Differential Equation Basics

Andrew Witkin

Carnegie Mellon University
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.

\[ x(t + \Delta t) = x(t) + \Delta t \, f(x,t) \]
Problem I: Inaccuracy

Error turns $x(t)$ from a circle into the spiral of your choice.
Problem II: Instability to Neptune!
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
\[ x(t + \Delta t) = x(t) + \Delta t \, f_{\text{mid}} \]
More methods…

• Euler's method is *1st Order*.
• The midpoint method is *2nd Order*.
• Just the tip of the iceberg. See *Numerical Recipes* for more.
• Helpful hints:
  – *Don't* use Euler's method (you will anyway.)
  – *Do* Use adaptive step size.
Modular Implementation

• Generic operations:
  – Get 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

- Dim(state)
- Get/Set State
- Deriv Eval
A Code Fragment

```c
void euler_step(sys,h){
    float time;
    get_state(sys,temp1, &time);
    deriv_eval(sys,temp2);
    vtimes(h,temp2);
    vtimes(h,temp2);
    vadd(temp2,temp1); /*
    set_state(sys,temp1,time + h);
}
```