Computational Astrophysics, Spring 2019
Third Assignment
Hybrid Star Cluster Simulation

Team "The Quasists"
Zorry Belcheva Ioannis Politoupolos Benjamin Gilliam

March 20, 2019

1 The problem: Simulating a cluster of thousands of Stars

1.1 The system: details

Star clusters contain anywhere from hundreds to millions of stars, each one interacting gravitationally with all the rest. These systems can be modeled directly using a direct N-body method, but this is often prohibitively expensive because the number of interactions to be calculated is proportional to N^2. In this exercise, we model the behavior of a cluster of 10^4 stars over 10 million years. The challenge we are faced with lies in balancing precision and computation time. Because our time is limited, we decide to break up the problem into more manageable parts. We want the speed of a tree-code and the precision of an N-body simulation, so we combine the two using AMUSE.

1.2 The problem: precision versus speed

In our approach to observe the dynamical evolution of such a star cluster through a simulation, we combine the ph4 direct N-body code and the Barnes-Hut Tree code. Compared to a computationally cumbersome N-body code, the number of interactions between particles to be calculated by the Barnes-Hut Tree-code is reduced to $Nlog(N)$. BHTree divides the simulation volume into cells, treating nearby cells individually, while treating particles in distant cells as a single large particle located at the center of mass of the cell. This reduces the time needed for computation and allows for the simulation of larger particle sets. Here we investigate the effects of combining the two codes by dividing the particles into two separate sets.

2 Code implementation

The `quasists` BitBucket folder for the assignment `assign03` contains the code `cluster.py` used to run the simulation, the folder `results` that contains all graphs, as well as draft files explaining our lines of thought and a `readme.txt` file.

2.1 The stars: creating objects and initial conditions

In 1911, H. C. Plummer developed a potential function which is still used today to model a spherically symmetric distribution of stars in a cluster. The Plummer potential is

$$\Phi_P(r) = -\sqrt{\frac{GM}{r^2 + a^2}}, \tag{1}$$

where a is the Plummer radius, a scale parameter which describes the characteristic size of the cluster core. The Plummer model is an equilibrium solution to Poisson's equation. It represents a reasonable approximation for the observed density profile of star clusters [3]. We will use a Plummer sphere to realize the initial conditions for our simulation.

The AMUSE function that creates a Plummer sphere of bodies gives them equal mass. However, we want them to follow a pre-given mass distribution, so we change the bodies' masses after initialisation, which also changes the

energy of the system and hence drives it out of equilibrium. Therefore, the `scale_to_standard()` function is used to re-virialize the system.

As in the previous assignments, the objects are created to be of the `Particles()` class within the AMUSE framework. The function `cluster()` creates a star cluster of N stars with masses ranging from M_{min} to M_{max}, and a virial radius r. The parameter `imf` is set to `salpeter` by default, and is used to control the initial mass function: the function that describes the distribution of masses. It is often represented as a power law:

$$\frac{dN}{dm} = Am^{\alpha}. \tag{2}$$

For the popular Salpeter (1955) distribution, $\alpha = -2.35$. In the `cluster()` function we implement two other possibilities: the Miller-Scalo (1979) mass distribution, which takes $\alpha = 1$ for $m < 1M_{\odot}$, and a regular power law distribution with a different exponent. The user can change between these three from the command line according to their preferences by using an option parser.

2.2 Combining Direct N-body and Tree codes

The code works as follows. The `hybrid_gravity()` function combines two codes: one for the direct N-body integration (ph4), and one for the Barnes-Hut tree code. Firstly, we split the particles into two groups: those below the given mass cut go to `light()`, and the rest go to `heavy()`. The two gravity codes then 'speak' to each other by using the `bridge` implemented in AMUSE.

2.3 Code summary

Our script `cluster.py` creates a Plummer sphere of N stars with various IMFs: Salpeter, Miller-Scalo and Exponential IMF with default $alpha = -2$. It splits the stars to `light()` and `heavy()` using a mass splitting parameter `m_cut` which can be changed from the command line. The code utilizes a Barnes-Hut tree model to evolve the gravity for `light()` stars and a ph4 N-body model to evolve the gravity for `heavy()` stars. It then bridges the two gravity models to a hybrid `combined_gravity()`. The code includes stellar evolution using the `SeBa()` model implemented in AMUSE. The new masses are channelled to both the `combined_gravity()` for gravity dynamics and the framework to later create an animation of the cluster. It evolves up to a $t_{end} = 10Myr$ in `tg_time_step` for the hybrid gravity, while the stellar evolution takes place every half `tg_time_step`. and channels the new masses. All timesteps are a function of the dynamical timescale of the cluster, `t_dyn_cluster` (see 3.2 Discussion 2). In the iteration, the half-mass radius, the core radius and the relative energy error (both in total and with respect to the previous iteration) are calculated. The code then plots the graphs described in 3.3 Discussion 3. The code can easily be changed to tree-only or N-body-only by altering the scheme parameter to '0' for hybrid (default), '1' for nbody and '2' for tree. The final energy error and wall-clock time are saved to file with the `m_cut` for the final plot of Discussion 3.
For the results, the initial conditions used are:

- $N = 10^4$ stars
- $R_{init} = 3$ pc
- $M_{min} = 0.1M_{\odot},\ M_{max} = 100M_{\odot}$
- `t_end` = 10 Myr
- Mass splitting parameter: `m_cut` $= 10M_{\odot}$
- Gravity timestep = `tg_time_step_frac` [default = 0.1] * t_{dyn}
- Diagonstics Stellar evolution timestep = `tse_time_step_frac` [default = 0.5] * Gravity timestep
- Diagonstics Bridge timestep = Gravity timestep * `bridge_time_step_frac` [default = 0.05]
- Diagonstics Combined Gravity timestep = Bridge timestep
- Barnes-Hut Tree timestep = 0.5 * `model.pararameter.timestep` [default set by AMUSE]
- Barnes-Hut Tree opening angle = `theta` [default = 0.3] (used to control the accuracy of the tree-code)

