Animation Physics
CPSC 533B

Course Details

- http://www.cs.ubc.ca/~rbridson/courses/533b-winter-2004
- Course schedule
- Assignments
- Resources (papers to read!)
- Final Project information

Instructor

- Robert Bridson
  - CICSR 189: usually 9:30-5:30
  - Drop by, or make an appointment
  - 822-1993 (or just 21993)
  - email rbridson@cs.ubc.ca

Evaluation

- 6 assignments (85%)
  - #1 is a warm-up (10%) - given out today
  - #2-#6 are each 15%
  - Mostly programming, with a little analysis (writing)
- Also a final project (15%)
  - Extend one of assignments #2-#6
  - Or: do what you want, but talk to me about it
  - Present in final class - informal talk, show movies
- Late: without a good reason, 20% off per day
  - For final project starts after final class
  - For assignments starts morning after due

Why?

- Animating natural phenomena: passive (secondary) motion
- Film/TV: passive motion difficult with traditional techniques
  - When you control every detail of the motion, it's hard to make it look like it's not being controlled!
- Games: difficult to handle everything convincingly with prescripted motion

- Instead: directly simulate the underlying physics to get realistic motion

Topics

- Particle Systems
  - most common simulated special effect
- Rigid Bodies
- Deformable Bodies
  - e.g. cloth and flesh
- Fluids
  - smoke and water

Particle Systems

- Read:
  Reeves, "Particle Systems…", SIGGRAPH'83
- Some phenomena is most naturally described as many small particles
  - Rain, snow, dust, sparks, gravel, …
- Others are difficult to get a handle on
  - Fire, water, grass, …

Particle Basics

- Each particle has a position
  - Maybe orientation, age, colour, velocity, temperature, radius, …
  - Call the state x
- Seeded randomly somewhere at start
  - Maybe some created each frame
- Move (evolve state x) each frame according to some formula
- Eventually die when some condition met

Example

- Sparks from a campfire
- Every frame (1/24 s) add 2-3 particles
  - Position randomly in fire
  - Initialize temperature randomly
- Move in specified turbulent smoke flow
  - Also decrease temperature
- Render as a glowing dot (blackbody radiation from temperature)
- Kill when too cold to glow visibly

Rendering

- We won't talk much about rendering in this course, but most important for particles
- The real strength of the idea of particle systems: how to render
  - Could just be coloured dots
  - Or could be shards of glass, or animated sprites (e.g. fire), or deforming blobs of water, or blades of grass, or birds in flight, or …

Motion (1st order)

- For each particle, have a simple 1st order differential equation:

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

- Need to solve this numerically forward in time from x(t=0) to x(frame1), x(frame2), x(frame3), …

Forward Euler

- Simplest method:

$$x_{n+1} = x_n + \Delta t\, v(x_n, t_n)$$

- Can show it's first order accurate:
  - Error accumulated by a fixed time is O(Δt)
- Thus it converges to the right answer
  - Do we care?

Aside on Error

- General idea - want error to be small
  - Obvious approach: make Δt small
  - But then need more time steps - expensive
- Also note - O(1) error made in modeling
  - Even if numerical error was 0, still wrong!
  - In science, need to validate against experiments
  - In graphics, the experiment is showing it to an audience: **does it look real?**
- So numerical error can be huge, as long as your solution has the right qualitative look

Forward Euler Stability

- Big problem with Forward Euler: it's not very stable
- Example: $dx/dt = -x, \quad x(0) = 1$
- Real solution e^{-t} smoothly decays to zero, always positive
- Run Forward Euler with Δt=11
  - x=1, -10, 100, -1000, 10000, …
  - Instead of 1, $1.7*10^{-5}$, $2.8*10^{-10}$, …

Linear Analysis

- Approximate
$$v(x,t) \approx v(x^*,t^*) + \frac{\partial v}{\partial x} \cdot (x - x^*) + \frac{\partial v}{\partial t} \cdot (t - t^*)$$
- Ignore all but the middle term (the one that could cause blow-up)
$$dx/dt = Ax$$
- Look at x parallel to eigenvector of A: the "test equation" $dx/dt = \lambda x$

The Test Equation

- Get a rough, hazy, heuristic picture of the stability of a method
- Note that eigenvalue λ can be complex
- But, assume that for real physics
  - Things don't blow up without bound
  - Thus **real** part of eigenvalue λ is ≤ 0
- Beware - nonlinear effects can cause instability

