國立清華大學
NATIONAL TSING HUA UNIVERSITY

11110PHYS401300
Computational Physics Lab
計算物理實作

Release date: 2022.11.21
Due: 2022.12.05
(submit to google classroom)

Homework 3

Reading Assignments

1. Inverse square root is essential in many physical process (such as gravitational acceleration). In video games, the performance is more important than the accuracy. Read the article "Fast inverse square root" http://www.lomont.org/papers/2003/InvSqrt.pdf

2. Direct N-body simulation is expensive, in which the computation needed increases as N^2. Barnes & Hut (1986) proposed a hierarchical $O(N \log N)$ force-calculation algorithm to speedup the acceleration calculations. Read the paper here https://ui.adsabs.harvard.edu/#abs/1986Natur.324..446B/abstract.

Programming Assignments

1. Create a cloud of particles ($N = 10^5$) distributed by a normal distribution (mean is zero and variance is one) in 3D Cartesian coordinates. Assuming the initial particle velocities and accelerations are distributed by the same normal distribution and the total mass of the system is 20. Run your N body simulation from $t = 0$ up to $t = 10$ with a constant time step $\Delta t = 0.01$. Use a soften length $r_{\rm soft} = 0.01$. Make snapshots of your particles projected on the $x - y$ plane at $t = 0, 2, 4, 6, 8, 10$. Make sure you have modified the particle size and the opacity to make it clear on the plots.

2. During the lectures, we only talked about Runge-Kutta methods. There is another popular method for solving initial value problem, called the Leap-frog scheme (or the "kick-drift-kick" scheme). For each time step Δt, each particle receives a half-step kick,

$$\boldsymbol{v}_i = \boldsymbol{v}_i + \frac{\Delta t}{2} \times \boldsymbol{a}_i, \tag{1}$$

followed by a full step drift,

$$\boldsymbol{x}_i = \boldsymbol{x}_i + \Delta t \times \boldsymbol{v}_i, \tag{2}$$

and then followed by another half-step kick. Please implement this algorithm in your `nbody.py` solver. Repeat the first problem with the same initial condition (use the same random number seed) and use this Leap-frog method.

3. In your `nbody.py` solver, (a) write a function in the `Particles` class to compute the total kinetic energy and the potential energy of the particles. (b) modify the `output` function in the `Particles` class to output the kinetic energy and the potential energy. (c) Repeat the first problem with the same initial condition and make plots of total kinetic energy, potential energy, and total energy as functions of time. Please compare methods with Euler, RK2, RK4, and Leapfrog. (d) What is the order of accuracy of this Leapfrog method?