3 Results and discussion

3.1 Discussion 1: Splitting the particle set in mass

It is reasonable to split the particle set in mass because, although they are few, the most massive objects will have the strongest effect on the dynamics of the system. An alternative splitting strategy could be to split the particles by distance from the center of the system. Because the force of gravity scales as $1/r^2$, the stars which are furthest away from other stars will contribute less to the motion of the other stars. This is less than ideal, because the distance of the stars to the center-of-mass of the system is constantly changing, where as the mass of most of the stars stays relatively constant, even with the addition of a stellar evolution code. Splitting the particle set according to distance from the center-of-mass of the system could be a good idea because of the naturally occuring mass-segregation that occurs in star clusters: lighter stars tend to migrate to wider orbits, while heavier stars tend to migrate slowly toward the center of the cluster. However, this splitting scheme seems like it would only be useful once the cluster has evolved sufficiently to segregate the stars according to their masses. Hence, we conclude that a split in mass is good enough, because the lighter masses will tend to migrate to the outskirts of the cluster anyway, lowering the importance of their gravitational influence.

3.2 Discussion 2: Bridge time step

For a given system of size R, the dynamical time scale, or crossing time, is defined as the time taken for a particle with a typical speed v to cross the system [1]:

$$t_{dyn} = R/v. \tag{3}$$

It can also be expressed in terms of the virial radius of the system R_{vir} and its total mass M_{tot} as

$$t_{dyn} = \sqrt{\frac{R_{vir}^3}{GM}}. \tag{4}$$

where G is the gravitational constant, and factors of order unity are neglected [1]. In our solution we have adapted a bridge time step that is of size 1/20 the gravitational time step, which is 1/10 of the dynamical time scale of the system. t_{dyn} is the typical time over which the cluster would change significantly, and we want to make sure that `combined_gravity()` updates accordingly over time steps that are not too large and will not produce any rapid, and hence unphysical, changes in the motion of the bodies.

3.3 Discussion 3: Hybrid code results

Here we present the outcome of our cluster simulations. Firstly, we ran the code for all 10^4 stars in either the N-body code or the tree code. The results are shown in figures 18 and 17 respectively.

Secondly, we ran the simulation for different mass splitting parameters: $m_{cut} = [2, 5, 8, 10, 20, 30, 40, 50, 60, 70, 80, 90]M_{\odot}$. Figures 4 to 15 show the final energy error, as well as the half-mass and core radii as a function of mass-cut for an opening angle of $\theta = 0.3$ for the respective mass splitting parameters.

Figure 19 shows the wall-clock time and final relative energy error for the different mass splitting parameter values.

3.4 Discussion 4: Discussion of results

As can be seen from the figures, the initial assumption is correct: it is advantageous to split the stars by mass and run a tree code for low mass stars and a direct N-body code for high mass stars. However, the advantage conferred is marginal. Running the simulation on 10,000 stars using the ph4 direct N-body code alone takes roughly six times as long as running the simulation using the BHTree code alone (see figure 19). The energy error for both runs, as well as for the hybrid runs, is approximately the same, however this would not be the case for a longer simulation run, e.g. for `t_end` = 1 Gyr. As expected (see [1]), the ratio of the halfmass radius and the core radius is always larger than one. Also, both measures increase and peak at $\sim$ 4-6 Myr, and then decrease. We cannot say if this is to be expected, as open clusters would gradually shrink and become globular, but our object is quite young, and (as far as we are concerned) there isn't a well established theory of the behaviour of a cluster at such an age - it is entirely possible that, for some period of time, the cluster appears to be 'expanding', as we observe here.

Figure 19 shows the expected behaviour of the wall-clock time as a function of `m_cut`: the code is faster with more bodies being 'light' i.e. gravitationally evolved by the tree code. However, the relative energy error behaves

opposite to what we would expect: first it drops down (for `m_cut` = 0.1 to 5 $M_\odot$), then there is a sharp peak at `m_cut` = 8 $M_\odot$, after which it drops rapidly, peaks again, drops again at `m_cut` = 40 $M_\odot$, and finally increases gradually. This could be due to some code-specific resonant feature that we fail to take into account, like the time-step being incoherent with the proposed mass cut values that show weird behaviour - `m_cut` = [8, 20 and 30] $M_\odot$. This could potentially be fixed by including proper hydrodynamics or a better stellar evolution model.

