Addenda to last class

- K. Sims, “Particle animation…”, SIGGRAPH’90
  - Ignore the parallel computing stuff
- There was an inconsistency in assignment #1 (y-axis vs. z-axis)
  - Updated PDF on the web
  - Vertical is now z-axis, horizontal is x-y plane
  - Welcome to a continual problem of axis labeling… (we’re not even looking at right-handed vs. left-handed)

Time Stepping

- Sometimes can pick constant Δt
  - One frame, or 1/8th of a frame, or …
- Often need to allow for variable Δt
  - Changing stability limit due to changing Jacobian
  - Difficulty in Newton converging
  - …
- But need to land at the exact frame time
  - So clamp Δt so you can’t overshoot the frame
- Some algorithms behave oddly if time step changes dramatically…
  - Be careful that last time step isn’t much smaller

Time Stepping Algorithm

- Set done = false
- While not done
  - Find good Δt
  - If $t+\Delta t \geq t_{frame}$
    - Set $\Delta t = t_{frame}-t$
    - Set done = true
  - Else if $t+1.5\Delta t \geq t_{frame}$
    - Set $\Delta t = 0.5(t_{frame}-t)$
  - …process time step…
  - Set $t = t+\Delta t$
- Write out frame data, continue to next frame

Another Word of Caution

- Even for linear problems, stability analysis still not bulletproof
  - Assumes constant time step
  - If time step varies, even under official stability limit, can actually go unstable!
  - See J. P. Wright, “Numerical instability due to varying time steps…”, JCP 1998
  - Safety margin really is a good idea!

1st order vs. 2nd order

- If particle state is just position (and colour, size, …) then 1st order motion
  - No inertia
  - Good for very light particles that stay suspended in air: smoke, dust, …
  - Good for some special cases (hacks)
- But most often, want inertia
  - State includes velocity, specify acceleration
  - Can then do parabolic arcs due to gravity, etc.

Second Order Particle Motion

- This puts us in the realm of standard Newtonian physics
  - F=ma
- Alternatively put:
  - dx/dt=v
  - dv/dt=F(x,v,t)/m (i.e. a(x,v,t))

What's New?

- If **x**=(x,v) this is just a special form of d**x**/dt=**v**(**x**,t)
- But since we know the special structure, can we take advantage of it?
  (i.e. better time integration algorithms)
  - More stability for less cost?
  - Handle position and velocity differently to better control error?

Linear Analysis

- Approximate acceleration:

$$a(x,v) \approx a_0 + \frac{\partial a}{\partial x} x + \frac{\partial a}{\partial v} v$$

- Split up analysis into different cases
- Begin with first term dominating: constant acceleration
  - e.g. gravity is most important

Constant Acceleration

- Solution is $v(t) = v_0 + a_0 t$

  $$x(t) = x_0 + v_0 t + \tfrac{1}{2} a_0 t^2$$
- No problem to get v(t) right:
  just need 1st order accuracy
- But x(t) demands 2nd order accuracy
- So we can look at mixed methods:
  - 1st order in v
  - 2nd order in x

Linear Acceleration

- Dependence on x and v dominates:
  a(x,v)=-Kx-Dv
- Do the analysis from last class:

$$\frac{d}{dt}\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} 0 & I \\ -K & -D \end{pmatrix}\begin{pmatrix} x \\ v \end{pmatrix}$$

- Eigenvalues of this matrix?

More Approximations…

- Typically K and D are symmetric semi-definite (there are good reasons)
  - What does this mean about their eigenvalues?
- Often, D is a linear combination of K and I ("Rayleigh damping"), or at least close to it
  - Then K and D have the same eigenvectors (but different eigenvalues)
  - Then the eigenvectors of the Jacobian are of the form $(u, \alpha u)^T$
  - [work out what α is in terms of λ_K and λ_D]

Simplification

- α is the eigenvalue of the Jacobian, and

$$\alpha = -\tfrac{1}{2}\lambda_D \pm \sqrt{\left(\tfrac{1}{2}\lambda_D\right)^2 - \lambda_K}$$

- Same as eigenvalues of $\begin{pmatrix} 0 & 1 \\ -\lambda_K & -\lambda_D \end{pmatrix}$
- Can replace K and D (matrices) with corresponding eigenvalues (scalars)
  - Just have to analyze 2x2 system