More Linear Analysis

- Forward Euler on test equation is

$$x_{n+1} = x_n + \Delta t \lambda x_n$$

- Solving gives

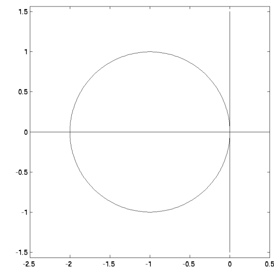
$$x_n = (1 + \lambda \Delta t)^n x_0$$

- So for stability, need

$$|1 + \lambda \Delta t| < 1$$

Stability Region

- Can plot all the values of λΔt on the complex plane where F.E. is stable:



Real Eigenvalue

- Say eigenvalue is real (and negative)
  - Corresponds to a damping motion, smoothly coming to a halt
- Then need:

$$\Delta t < \frac{2}{|\lambda|}$$

- Is this bad?
  - If eigenvalue is big, could mean small time steps
  - But, maybe we really need to capture that timescale anyways, so no big deal

Imaginary Eigenvalue

- If eigenvalue is pure imaginary…
  - Oscillatory or rotational motion
- Cannot make Δt small enough
- Forward Euler unconditionally unstable for these kinds of problems!
- Need to look at other methods

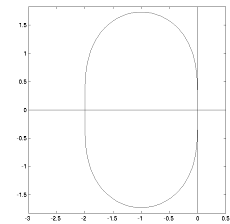
Runge-Kutta Methods

- Also “explicit”
  - next x is an explicit function of previous
- But evaluate v at a few locations to get a better estimate of next x
- E.g. midpoint method (one of RK2)

$$x_{n+\frac{1}{2}} = x_n + \tfrac{1}{2}\Delta t v\left(x_n, t_n\right)$$
$$x_{n+1} = x_n + \Delta t v\left(x_{n+\frac{1}{2}}, t_{n+\frac{1}{2}}\right)$$

Midpoint RK2

- Second order: error is $O(\Delta t^2)$ **when smooth**
- Larger stability region:



- But still not stable on imaginary axis: no point

Modified Euler

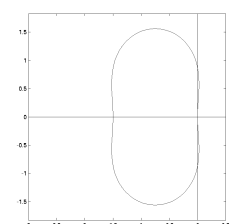
- (Not an official name)
- Lose second-order accuracy, get stability on imaginary axis:

$$x_{n+\alpha} = x_n + \alpha \Delta t v\left(x_n, t_n\right)$$
$$x_{n+1} = x_n + \Delta t v\left(x_{n+\alpha}, t_{n+\alpha}\right)$$

- Parameter α between 0.5 and 1 gives trade-off between imaginary axis and real axis

Modified Euler (2)

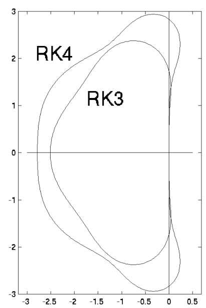
- Stability region for α=2/3



- Great! But twice the cost of Forward Euler
- Can you get more stability per v-evaluation?

Higher Order Runge-Kutta

- RK3 and up naturally include part of the imaginary axis



TVD-RK3

- RK3 useful because it can be written as a combination of Forward Euler steps and averaging: can guarantee stuff!

$$
\begin{aligned}
\tilde{x}_{n+1} &= x_n + \Delta t v(x_n, t_n) \\
\tilde{x}_{n+2} &= \tilde{x}_{n+1} + \Delta t v(\tilde{x}_{n+1}, t_{n+1}) \\
\tilde{x}_{n+\frac{1}{2}} &= \tfrac{3}{4} x_n + \tfrac{1}{4} \tilde{x}_{n+2} \\
\tilde{x}_{n+\frac{3}{2}} &= \tilde{x}_{n+\frac{1}{2}} + \Delta t v\left(\tilde{x}_{n+\frac{1}{2}}, t_{n+\frac{1}{2}}\right) \\
x_{n+1} &= \tfrac{1}{3} x_n + \tfrac{2}{3} \tilde{x}_{n+\frac{3}{2}}
\end{aligned}
$$

RK4

- Often most bang for the buck

$$
\begin{aligned}
v_1 &= v(x_n, t_n) \\
v_2 &= v\left(x_n + \tfrac{1}{2}\Delta t v_1, t_{n+\frac{1}{2}}\right) \\
v_3 &= v\left(x_n + \tfrac{1}{2}\Delta t v_2, t_{n+\frac{1}{2}}\right) \\
v_4 &= v(x_n + \Delta t v_3, t_{n+1}) \\
x_{n+1} &= x_n + \Delta t\left(\tfrac{1}{6} v_1 + \tfrac{2}{6} v_2 + \tfrac{2}{6} v_3 + \tfrac{1}{6} v_4\right)
\end{aligned}
$$