3.5 Different initial mass functions (IMF)

Figures 20, 23 and 22 show histograms of the mass distribution when the 3 different IMFs. They are used to produce the following graphs, given the same initial conditions, to illustrate the difference using another IMF makes:

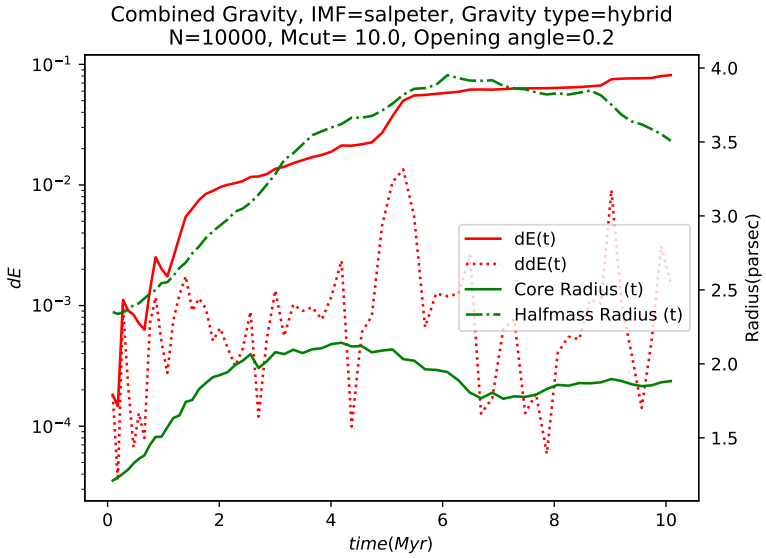

Figure 1: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time using a **Salpeter** IMF.

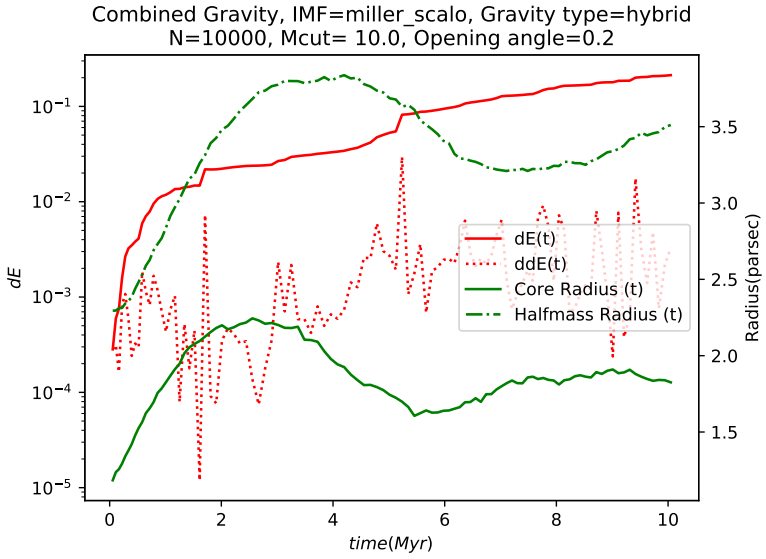

Figure 2: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time using a **Miller-Scalo** IMF.

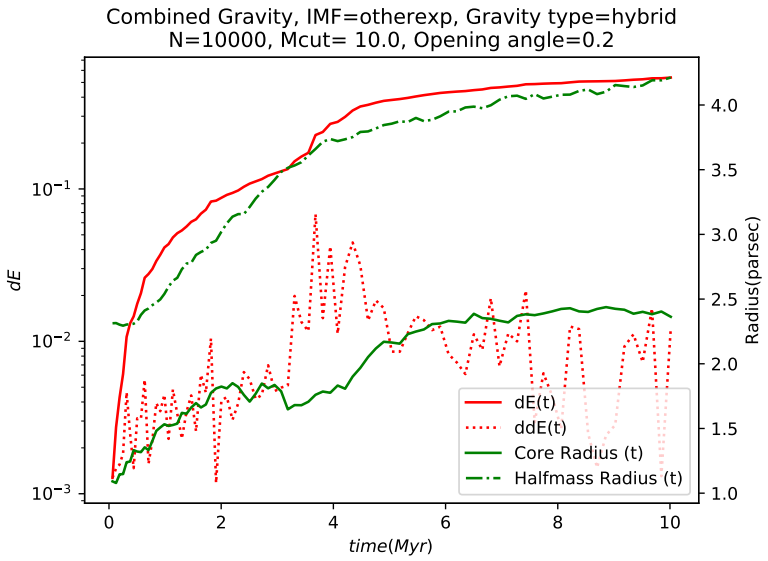

Figure 3: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time using a **different power-law** IMF.

4 Possible improvements

- Integrating to a higher total end time `t_end`: in principle, it would be informative to run the code for, say 1 Gyr. In general, a cluster of age 10 million years would be considered young even for an open cluster (and we cannot talk about it being globular at all here). However, in reality such young objects can hardly be observed due to dust being present around the newly-formed system, obscuring most of the light. Hence, it is difficult to make a conclusion whether the results we are getting here are sensible. On the other hand, if we were to run the code for 1 Gyr, stellar evolution would become of great importance, as well as hydrodynamics (due to stellar wind), with the gas in the cluster playing a large role in its evolution.
- Include hydrodynamics.
- Make an animation of the cluster evolution.
- Investigate the effect of a star going supernova inside the cluster.
- Generally using more accurate (N-body) integrators whenever computational power allows for that.
- Investigate the effect of the mass-splitting parameter at a more discrete (and closely spaced) range around the critical points where the code behaves unexpectedly.
- Even smaller evolving timesteps.
- Investigate the simulation outcomes using both a constant seed value (to use the same Plummer model) and a changing seed value, to see whether they behave as expected at every run.

