ODE's & Particle Dynamics

Duotun Wang

September 21, 2019

1 Problem A: Ballistic Motion

Given a location of a gun at (0,0,0), write an 3D artillery simulator that can take in the mass of the projectile, amount of powder, the azimuth and elevation of the gun barrel. Use the amount of powder and the mass of the projectile to determine the muzzle velocity. Account for gravity and air friction. Assume that one kilogram of powder produces 10,000 newtons of force. Assume instantaneous acceleration as a result of the powder going off. Air friction coefficient is constant. Set it to be 50 kg/s initially. The gun and target are both on the X-Z plane.

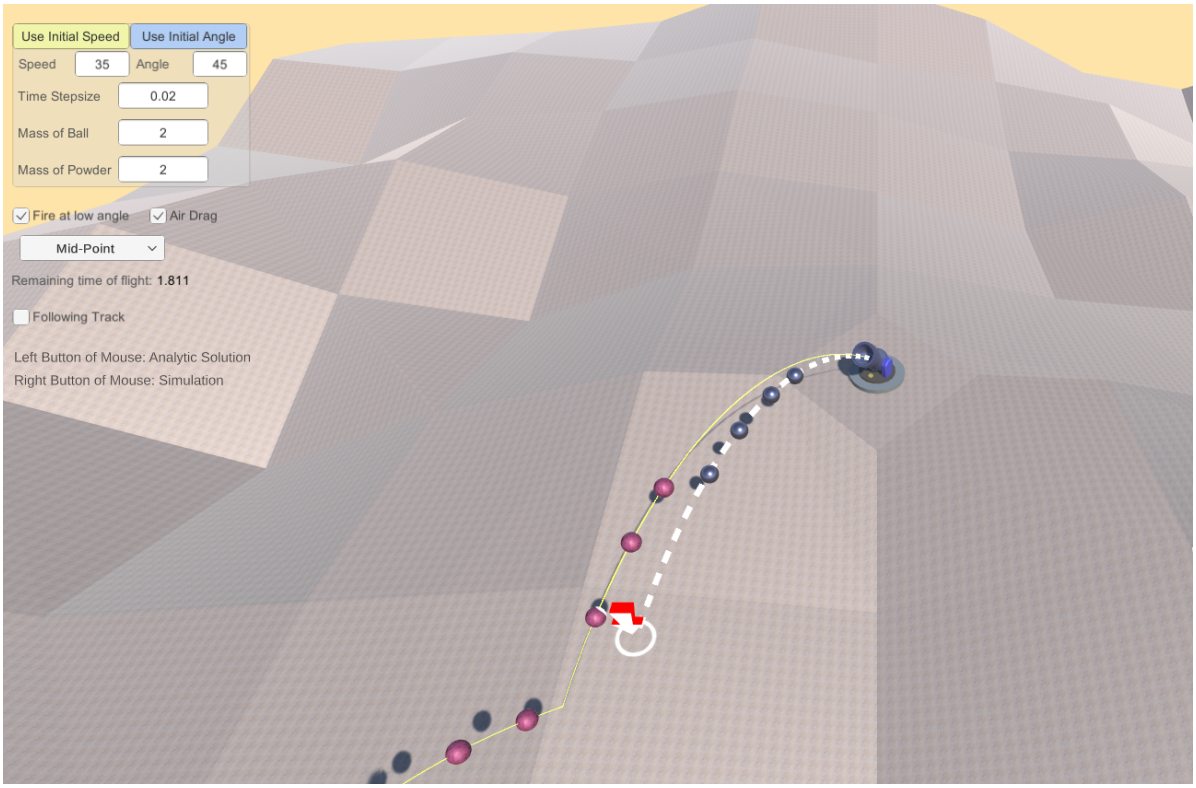


Figure 1: Scene: Cannon and Control Panel

1.1 Methods

I have implemented the following integeration methods, the same case for Problem B:

- Back Euler Method

  $v(t + \Delta t) = v(t) + a(t) * \Delta t$

  $x(t + \Delta t) = x(t) + v(t + \Delta t) * \Delta t$

- Midpoint

  $v(t + \Delta t) = v(t) + a(t + 0.5 * \Delta t) * \Delta t$

  $x(t + \Delta t) = x(t) + v(t + 0.5 * \Delta t) * \Delta t$

- Runge-Kutta Forth Order (RK4)

  $x_1 = \Delta t * v(t)$

  $x_2 = \Delta t * v(t + 0.5 * \Delta, x_1)$

  $x_3 = \Delta t * v(t + 0.5 * \Delta, x_2)$

  $x_4 = \Delta t * v(t + \Delta, x_3)$

  $x(t + \Delta t) = x(t) + \frac{1}{6} * x_1 + \frac{1}{3} * x_2 + \frac{1}{3} * x_3 + \frac{1}{6} * x_4$

  $v_1 = \Delta t * a(t)$

  $v_2 = \Delta t * a(t + 0.5 * \Delta, a_1)$

  $v_3 = \Delta t * a(t + 0.5 * \Delta, a_2)$

  $v_4 = \Delta t * a(t + \Delta, a_3)$

  $v(t + \Delta t) = v(t) + \frac{1}{6} * a_1 + \frac{1}{3} * a_2 + \frac{1}{3} * a_3 + \frac{1}{6} * a_4$

1.2 Running System

The whole simulation system is run on Unity Engine 2019.2.4f1, Windows 10 version. Nvidia Geforce 1660 Ti is utilized for rendering and the fps is fixed around 90. Matlab R2019b is used for further simulation analysis.

