Project Title: Comparison of Multiple Solvers

Yusuke Niiro

1） Numerical Solvers Implemented

1. Euler Method

   - Computing Derivatives: 1 time

   → Computation time should be shorter than Midpoint/ Heun/ Runge-Kutta Method. But should have more errors than these three methods.

2. Midpoint Method

   - Computing Derivatives: 2 times

   → Computation time should be longer than Euler Method, but shorter than Runge-Kutta Method. Additionally, it should be more realistic than Euler Method.

3. Heun Method

   - Computing Derivatives: 2 times

   → Same expectation as Midpoint Method.

4. Runge-Kutta Method

   - Computing Derivatives: 4 times

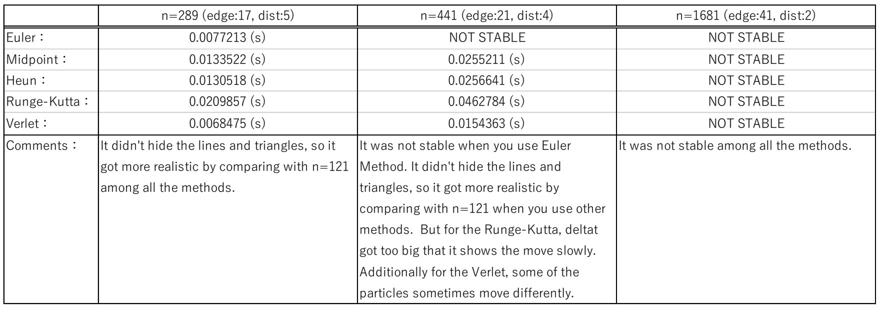
   → Computation time should be longer than Midpoint/ Heun/ Runge-Kutta Method. But should have less errors than these methods.

5. Verlet Method

   → Only this method renders the collision by location difference by comparing other methods are using velocity reflection. But still it computes the derivatives once so computation should not be that long.

2） Benchmarks

	n=9 (edge:3, dist:40)	n=25 (edge:5, dist:20)	n=81 (edge:9, dist:10)	n=121 (edge:11, dist:8)
Euler：	0.0000255 (s)	0.0001036 (s)	0.0007382 (s)	0.0016726 (s)
Midpoint：	0.000036 (s)	0.0001546 (s)	0.0011994 (s)	0.0027969 (s)
Heun：	0.0000388 (s)	0.0001581 (s)	0.0011948 (s)	0.0028224 (s)
Runge-Kutta：	0.0000605 (s)	0.0002986 (s)	0.0020831 (s)	0.0049068 (s)
Verlet：	0.0000281 (s)	0.000105 (s)	0.0007688 (s)	0.0018105 (s)
Comments：	It hides the lines and triangles, so it was less realistic among all the methods.	It hides the lines and triangles, so it was less realistic among all the methods.	It didn't hide the lines and triangles, so it got more realistic among all the methods.	It didn't hide the lines and triangles, so it got more realistic by comparing with n=81 among all the methods.



	n=289 (edge:17, dist:5)	n=441 (edge:21, dist:4)	n=1681 (edge:41, dist:2)
Euler：	0.0077213 (s)	NOT STABLE	NOT STABLE
Midpoint：	0.0133522 (s)	0.0255211 (s)	NOT STABLE
Heun：	0.0130518 (s)	0.0256641 (s)	NOT STABLE
Runge-Kutta：	0.0209857 (s)	0.0462784 (s)	NOT STABLE
Verlet：	0.0068475 (s)	0.0154363 (s)	NOT STABLE
Comments：	It didn't hide the lines and triangles, so it got more realistic by comparing with n=121 among all the methods.	It was not stable when you use Euler Method. It didn't hide the lines and triangles, so it got more realistic by comparing with n=121 when you use other methods. But for the Runge-Kutta, deltat got too big that it shows the move slowly. Additionally for the Verlet, some of the particles sometimes move differently.	It was not stable among all the methods.

Table 1. Computation Time Comparison

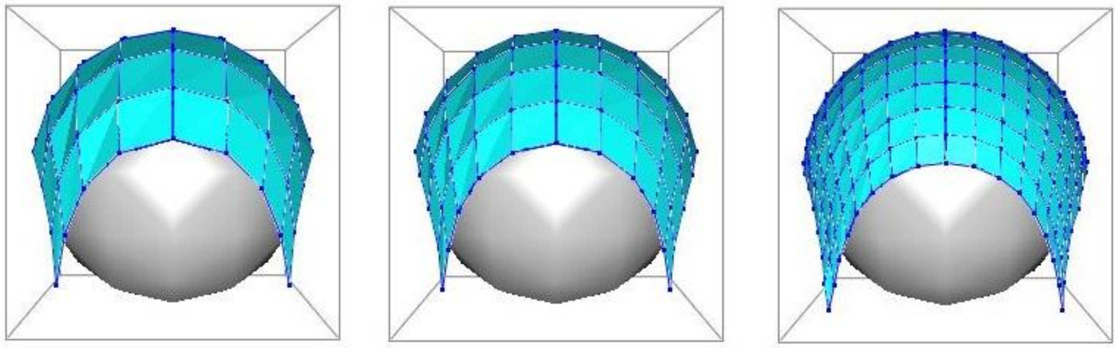


Image 1. Left Image is made by n=81 (edge:9, dist:10), Center Image is made by n=121 (edge:11, dist:8), Right Image is made n=289 (edge:17, dist:5). ("n" is the number of particles.)

3) Conclusion

By looking at the computation time result, computation time showed up like I expected. Shortest computation time was from Euler and Verlet Method with only computes the derivatives only once and longest was from Runge-Kutta which computes the derivatives 4 times. So, this means that if there are more times to compute the derivatives, computation time will get longer. But, if the computation time get longer, it means that update will be slower, so it is important to see the balance. Additionally, if you look when condition is n=441, we notice that error of particles had influence the showing and make it unstable. So that is why Euler method was not unstable and other methods were stable. For the condition n=1681, reason why it was not stable among all the methods, because the deltat and computation time got long and it was difficult to get closer to the correct motion.