Bibliography

[1] AMUSE documentation
[2] Simon Portegies Zwart, Jeroen Bédorf (2014) "Computational Gravitational Dynamics with Modern Numerical Accelerators" https://arxiv.org/pdf/1409.5474.pdf
[3] Plummer, H. C. (1911) "On the problem of distribution in globular star clusters", Monthly Notices of the Royal Astronomical Society. vol 71, p 460

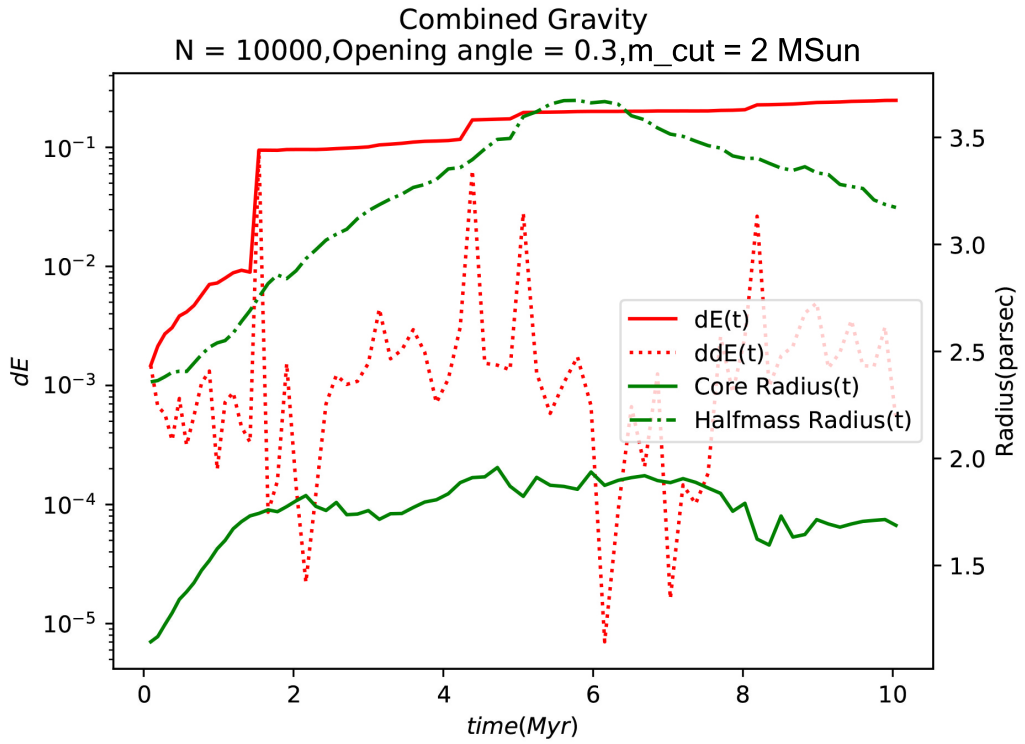

Figure 4: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 2M_{\odot}$.

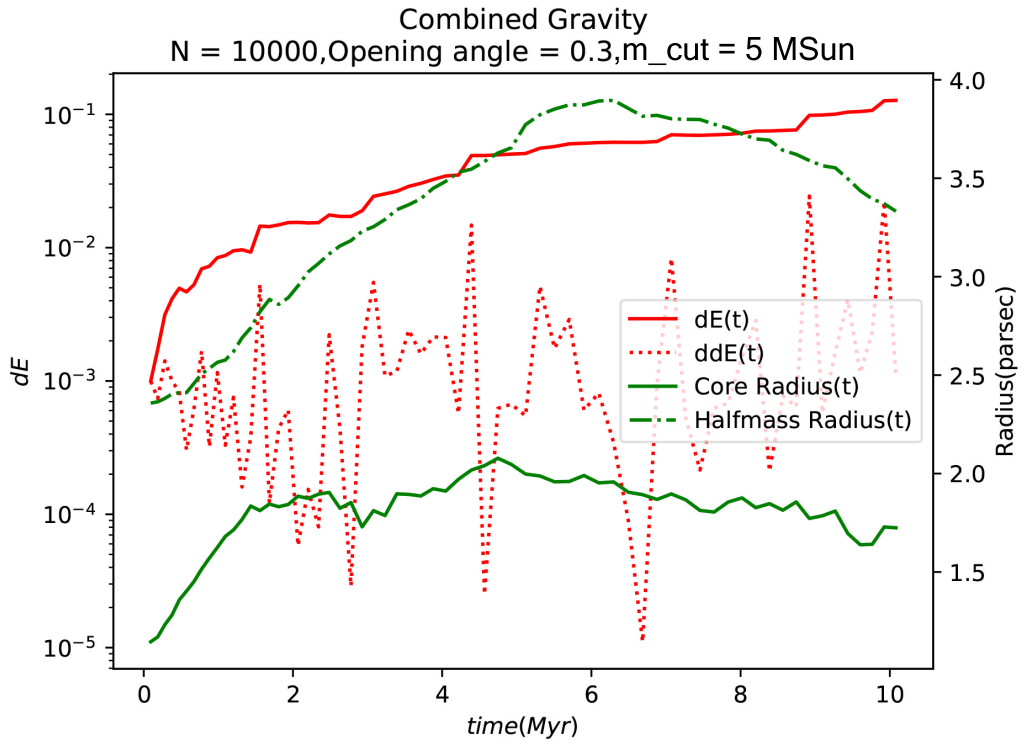

Figure 5: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 5M_{\odot}$.