1.3 Analysis

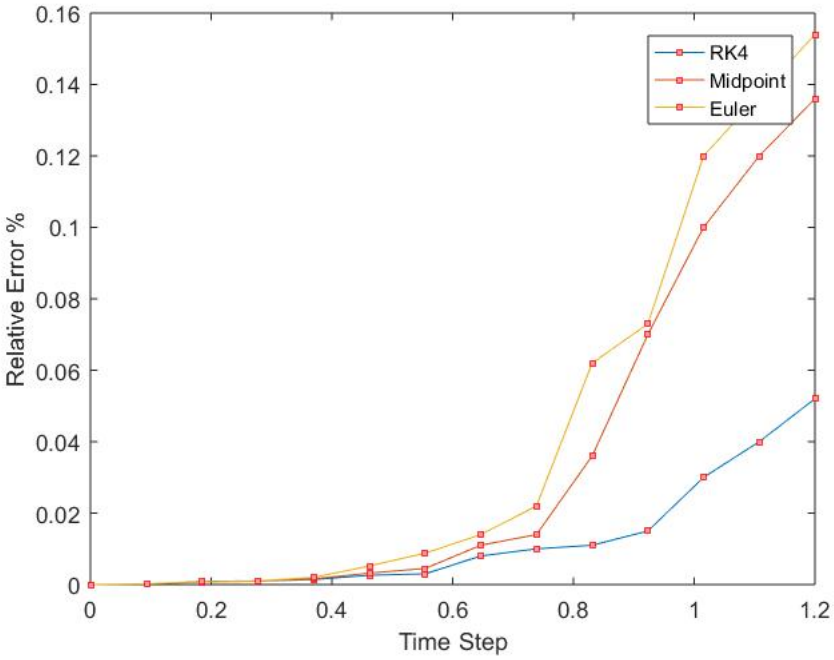
To compare integration methods for this canon simulation, not only I implemented some things to compare them directly on Unity, I also recorded the simulated positions to analyze simulation error in a more precise way. I considered the condition without air friction for simplicity, which means we can compare our simulated results with corresponding analytic solutions. As for interface part, we can freely change parameters like mass of ball, mass of powder, time step, and air drag to control the simulated results of the canon scene.

1.3.1 Accuracy

I measured relative error for all three integration methods to analyze accuracy. I manually created a projecting scene with $air_friction = 0$, $mass\_ball = 2kg$, $amount\_of\_powder = 2kg$, $force\_per\_kg\_powder = 10000N$, $contact_time = 2e^{-3}$ sec, $angle = 45°$. Every method will perform a whole projection motion and all the position information of thrown ball will be recorded until the ball first contacts with ground.

Averaged Relative Distance error %	Euler	Midpoint	RK4
$\Delta t = 0.01$	0.012	0.009	0.007
$\Delta t = 0.05$	0.2	0.042	0.03
$\Delta t = 0.1$	1.2	0.42	0.07

Table 1: Accuracy Analysis



1.3.2 Efficiency

I measured computation time for all three integration methods to analyze efficiency. I manually created a projecting scene with $air_friction = 0$, $mass\_ball = 2kg$, $amount\_of\_powder = 2kg$, $force\_per\_kg\_powder = 10000N$, $contact_time = 2e^{-3}$ sec, $angle = 45°$. Every method will perform 500 Δt and after that, corresponding computation time of three methods are compared in the following table.

From the table, it is obvious to find that the euler method is the most efficient method. We can also obtain the similar conclusion in a theoretical way: midpoint and RK4 in fact split the step size into smaller one, which will inversely add iteration times.

Computation Time (μs)	Euler	Midpoint	RK4
$\Delta t = 0.02$	7	16	28
$\Delta t = 0.05$	13	27	41
$\Delta t = 0.5$	18	38	57

Table 2: Efficiency Analysis

1.3.3 Stability

For stability, I directly test three methods with large time step On Unity. When the time step is set to 2 or more, the euler method cannot simulte the projectile motion. Actually, the ball cannot even move out of the turret. The midpoint method explodes out at the time step of around 3.3 and the RK4 method explodes out at the time step of around 4. Thus, from the above simple testing, the RK4 method is the most stable.

