Runge-Kutta Integration

CS-116B: Computer Graphics Algorithms
Spring 2018

Runge-Kutta Integration

Runge-Kutta: a numerical integration scheme.

Fourth degree Runge-Kutta integration:

$a_1 = v^t$
$b_1 = v^t + (\Delta t / 2) * a_2$
$c_1 = v^t + (\Delta t / 2) * b_2$
$d_1 = v^t + \Delta t * c_2$
$x^{t+1} = x^t + \Delta t / 6 * (a_1 + 2b_1 + 2c_1 + d_1)$

$a_2 = f(x^t, v^t) / m$
$b_2 = f(x^t + \Delta t / 2 * a_1, v^t + \Delta t / 2 * a_2) / m$
$c_2 = f(x^t + \Delta t / 2 * b_1, v^t + \Delta t / 2 * b_2) / m$
$d_2 = f(x^t + \Delta t * c_1, v^t + \Delta t * c_2) / m$
$v^{t+1} = v^t + \Delta t / 6 * (a_2 + 2b_2 + 2c_2 + d_2)$

Source: *Real Time Physics Class Notes*, p. 14.

Runge-Kutta Integration: stability

The Runge-Kutta integration scheme is *conditionally stable*.

Conditionally stable means “there is a certain range for the time step Δt size for which the simulation is stable.”

To be conditionally stable, you must adjust the spring ***k*** value.

Notes:

“The stiffer the springs, the smaller the time step required to keep the simulation stable.”

“In real-time situation, e.g. in a computer game, it is essential that an integration is unconditionally stable meaning stable in all circumstances and for the time step size given by the required frame rate.”

Source: *Real Time Physics Class Notes*, p.15.

Runge-Kutta Integration: code

```
void StepSimulation (float dt)
{
  float F;  // total force
  float A;  // acceleration
  float Vnew;  // new velocity at time t + dt
  float Snew;  // new position at time t + dt
  float k1, k2, k3, k4;

  F = (T - (C * V));
  A = F / M;
  k1 = dt * A;
  F = (T - (C * (V + k1 / 2)));
  A = F / M;
  k2 = dt * A;
  F = (T - (C * (V + k2 / 2)));
  A = F / M;
  k3 = dt * A;
  F = (T - (C * (V + k3)));
  A = F / M;
  k4 = dt * A;
// Calculate the new velocity at time t + dt
// where V is the velocity at time t
  Vnew = V + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
// Calculate the new displacement at time t + dt
// where S is the displacement at time t
  Snew = S + Vnew * dt;
// Update old velocity and displacement with the new ones
  V = Vnew;
  S = Snew;
}
```

Source: *Physics for Game Developers*, pp.155-156.

Runge-Kutta Integration: code

```
// Global variables
  float T;  // thrust
  float C;  // drag coefficient
  float V;  // velocity
  float M;  // mass
  float S;  // displacement
```

Source: *Physics for Game Developers*, pp.149