Science, Engineering and Social Science Series
ISSN/e-ISSN: 2541 – 0369/2613 – 988X
Vol. 4, No. 2, 2020, Printed in the Indonesia

Performance Comparison between Direct Method and Tree-code used in N-body Simulation using Python

Muhammad Isnaenda Ikhsan[1*]
[1]*Departement of Science, Institut Teknologi Sumatera*

N-body simulation is a great tool to study the dynamics of many body system in astronomy. There a many method that could be used to perform N-body simulation. In this study, we compare two methods such as a direct method and tree-code to perform N-body simulation code over Python software. The advantages of direct method it's very straightforward and could be very accurate while the disadvantages are this method could take a lot of computational resources. In other hand, tree-code could perform much better in term of computational time, but lack of accuracy and limited to certain cases application. The result shows that performing N-body simulation using both method is very possible to be done in a computer with modest specification. The direct method and tree code perform similarly in small number particles (N<50) case, but tree-code is much faster as the number of particles increase (N>100). For N = 500, tree-code is 150% faster than direct method and for N=1000, tree-code is 200% faster than direct method. In term of energy conservation, both methods perform well and similar.

Keywords: N-body, direct method, tree-code, python.

1. INTRODUCTION

Gravitational N-body simulations is a numerical solution of N particles motions equations for interacting gravitationally, are widely used tools in astrophysics, with applications from few body or solar system like systems all the way up to galactic and cosmological scales. Here, this method could be applied to almost all astronomical system, N-body simulation have many kinds of variations. Different kind of simulation methods is developed depend on the dynamical system which will be studied.

The idea of simulating the dynamic of N-body system started even before digital computer invented. Holmberg use N-body simulation concept to study how tidal force act on stars in galaxy encounter event [1]. The lightbulbs were used to represent the stars and calculate the changes of light fluxes to represent the gravity force, because both light fluxes and gravity are inversely proportional to square distance. As digital computer invented, one of the pioneers of N-body simulation is Aarseth [2]. He used N-body simulation to study dense stellar system such as globular cluster and galaxy cluster [3,4,5]. Recently, N-body simulation are widely to simulate the interaction of massive objects e.g. black holes and neutron stars. Arca-Sedda uses N-body simulations to observe the dynamic evolution of multiple black holes in the area around supermassive black holes [6]. In the cases that involving massive objects such as black holes, a post-Newtonian approximation is needed, which adds relativistic equation in its gravity equation.

The simplest and most accurate N-object simulation method is using the direct integration method [7]. However, this method does not suitable for the all N-body system. For a system of particles with small N, this method works well and fast, but as the value of N increase, the computer resources used will be increase significantly. Therefore, many other alternative methods

[*]*Email Address: isnaenda.ikhsan@staff.itera.ac.id*

are developed that maybe are not as accurate as the direct method but in terms of resource efficiency is far better than the direct method. One of the alternative methods are widely to use the tree-code method. Tree-code method is better and efficient in terms of computational resource used by the program compared to the direct method because the computational operations required by the tree-code method are far less than the direct method for the same number of N objects [8].

Apart from the simulation methods, the selection of an integrator for this simulation is also important in carrying out N-object simulations. In the case of a system that has a lot of N objects, it usually does not require a high level of accuracy, so that it can use an integrator with intermediate accuracy but fast in term of calculation. Whereas in cases such as solar system, very high accuracy is needed. In this domain, so called celestial mechanics domain, very high accuracy is required to correctly evaluate the perturbative terms and to avoid being dominated by numerical noise such as time discretization and round-offs errors

The purpose of this study is to show the performance of N-body simulation that can be done on modern personal computer using Python programming language. For the sake of performance, N-body simulation code usually written in Fortran or C, here we try to do the simulation using Python to see how well the simulation could be done.

The second part of this articles is to explain the simulation methods, the third part is to show the result and discussion, and the last part is conclusion.

2. METHODOLOGY

A. Direct Method and Tree-code Methods

The main problem of N-body simulation is how we calculate the acceleration on each particle due to gravity of other particles. Acceleration on each particle could be described using Newtonian gravitational equation of motion:

$$\ddot{\boldsymbol{r}}_i = -G \sum_{j=1, j\neq i}^{N} m_j \frac{r_i - r_j}{|r_i - r_j|^3}, \quad \text{..... (1)}$$