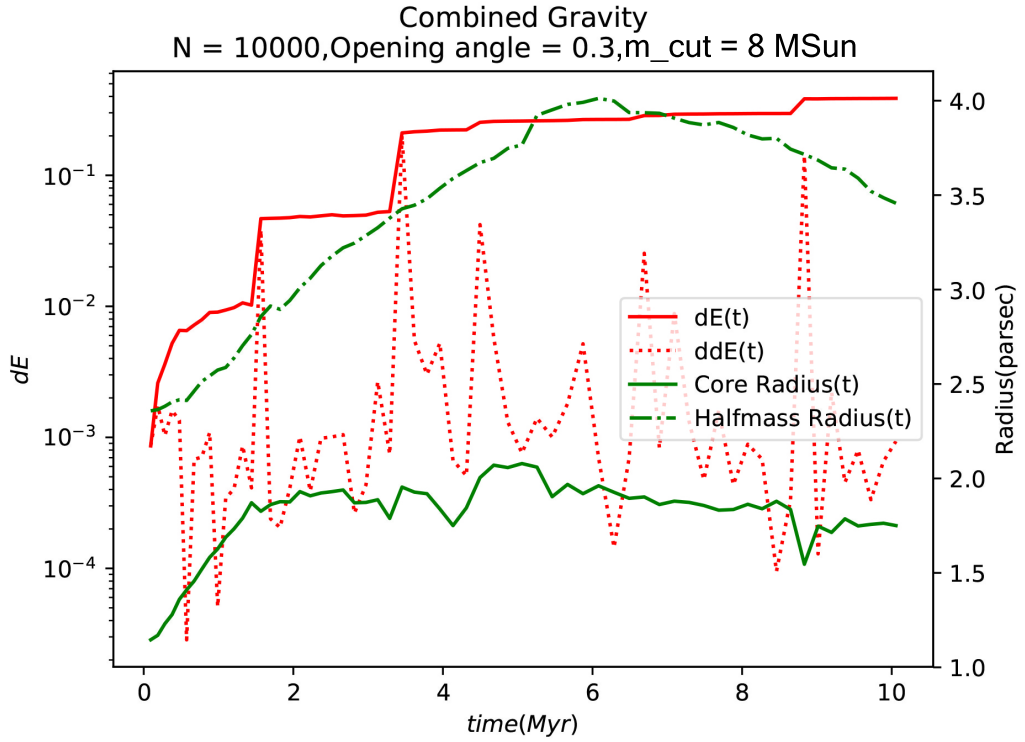

Figure 6: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 8M_{\odot}$.

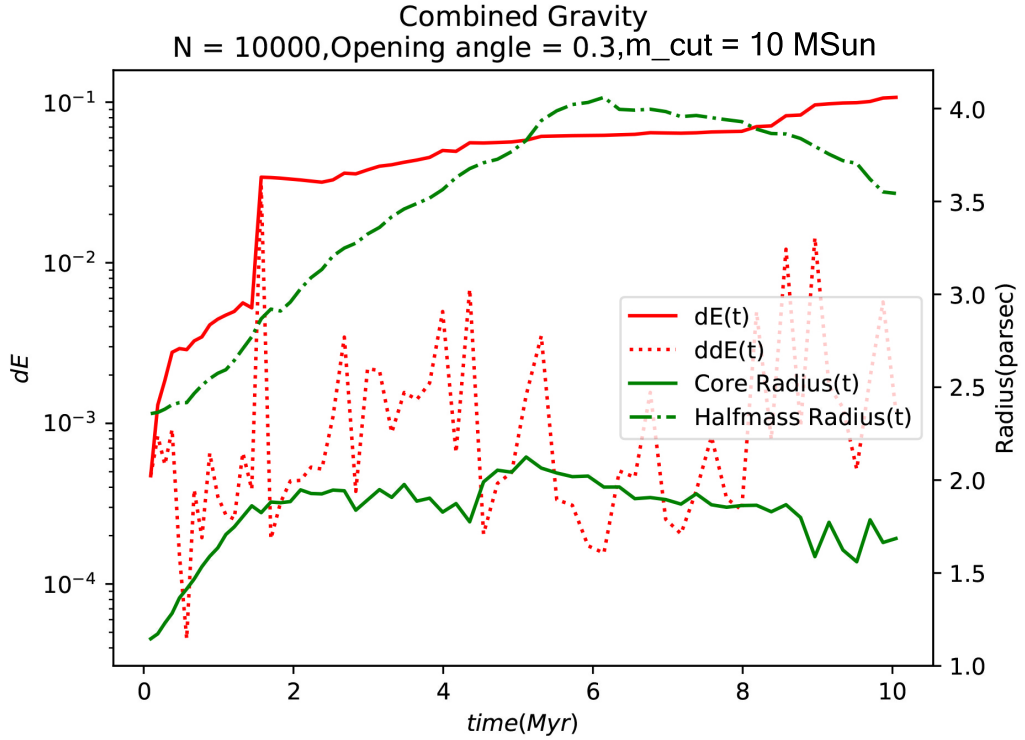

Figure 7: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 10M_{\odot}$.