Time Step Control

- Hack: try until it looks like it works
- Stability based:
  - Figure out a bound on eigenvalues of Jacobian
  - Scale back by a fudge factor (e.g. 0.9, 0.5)
- Adaptive error based:
  - Usually not worth the trouble in graphics

Implicit Methods

- Often don't want to be restricted by stability ("stiffness")
- Implicit methods can be unconditionally stable
- Key ingredient:
  - Next x is an implicit function of previous
  - Need to solve a system of equations

Backward Euler

- The simplest implicit method:

$$x_{n+1} = x_n + \Delta t v\left(x_{n+1}, t_{n+1}\right)$$

- First order accurate
- Test equation shows stable when $\left|1 - \lambda \Delta t\right| > 1$
- This includes everything except a circle in the positive real-part half-plane
- It's stable even when the physics is unstable!
- This is the biggest problem: damps out motion unrealistically

Aside: Solving Systems

- If v is linear in x, just a system of linear equations
  - If very small, use determinant formula
  - If small, use LAPACK
  - If large, life gets more interesting…
- If v is mildly nonlinear, can approximate with linear equations ("semi-implicit")

$$\begin{aligned} x_{n+1} &= x_n + \Delta t v\left(x_{n+1}\right) \\ &\approx x_n + \Delta t\left(v(x_n) + \frac{\partial v(x_n)}{\partial x}(x_{n+1} - x_n)\right) \end{aligned}$$

Newton's Method

- For more strongly nonlinear v, need to iterate:
  - Start with guess x_n for x_{n+1} (for example)
  - Linearize around current guess, solve linear system for next guess
  - Repeat, until close enough to solved
- Note: Newton's method is **great** when it works, but it might not work
  - If it doesn't, need to reduce time step size to make equations easier to solve, and try again

Newton's Method: B.E.

- Start with $x^0=x_n$ (guess for x_{n+1})
- For k=1, 2, … find $x^{k+1}=x^k+\Delta x$ by solving

$$x^{k+1} = x_n + \Delta t\left(v(x^k) + \frac{\partial v(x^k)}{\partial x}(x^{k+1} - x^k)\right)$$
$$\Rightarrow \left(I - \Delta t \frac{\partial v(x^k)}{\partial x}\right)\Delta x = x_n + \Delta t\, v(x^k) - x^k$$

- Stop when right-hand side is small enough, set $x_{n+1}=x^k$

Trapezoidal Rule

- Can improve by going to second order:

$$x_{n+1} = x_n + \Delta t\left(\tfrac{1}{2} v(x_n, t_n) + \tfrac{1}{2} v(x_{n+1}, t_{n+1})\right)$$

- This is actually just a half step of F.E., followed by a half step of B.E.
  - F.E. is under-stable, B.E. is over-stable, the combination is **just right**
- Stability region is the left half of the plane: **exactly** the same as the physics!
- Really good for pure rotation (doesn't amplify or damp)

Monotonicity

- Test equation with real, negative λ
  - True solution is $x(t)=x_0e^{\lambda t}$, which smoothly decays to zero, doesn't change sign (**monotone**)
- Forward Euler at stability limit:
  - $x=x_0$, $-x_0$, x_0, $-x_0$, …
- Not smooth, oscillating sign: garbage!
- So monotonicity limit stricter than stability
- RK3 has the same problem
  - But the even order RK are fine for linear problems
  - TVD-RK3 designed so that it's fine when F.E. is, even for nonlinear problems!

Monotonicity and Implicit Methods

- Backward Euler is unconditionally monotone
  - No problems with oscillation, just too much damping
- Trapezoidal Rule suffers though, because of that half-step of F.E.
  - Beware: could get ugly oscillation instead of smooth damping

Summary 1

- Particle Systems: useful for lots of stuff
- Need to move particles in velocity field
- Forward Euler
  - Simple, first choice unless problem has oscillation/rotation
- Runge-Kutta if happy to obey stability limit
  - If time step fixed elsewhere, modified Euler may be cheapest method
  - RK4 general purpose workhorse
  - TVD-RK3 for more robustness with nonlinearity (more on this later in the course!)

Summary 2

- If stability limit is a problem, look at implicit methods
  - e.g. need to guarantee a frame-rate, or explicit time steps are way too small
- Trapezoidal Rule
  - If monotonicity isn't a problem
- Backward Euler
  - Almost always works, but may over-damp!