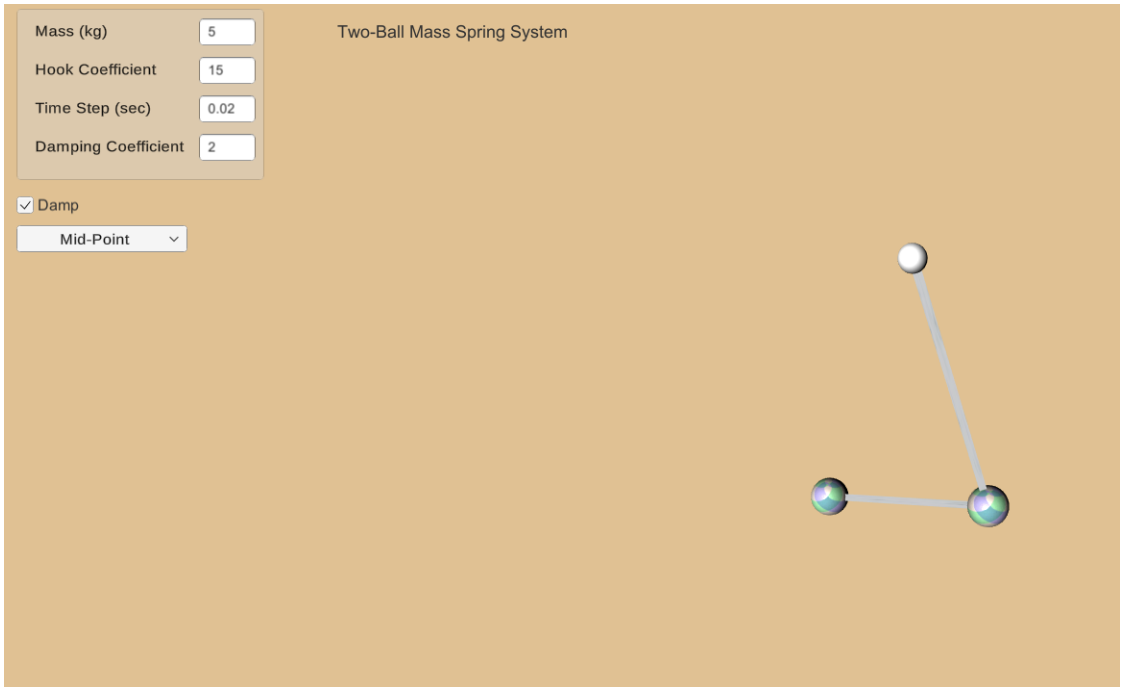


Figure 2: Scene: Spring-Mass Simulator of Two Balls

2 Problem B: Spring Mass Simulator

A spring hands vertically in its equilibrium or resting position. Given a user-defined mass m attached to the spring with the spring constant k, not stretched at first. Simulate the motion of the spring and mass under the effects of spring and gravitational forces. Assume the mass is 5 kg *and* $k = 15\ kg/s^2$. Then, set the mass to be 10 kg and $k = 20kg/s^2$.

2.1 Analysis

To compare iterative methods for this spring-mass simulation, I designed proper interfaces to make sure that I can compare these three methods in damped methods or not. Actually, for simplicity, I also designed a very simple scene with one spring, one ball to analyse accuracy of three integration methods.

2.1.1 Accuracy

I measured relative error for all three integration methods to analyze accuracy in the undamped condition. I manually create a spring-simulating scene with $damping = 0$, $mass\_ball = 5kg$, $hook\_coefficient = 15$.. Information from the following table apparently indicates that the RK4 method should be the most accurate integration method.

Averaged Relative Distance error %	Euler	Midpoint	RK4
$\Delta t = 0.01$	0.007	0.002	0.0008
$\Delta t = 0.05$	0.01	0.004	0.002
$\Delta t = 0.1$	0.08	0.022	0.0105

Table 3: Accuracy Analysis

2.1.2 Efficiency

I utlized the same idea of problem A to analyze efficiency in this problem. I manually create a spring-simulating scene with $damping = 0$, $mass\_ball = 5kg$, $hook\_coefficient = 15$. Every method will perform 500 Δt and after that, corresponding computation time of three methods are compared in the following table.

Computation Time (μs)	Euler	Midpoint	RK4
$\Delta t = 0.02$	5	12	23
$\Delta t = 0.05$	11	24	43
$\Delta t = 0.5$	16	35	59

Table 4: Efficiency Analysis

2.1.3 Stability

For stability, theoretically, explicit euler method is always unstable because it lack in enough considerations for potentially future states. During experiments, I directly tested three methods with large time step. When the $\delta t = 0.5$, the euler method blows up. When the $\delta t = 0.71$, the midpoint method blows up. When the $deltat = 0.73$, the RK4 method blows up. Thus, the RK4 method is more stable that the other two methods. Actually, if hook constant (k) is big, step size (h) must be small. When $h > 1/2k$, things explode.