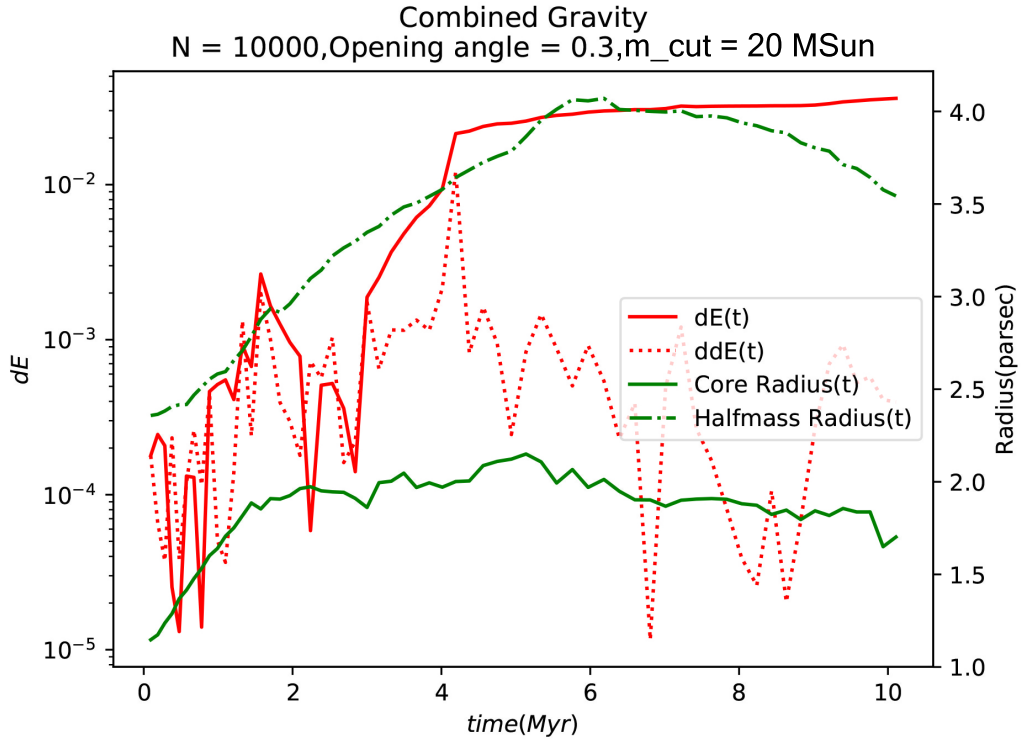

Figure 8: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 20M_{\odot}$.

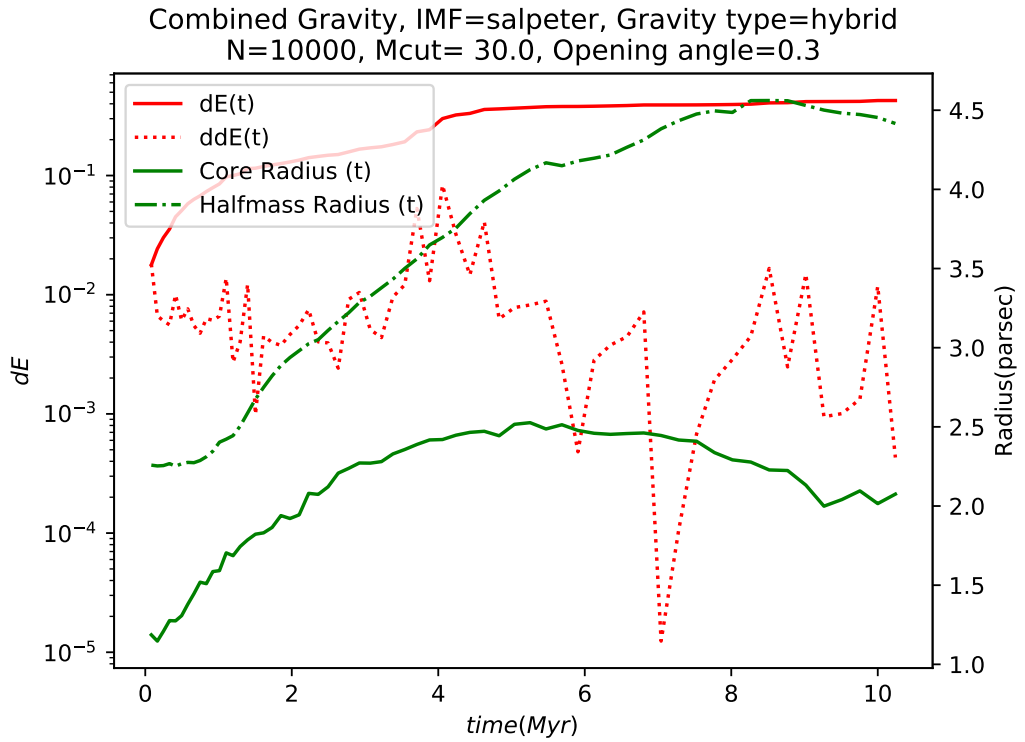

Figure 9: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 30M_{\odot}$.

