h & \\
h & \\
6h & \\
\end{array}
\]

\[ h > 1/k: \text{ oscillate.} \quad h > 2/k: \text{ explode!} \]
The Midpoint Method

- Also known as second order Runge-Kutta:

\[ k_1 = h(f(x_0, t_0)) \]

\[ k_2 = hf(x_0 + \frac{k_1}{2}, t_0 + \frac{h}{2}) \]

\[ x(t_0 + h) = x_0 + k_2 + O(h^3) \]
The Midpoint Method

a. Compute an Euler step
\[ \Delta x = \Delta t f(x, t) \]

b. Evaluate \( f \) at the midpoint
\[ f_{\text{mid}} = f\left(\frac{x + \Delta x}{2}, \frac{t + \Delta t}{2}\right) \]

c. Take a step using the midpoint value
\[ x(t + \Delta t) = x(t) + \Delta t f_{\text{mid}} \]
4th-Order Runge-Kutta

\[ k_1 = hf(x_0, t_0) \]

\[ k_2 = hf(x_0 + \frac{k_1}{2}, t_0 + \frac{h}{2}) \]

\[ k_3 = hf(x_0 + \frac{k_2}{2}, t_0 + \frac{h}{2}) \]

\[ k_4 = hf(x_0 + k_3, t_0 + h) \]

\[ x(t_0 + h) = x_0 + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + O(h^5) \]
q-Stage Runge-Kutta

General Form:

\[ x(t_0 + h) = x_0 + h \sum_{i=1}^{q} w_i k_i \]

where:

\[ k_i = f \left( x_0 + h \sum_{j=1}^{i-1} \beta_{ij} k_j \right) \]

Find the constant that ensures accuracy \( O(h^n) \).
stability is all stability is all stability is all

• If your step size is too big, your simulation blows up. It isn’t pretty.

• Sometimes you have to make the step size so small that you never get anyplace.

• Nasty cases: cloth, constrained systems.
Implicit Euler Method

\[ x(t_0 + h) = x(t_0) + h \dot{x}(t_0) \]

\[ x(t_0 + h) = x(t_0) + h \dot{x}(t_0 + \Delta t) \]
Implicit Euler for $\dot{x} = -kx$

\[
x(t + h) = x(t) + h \dot{x}(t + h)
\]
\[
= x(t) - h k x(t + h)
\]
\[
= \frac{x(t)}{1 + hk}
\]
One Step: Implicit vs. Explicit

\[ \dot{x} = -x, \quad x(0) = 1 \]

**Correct Solution:**  
\[ x(h) = e^{-hk} \]

**Implicit Euler Step:**  
\[ x(h) = \frac{1}{1+hk} \]

**Explicit Euler Step:**  
\[ x(h) = 1 - hk \]
Modular Implementation

- Generic operations:
  - Get \( \text{dim}(x) \)
  - Get/set \( x \) and \( t \)
  - Deriv Eval at current \((x,t)\)

- Write solvers in terms of these.
  - Re-usable solver code.
  - Simplifies model implementation.
Solver Interface

System

Dim(state)

Get/Set State

Solver

Deriv Eval
void eulerStep(Sys sys, float h) {
    float t = getTime(sys);
    vector<float> x0, deltaX;
    t = getTime(sys);
    x0 = getState(sys);
    deltaX = derivEval(sys, x0, t);
    setState(sys, x0 + h*deltaX, t+h);
}
Particle Systems
Particle Systems

Clouds
Smoke
Fire
Waterfalls
Fireworks

 Reeves ’83, the Wrath of Khan
Batman Returns, using Reynold’s flocking algorithms
Karl Sims, Particle Dreams
Spring-Mass Systems

Cloth in 2D
Jello in 3D
Cloth Simulation

Cloth forces:
- Blue (short horizontal & vertical) = stretch springs
- Green (diagonal) = shear springs
- Red (long horizontal & vertical) = bend springs
Cloth

Many types of cloth
Very different properties
Not a simple elastic surface
Woven fabrics tend to be very stiff
Anisotropic

Breen ‘95
Artificial Fish