Two Regimes

- Still messy! Simplify further
- If D dominates (e.g. air drag, damping)

$$\alpha \approx \{-\lambda_D, 0\}$$

  - Exponential decay and constant
- If K dominates (e.g. spring force)

$$\alpha \approx \pm i\sqrt{\lambda_K}$$

Three Test Equations

- Constant acceleration (e.g. gravity)
  - a(x,v,t)=g
  - Want exact (2nd order accurate) position
- Position dependence (e.g. spring force)
  - a(x,v,t)=-Kx
  - Want stability but low damping
  - Look at imaginary axis
- Velocity dependence (e.g. damping)
  - a(x,v,t)=-Dv
  - Want stability, smooth decay
  - Look at negative real axis

Explicit methods from before

- Forward Euler
  - Constant acceleration: bad (1st order)
  - Position dependence: very bad (unstable)
  - Velocity dependence: ok (conditionally monotone/stable)
- RK3 and RK4
  - Constant acceleration: great (high order)
  - Position dependence: ok (conditionally stable, but damps out oscillation)
  - Velocity dependence: ok (conditionally monotone/stable)

Implicit methods from before

- Backward Euler
  - Constant acceleration: bad (1st order)
  - Position dependence: ok (stable, but damps)
  - Velocity dependence: good (monotone, 1st order)
- Trapezoidal Rule
  - Constant acceleration: great (2nd order)
  - Position dependence: great (stable, no damping)
  - Velocity dependence: good (stable but only conditionally monotone --- though maybe fixable)

New methods!

- This is again a big subject
- Again look at explicit methods, implicit methods
- Also can treat position and velocity dependence differently:
  mixed implicit-explicit methods

Symplectic Euler

- Like Forward Euler, but updated velocity used for position

$$v_{n+1} = v_n + \Delta t\, a(x_n, v_n)$$
$$x_{n+1} = x_n + \Delta t\, v_{n+1}$$

- Some people flip the steps (= relabel v_n)
- (Symplectic means certain qualities are preserved in discretization; useful in science, not necessarily in graphics)
- [work out test cases]

Symplectic Euler performance

- Constant acceleration: bad
  - Velocity right, position off by O(Δt)
- Position dependence: good
  - Stability limit $\Delta t < \dfrac{2}{\sqrt{K}}$
  - No damping!
- Velocity dependence: ok
  - Monotone limit $\Delta t < 1/D$
  - Stability limit $\Delta t < 2/D$

Tweaking Symplectic Euler

- [sketch algorithms]
- Stagger the velocity to improve x
- Start off with $v_{1/2} = v_0 + \frac{1}{2}\Delta t\, a(x_0, v_0)$
- Then proceed with

$$v_{n+1/2} = v_{n-1/2} + \tfrac{1}{2}(t_{n+1} - t_{n-1})\, a(x_n, v_{n-1/2})$$
$$x_{n+1} = x_n + \Delta t\, v_{n+1/2}$$

- Finish off with $v_N = v_{N-1/2} + \frac{1}{2}\Delta t\, a(x_N, v_{N-1/2})$

Staggered Symplectic Euler

- Constant acceleration: great!
  - Position is exact now
- Other cases not effected
  - Was that magic? Main part of algorithm unchanged (apart from relabeling) yet now it's more accurate!
- Only downside to staggering
  - At intermediate times, position and velocity not known together
  - May need to think a bit more about collisions and other interactions with outside algorithms…

A common explicit method

- May see this one pop up:

$$v_{n+1} = v_n + \Delta t\, a(x_n, v_n)$$
$$x_{n+1} = x_n + \Delta t\left(\tfrac{1}{2} v_n + \tfrac{1}{2} v_{n+1}\right) = x_n + \Delta t v_n + \tfrac{1}{2}\Delta t^2 a_n$$

- Constant acceleration: great
- Velocity dependence: ok
  - Conditionally stable/monotone
- Position dependence: **BAD**
  - Unconditionally unstable!

An Implicit Compromise

- Backward Euler is nice due to unconditional monotonicity
  - Although only 1st order accurate, it has the right characteristics for damping
- Trapezoidal Rule is great for everything except damping with large time steps
  - 2nd order accurate, doesn't damp pure oscillation/rotation
- How can we combine the two?

Implicit Compromise

- Use Backward Euler for velocity dependence, Trapezoidal Rule for the rest:

$$x_{n+1} = x_n + \Delta t\left(\tfrac{1}{2} v_n + \tfrac{1}{2} v_{n+1}\right)$$
$$v_{n+1} = v_n + \Delta t\, a\left(\tfrac{1}{2} x_n + \tfrac{1}{2} x_{n+1}, v_{n+1}, t_{n+1/2}\right)$$

- Constant acceleration: great (2nd order)
- Position dependence: great (2nd order, no damping)
- Velocity dependence: good (unconditionally monotone, but only 1st order accurate)