where dots denote differentiation with respect of time, G is gravitational constant, N is total number of particles, m is mass of each particle, and r is the distance between particles. In this simulation we put G = 1 and express other parameters such as mass, distance, velocity, and acceleration using arbitrary scaled units [9]. If we know the initial position and velocity for each particle, there are exist unique solution for eq. (1) to determine position and velocity of particle at certain time. First, we use Direct integration Method using Hermite scheme integrator to perform N-body simulation [10]. This method is pretty straightforward, gravity is calculated directly from eq. (1). Then by using Hermite integrator we could find numerical solution of velocity and position for each particle at certain time-step. By doing this over and over, we could get the velocity and position of the particle from initial time to final time for every time-step. The advantages of this method are that it is very accurate for system that have equal mass. The accuracy could be controlled easily by increase or decrease the time-step we took. To put simply, if we want the simulation to have high accuracy, we have to decrease the time-step as small as possible, and vice versa. The problem is this method is very resource demanding because for every iteration or every time-step, the program needs to make calculation in the order of $O(N^2)$. That mean for every N particle, the program needs to calculate N^2 times for every time-step. When we want to make high accuracy simulation, we need much more computational resource and it will take much longer to simulate the system. But we could sacrifice the accuracy for faster simulation time, the simulation would still represent the physics of the system at certain degrees but not as accurate.

The second simulation is done by using Tree-code method. Tree-code method is more efficient in term of computation time, because it only needs to do calculation in the order of $O\ (N \log N)$ per time-step which is smaller than the direct method. The downside of this method is that this method is an approximation so there will always be small force errors. This method also not great to simulate collisional system where close encounters is important. This method could provide fast calculation because force from very distant particles does not need to be calculated with very high accuracy.

B. Initial Condition and Simulation

For both methods we use same initial condition for the position and velocity of particles. Plummer distribution is used to generate position and velocity [11]. This distribution usually used to represent spherical distribution of globular star cluster of ellipse galaxy. Plummer distribution is used because it simple to code and still represent the physic of real system.

For the simulations we did two kinds of simulation. First simulation is done to observe the computation time increase as the number of particles increased. Table I show the parameter that used in the first simulation. In the simulation 1 (sim-1), the program will simulate the system with N ranged from N=2 to N=50, where each run the program will do 1000 iteration. We did the same with simulation 2 (sim-2) and 3 (sim-3) but with different N and number of iterations.

Table I. Parameters for the first simulation

Parameters	Sim-1	Sim-2	Sim-3
Number of particles (N)	50	500	1000
Number of iterations	1000	100	20

The second simulation is to inspect the progression of total energy of the system from both methods. For this purpose, we chose the parameters as showed in Table II. The parameters are chosen because it's not too big or small system and still represent the dynamic of N-body system. To do this, for every iteration of the simulation the program will calculate the total energy of the system which is kinetics energy and potential energy. Then it will compare the total energy of the system with the initial energy, this way we get the energy loss of the system for every iteration. As simulation progress the energy loss will always be bigger after each time-step, so we want to see how this error behave on direct methods and tree-code method. This simulation is performed in personal computer with high specification (Windows 10, Intel i5 4200U 1.6 ~ 2.3 GHz, RAM 8GB, GPU Nvidia GeForce 720M). All simulation is executed in roughly under 6 hours computing time.

Table II. Parameters for the second simulation

Parameters	Value
Number of particles (N)	250
Time-step (dt)	0.5
End Time (t_end)	100

3. RESULTS AND DISCUSSION

For the first simulation, we run the simulation three times with different N for each run. The initial condition for all simulation is using Plummer distribution. For the simulation 1 (Sim-1) The program will calculate the simulation from N=2 and after each time-step it will increase N by one until it reaches N=50. For each N value the program did 1000 iterations. Figure 1 showed the calculation time for every N for both direct and tree-code method. We did the same run for Sim-2 (Figure 2) and Sim-3 (Figure 3) but with different value of maximum N and the number of iteration (see Table 1). It is obvious on all three simulation that the computation time will increase as the number of particles increased for both methods. The difference between direct method and tree-code is that the computation time on direct method will follow quadratic curve, so it will be increase more rapidly than the tree-code method, which follow logarithmic curve. Except, in Sim-1 where the value N is still relatively small, both direct and tree-code methods performance is almost the same, in fact the direct method performance is a little bit better and consistent. In this domain, where the number of N is still small, the tree-code method cannot take advantages of its method yet because the particles is not too clumpy and the distance for all particles is still relatively the same. The tree-code method will take advantages of its method when there are many particles clumping together, then it can be treated as one particle by distant particle to reduce the calculation operation. That is why in Sim-2 and Sim-3 it clearly shown that as N increase the tree-code performs better than the direct method. Because when the N become large, particles tend to get close to each other due to gravity. It is necessary to mention that in Sim-2 and Sim-3 the computation time is almost the same even though the Sim-3 simulate much larger N which double than Sim-2. This happen due to the number of iterations per N is different for both simulations. In Sim-2 the number of iterations is 100 times, so for every N, it makes 100 times calculation. Then, in Sim-3 it only makes 20 iteration per N. For the illustration for N=500, Sim-2 will make calculation 100×500 which is 50000 calculation per time-step and for Sim-3 it will only make 20×500 calculation which is 10000 calculation per time-step. So, for the same number of N, Sim-2 will make calculation 5 times more than the Sim-3, that is why the Sim-2 will take much more computation time than Sim-3 in the same number of N. If we took the ratio of the maximum computation time for direct method and tree-code, we found that in Sim-2 the ratio is $t_{direct} : t_{tree} = 2.5 : 1$ and in Sim-3 the ratio is $t_{direct} : t_{tree} = 4 : 1$. That mean, in Sim-2 tree-code is 150% faster than direct method, and in Sim-3 tree-code is 200% faster than direct method. This mean as the number of N increase, we see that tree-code method become much faster than direct method or in another word, the direct method become significantly slower.