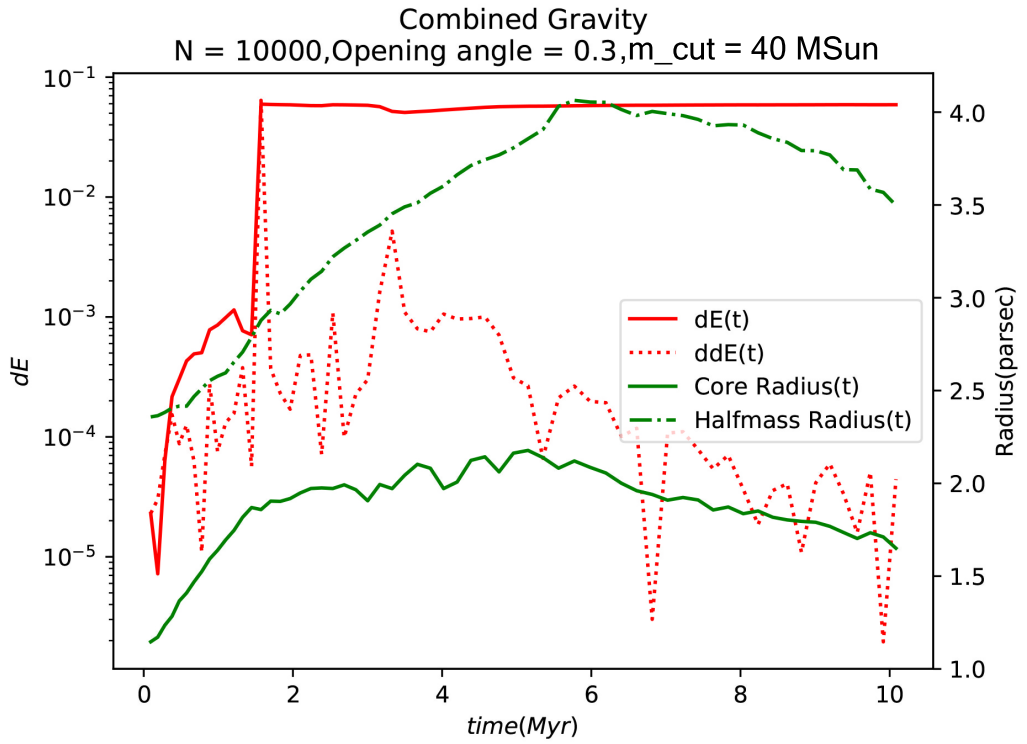

Figure 10: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 40M_{\odot}$.

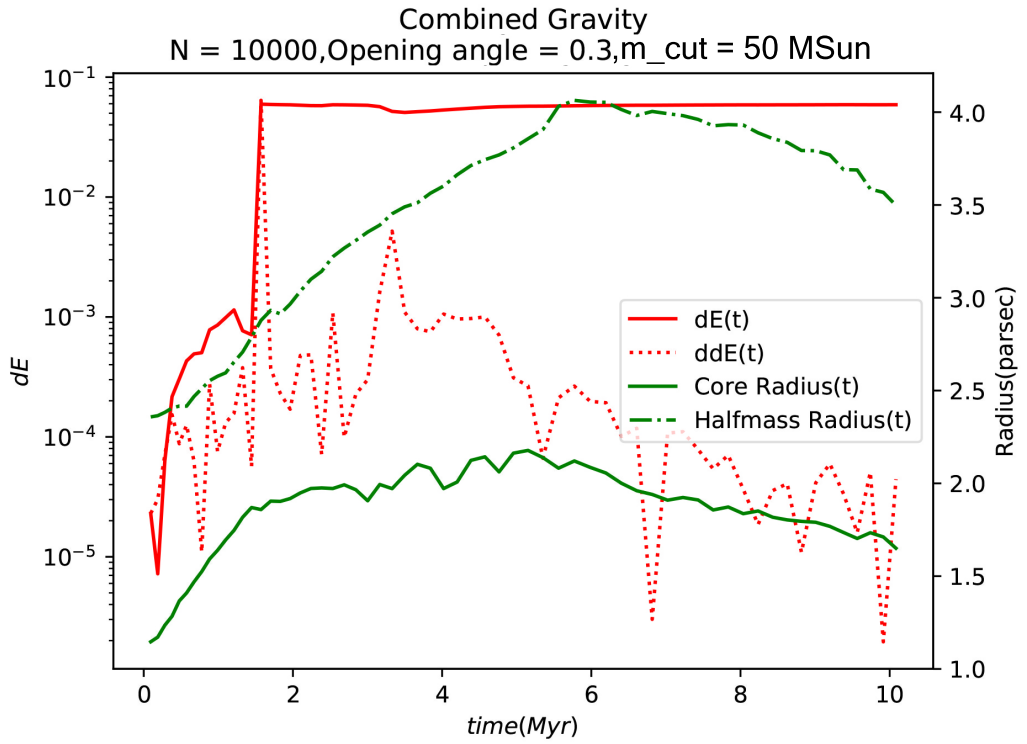

Figure 11: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 50M_{\odot}$.

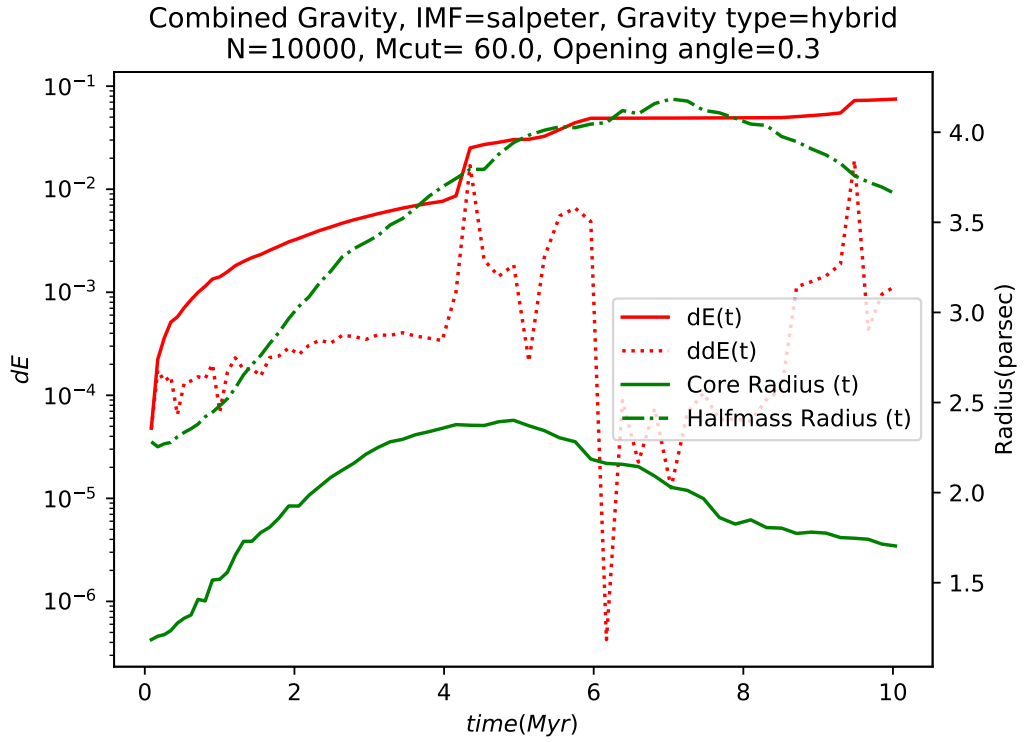

Figure 12: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 60M_{\odot}$.

