Numerical Integration

Erin Catto
Blizzard Entertainment
**Basic Idea**

- Games use differential equations for physics.
- These equations are *hard* to solve exactly.
- We can use numerical integration to solve them approximately.
Overview

- Differential Equations
- Numerical Integrators
- Demos
Typical Game Loop

Start $t = 0$ → Choose $\Delta t$ → Player Input → Simulate $\Delta t$ → Render → $t = t + \Delta t$ → Choose $\Delta t$
Simulation

- Animation
- AI
- Physics
  - Differential Equations
What is a differential equation?

- An equation involving derivatives.

\[ \text{rate of change of a variable} = \text{a function} \]
Anatomy of Differential Equations

- State
  - Dependent variables
  - Independent variables
- Initial Conditions
- Model
  - The differential equation itself
Projectile Motion

- **State**
  - Independent variable: time \( t \)
  - Dependent variables: position \( y \) and velocity \( v \)

- **Initial Conditions**
  - \( t_0, y_0, v_0 \)
Projectile Motion

- Model: vertical motion

\[ ma = F \]

\[ m \frac{d^2 y}{dt^2} = -mg \]

\[ \frac{d^2 y}{dt^2} = -g \]
First Order Form

- Numerical integrators need differential equations to be put into a special format.

\[
\frac{dx}{dt} = f(t, x) \]

\[x(0) = x_0\]
First Order Form

- Arrays of equations work too.

\[
\begin{align*}
\frac{dx_1}{dt} &= f_1(t, x_1, \ldots, x_n) \\
& \vdots \\
\frac{dx_n}{dt} &= f_n(t, x_1, \ldots, x_n)
\end{align*}
\]

\[
\frac{dx}{dt} = f(t, x)
\]
Projectile Motion
First Order Form

\[ \frac{d^2 y}{dt^2} = -g \]

\[ \frac{dy}{dt} = v \]

\[ \frac{dv}{dt} = -g \]
Projectile Motion
First Order Form

\[ y(0) = y_0 \]

\[ v(0) = v_0 \]
Mass-Spring Motion

- Consider the vertical motion of a character.
Mass-Spring Motion

- **State**
  - time: $t$
  - position: $x$
  - velocity: $v$

- **Initial Conditions**
  - $t_0$, $x_0$, $v_0$
Mass-Spring Motion

- Idealized model

\[ ma = F \]
\[ m \frac{d^2 x}{dt^2} = -kx \]
\[ \frac{d^2 x}{dt^2} = -\frac{k}{m} x \]
Mass-Spring Motion

- First Order Form

\[
\frac{dx}{dt} = v
\]

\[
\frac{dv}{dt} = -\frac{k}{m} x
\]

\[
x(0) = x_0
\]

\[
v(0) = v_0
\]
Solving Differential Equations

- Sometimes we can solve our DE exactly.
- Many times our DE is too complicated to be solved exactly.
Hard Problems

- Nonlinear equations
- Multiple variables
Hard Problems

- Projectile with air resistance

\[ m \frac{dv}{dt} = -c \left( v \cdot v \right) \frac{v}{\|v\|} - mg \]
Hard Problems

- Mass-spring in 3D

\[ m \frac{dv}{dt} = -k \left( \|x\| - L_0 \right) \frac{x}{\|x\|} \]
Hard Problems

- Numerical integration can help!
  - Handles nonlinearities
  - Handles multiple variables
Numerical Integration

- Start with our first order form

\[ \frac{dx}{dt} = f(t, x) \]

\[ x(0) = x_0 \]
A Simple Idea

- Approximate the slope.

\[ \text{slope} \approx \frac{x(t+h) - x(t)}{h} \]
A Simple Idea

- Forward difference:

\[
\frac{dx}{dt} \approx \frac{x(t+h) - x(t)}{h}
\]
A Simple Idea

- Shuffle terms:

\[ \frac{x(t + h) - x(t)}{h} = f(t, x(t)) \]

\[ x(t + h) = x(t) + hf(t, x(t)) \]
A Simple Idea

- Using this formula, we can make a time step $h$ to find the new state.
- We can continue making time steps as long as we want.
- The time step is usually small

$$x(t + h) = x(t) + h f(t, x(t))$$
Explicit Euler

\[ x(t + h) = x(t) + h f(t, x(t)) \]

- This is called the *Explicit Euler* method.
- All terms on the right-hand side are known.
- Substitute in the known values and compute the new state.
What If …

\[ x(t + h) = x(t) + h f(t + h, x(t + h)) \]

- This is called the *Implicit Euler* method.
- The function depends on the new state.
- But we don’t know the new state!
Implicit Euler

\[ x(t + h) = x(t) + h f(t + h, x(t + h)) \]

- We have to solve for the new state.
- We may have to solve a nonlinear equation.
- Can be solved using Newton-Raphson.
- Usually impractical for games.
Implicit vs Explicit

- Explicit is fast.
- Implicit is slow.
- Implicit is more stable than explicit.
- More on this later.
Opening the Black Box

- Explicit and Implicit Euler don’t know about position or velocity.
- Some numerical integrators work with position and velocity to gain some advantages.
The Position ODE

\[ \frac{dx}{dt} = v \]

- This equation is trivially linear in velocity.
- We can exploit this to our advantage.
Symplectic Euler

\[
\frac{v(t+h) - v(t)}{h} = f(t, x(t), v(t))
\]

\[
\frac{x(t+h) - x(t)}{h} = v(t+h)
\]

- First compute the new velocity.
- Then compute the new position using the new velocity.
Symplectic Euler

- We get improved stability over Explicit Euler, without added cost.
- But not as stable as Implicit Euler
Verlet

- Assume forces only depend on position.
- We can eliminate velocity from Symplectic Euler.

\[
\begin{align*}
\frac{v(t+h) - v(t)}{h} &= f(t, x(t)) \\
\frac{x(t+h) - x(t)}{h} &= v(t+h)
\end{align*}
\]
Verlet

- Write two position formulas and one velocity formula.

\[
\begin{align*}
x_1 &= x_0 + h v_1 \\
x_2 &= x_1 + h v_2 \\
v_2 &= v_1 + h f_1
\end{align*}
\]
Eliminate velocity to get:

\[ x_2 = 2x_1 - x_0 + h^2 f_1 \]
Assume constant force:

\[ x(t + h) = x(t) + v(t)h + \frac{1}{2}ah^2 \]

- Exact for projectiles (parabolic motion).
Demos

- Projectile Motion
- Mass-Spring Motion
Integrator Quality

1. Stability
2. Performance
3. Accuracy
Stability

- Extrapolation
- Interpolation
- Mixed
- Energy
Performance

- Derivative evaluations
- Matrix Inversion
- Nonlinear equations
- Step-size limitation
Accuracy

- Accuracy is measured using the Taylor Series.

\[ x(t + h) = x(t) + x'(t)h + \frac{1}{2} x''(t)h^2 + \cdots \]
Accuracy

- First-order accuracy is usually sufficient for games.
- You can safely ignore RK4, BDF, Midpoint, Predictor-Corrector, etc.
- Accuracy $\neq$ Stability
Further Reading & Sample Code

- Hairer, Geometric Numerical Integration