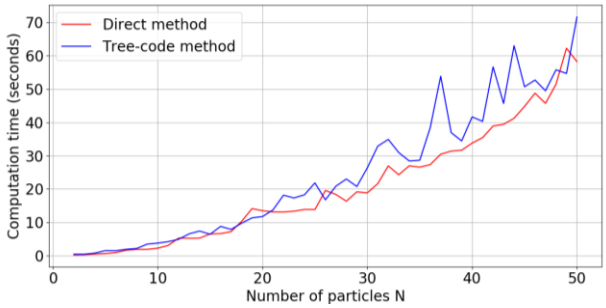


Figure 1. Calculation time for N= 2 to N=50.

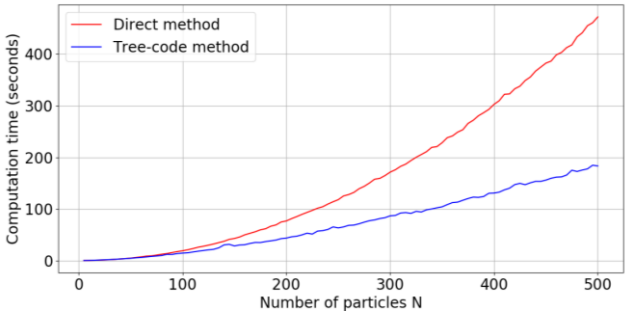


Figure 2. Calculation time for N= 5 to N=500 (Sim-2).

In the next simulation, we did the simulation using parameters shown in Table II. We want to observe how the total energy of the system, kinetic energy and potential energy, change as the system evolving dynamically. We did several simulations using Plummer distributions for initial. The initial is generated randomly for each simulation but still the same for both direct and tree-code method. Figure 4 showed how the total energy change on the system on each simulation. In all figures, it appears that in direct method simulation, the total energy difference changes rapidly in the early part of the simulation. But this feature did not appear in tree-code simulation. This could happen because energy spike like this usually appear due to close encounter that happen in the system. When we use direct method, it will calculate all the close encounter that happen in the system, but in tree-code this close encounter is not treated well and calculated only using approximation. That is why in almost all simulation the spikes-like feature appeared in direct method simulation but not in the tree-code simulation. In this these three simulations, the direct method and tree-code perform relatively similar in term of energy conservation. There is one case where the direct method performs better (see Figure 4 (a)) and also one case where tree-code is better (see Figure 4(c)). But generally, the performance from both methods is the same, in all simulation that has been done it show that neither method outperform each other significantly.

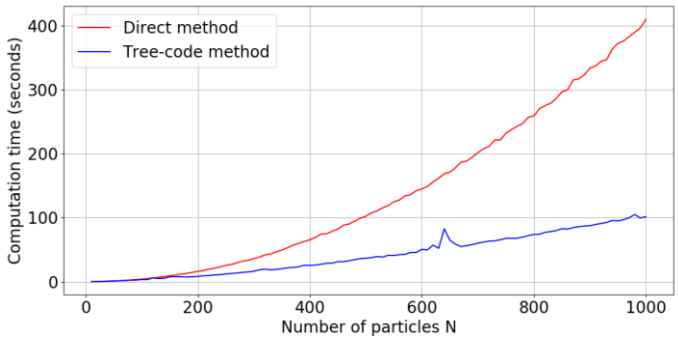
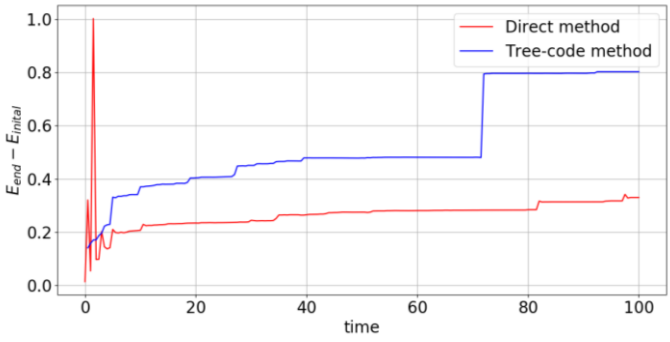
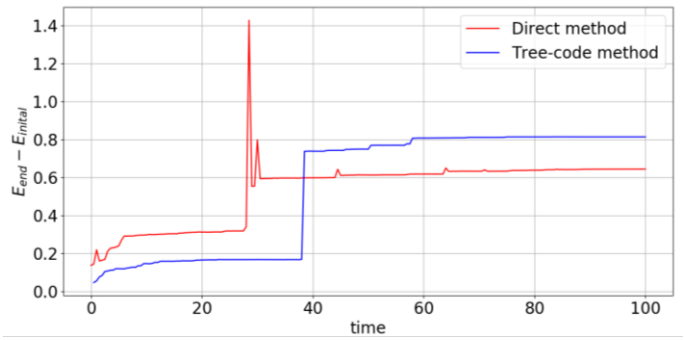