Time scales

- [work out]
- For position dependence, characteristic time interval is $$\Delta t = O\left(\frac{1}{\sqrt{K}}\right)$$
- For velocity dependence, characteristic time interval is $$\Delta t = O\left(\frac{1}{D}\right)$$
- Note: matches symplectic Euler stability limits

Mixed Implicit/Explicit

- For some problems, that square root can mean velocity limit **much** stricter
- Or, we know we want to properly resolve the position-based oscillations, but don't care about damping
- Go explicit on position, implicit on velocity
  - Also cuts the number of equations to solve in half
  - Often, a(x,v) is linear in v, though nonlinear in x; this way we avoid Newton iteration

Newmark Methods

- A general class of methods
$$x_{n+1} = x_n + \Delta t v_n + \tfrac{1}{2}\Delta t^2\left[(1-2\beta)a_n + 2\beta a_{n+1}\right]$$
$$v_{n+1} = v_n + \Delta t\left[(1-\gamma)a_n + \gamma a_{n+1}\right]$$
- Includes Trapezoidal Rule for example (β=1/4, γ=1/2)
- The other major member of the family is Central Differencing (β=0, γ=1/2)
  - This is mixed Implicit/Explicit

Central Differencing

- Rewrite it with intermediate velocity:
$$v_{n+1/2} = v_n + \tfrac{1}{2}\Delta t\, a(x_n, v_n)$$
$$x_{n+1} = x_n + \Delta t\, v_{n+1/2}$$
$$v_{n+1} = v_{n+1/2} + \tfrac{1}{2}\Delta t\, a(x_{n+1}, v_{n+1})$$
- Looks like a hybrid of:
  - Midpoint (for position), and
  - Trapezoidal Rule (for velocity - split into Forward and Backward Euler half steps)

Central: Performance

- Constant acceleration: great
  - 2nd order accurate
- Position dependence: good
  - Conditionally stable, no damping
- Velocity dependence: good
  - Stable, but only conditionally monotone
- Can we change the Trapezoidal Rule to Backward Euler and get unconditional monotonicity?

Staggered Implicit/Explicit

- Like the staggered Symplectic Euler, but use B.E. in velocity instead of F.E.:

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

- Constant acceleration: great
- Position dependence: good (conditionally stable, no damping)
- Velocity dependence: good (unconditionally monotone, but 1st order)

Time Integration Summary

- Depends a lot on the problem
  - What's important: gravity, position, velocity?
- Explicit methods from last class are bad
- Symplectic Euler is a great fully explicit method (particularly with staggering)
  - Switch to implicit velocity step for more stability
- Implicit Compromise method
  - Fully stable, nice behaviour
- Central Differencing and Trapezoidal Rule
  - More accurate velocity, but may have monotonicity issues for strong damping…

Example Forces

- Gravity: $F_{gravity}$=mg (a=g)
- If you want to do orbits

$$F_{gravity} = -GmM_0 \frac{x - x_0}{\left|x - x_0\right|^3}$$

- Note x_0 could be a fixed point (e.g. the Sun) or another particle
  - But make sure to add the opposite and equal force to the other particle if so!

Spring Forces

- Springs: F_{spring}=-K(x-x_0)
  - x_0 is the attachment point of the spring
  - Could be a fixed point in the scene
  - …or somewhere on a character's body
  - …or the mouse cursor
  - …or another particle (but please add equal and oppposite force!)

Spring Damping

- Simple damping: F_{damp}=-D(v-v_0)
  - But this damps rotation too!
- Better spring damping:
  F_{damp}=-D(v-v_0)•u u
  - Here u is (x-x_0)/|x-x_0|, the spring direction
- [work out 1d case]
- Critical damping $D = 2\sqrt{mK}$

Nonzero Rest Length Spring

- Need to measure the "strain": the fraction the spring has stretched from its rest length L

$$F_{spring} = -K\left(\frac{|x - x_0|}{L} - 1\right)\frac{x - x_0}{|x - x_0|}$$

Drag Forces

- Air drag: F_{drag}=-Dv
  - If there's a wind blowing with velocity v_w then F_{drag}=-D(v-v_w)
- D should be proportional to cross-section exposed to wind
  - Think sheets of paper, leaves…
- Depends in a difficult way on shape too
- How do you come up with a good wind velocity?

Wind

- Later in the course: actually directly simulate the wind
- For now: fake it
  - Random “turbulence”
  - Superposition of basic flow elements
    - Constant wind, vortices, …
  - Key ingredient is incompressibility

Incompressibility

- Air is basically incompressible
  - Acoustic waves are so small as to be ignored usually
  - Large shock waves only around supersonic objects
- The volume of air going into a region of space equals the volume leaving it
- [derive divergence condition]