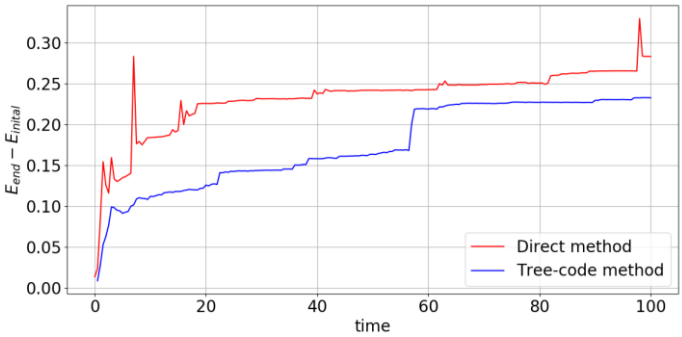
Figure 3. Calculation time for N= 10 to N=1000 (Sim-3).



(a)



(b)



(c)

Figure 4. Total energy change of the system for three simulation. (a) Simulation where total energy change in direct method is smaller than tree-code, (b) total energy change in both methods is almost equal, and (c) total energy change in direct method is bigger than tree-code.

4. CONCLUSIONS

Performing simple N-body simulation using direct method and tree-code in Python is very feasible to be done. The biggest different between direct and tree-code method is the time it spends on simulation. If our particle is not that much, direct method is the choice because its accuracy. But if we want fast calculation on many particles system and willing to sacrifice some accuracy, then the tree-code is wise choice. However, both methods perform similar in term of energy conservation. So, this kind of simple simulation could be used to give us more physical insight about gravitational system rather than to be used to accurately simulating the dynamic of real N-body system.

References

1. Holmberg, Eric. **(1941)**. *On* The Clustering Tendencies Among The Nebulae. The Astrophysical Journal, Vol. 94, No. 3, p.285.
2. Aarseth, Sverre J. **(1963).** Dynamical evolution of clusters of galaxies, I. Monthly Notices of the Royal Astronomical Society, Vol. 126, p.223.
3. Aarseth, Sverre J. **(1966).** Dynamical evolution of clusters of galaxies, II. Monthly Notices of the Royal Astronomical Society, Vol. 132, p.35-65.
4. Aarseth, Sverre J. **(1973).** Computer Simulations of Star Cluster Dynamics. Vistas in Astronomy, vol. 15, Issue 1, pp.13-37.

5. Aarseth, Sverre J. **(1974).** Dynamical evolution of simulated star clusters. I. Isolated models. Astronomy and Astrophysics, vol. 35, no. 2, p. 237-250.
6. Arca Sedda, Manuel. **(2020).** Dissecting the properties of neutron star-black hole mergers originating in dense star clusters. Communications Physics, Volume 3, Issue 1(43).
7. Aarseth, Sverre J. **(1971).** Direct Integration Methods of the N-Body Problem. Astrophysics and Space Science, Volume 14, Issue 1, pp.118-132.
8. Barnes, Josh & Hut, Piet. **(1986).** A hierarchical O(N log N) force-calculation algorithm. Nature, Volume 324, Issue 6096, pp. 446-449.
9. Heggiew, D. C., Mathieu, R. D. **(1986).** Standardised Units and Time Scales. Lecture Notes in Physics, Vol. 267, edited by P. Hut and S. McMillan. Springer-Verlag, Berlin Heidelberg New York, p.233.
10. Makino, Junichiro; Aarseth, Sverre J. **(1992).** On a Hermite Integrator with Ahmad-Cohen Scheme for Gravitational Many-Body Problems. Publications of the Astronomical Society of Japan, v.44, p.141-151.
11. Plummer, H. C. **(1911).** On the problem of distribution in globular star clusters. Monthly Notices of the Royal Astronomical Society, Vol. 71, p.460-470

Received: 24 February 2020, Accepted: 01 May 2020