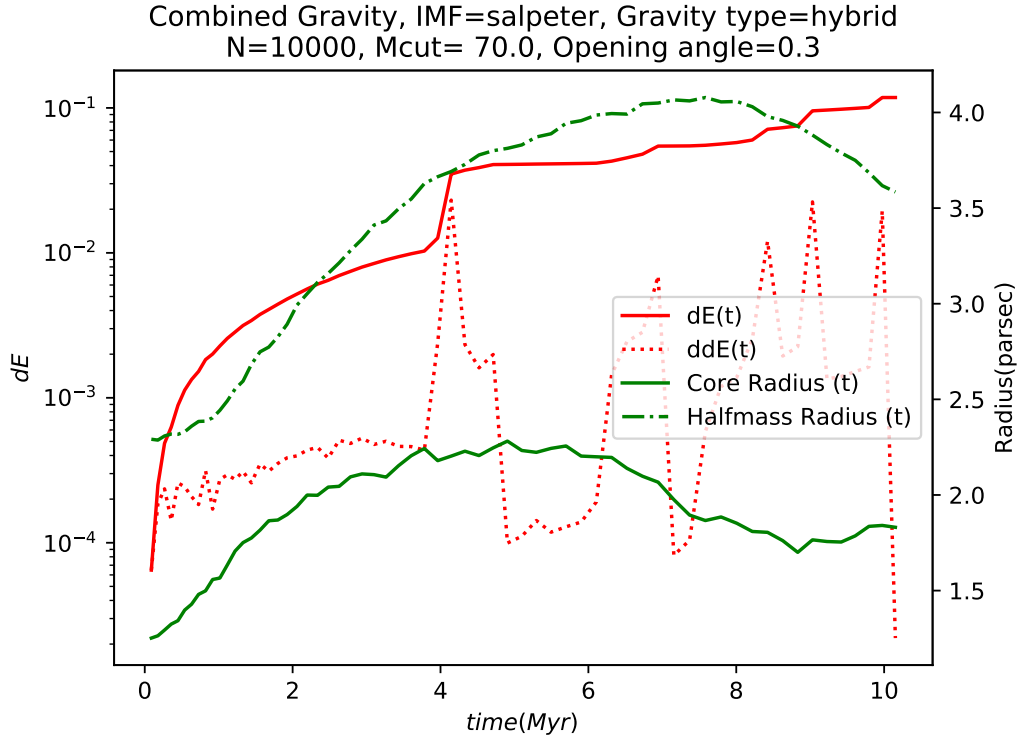

Figure 13: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 70M_{\odot}$.

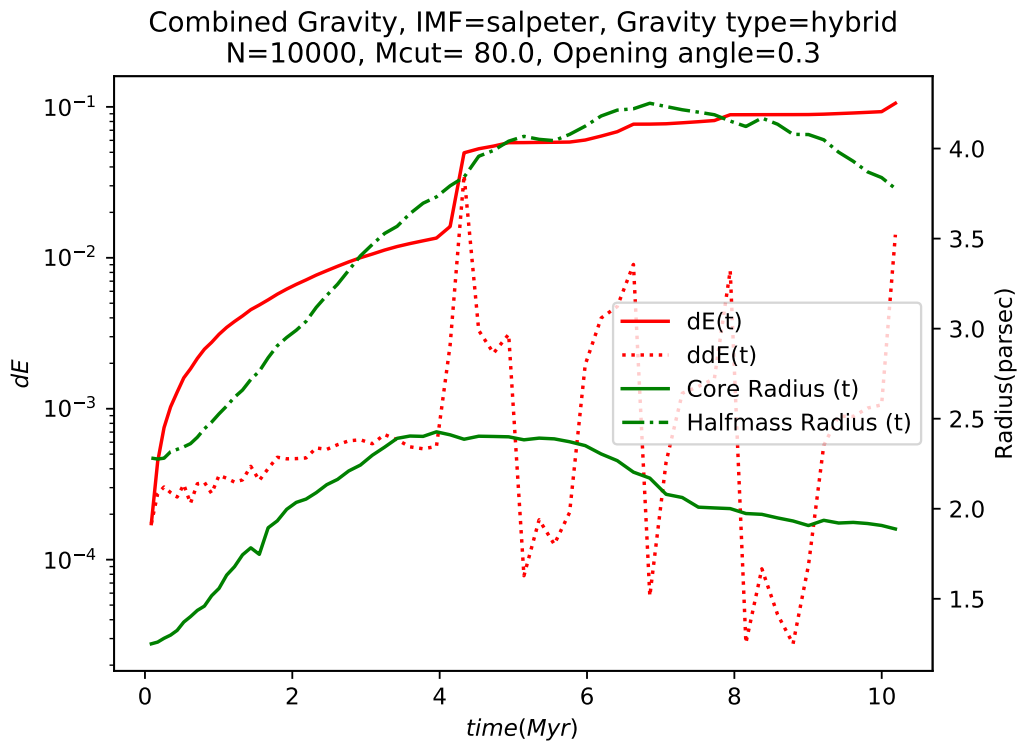

Figure 14: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 80 M_{\odot}$.

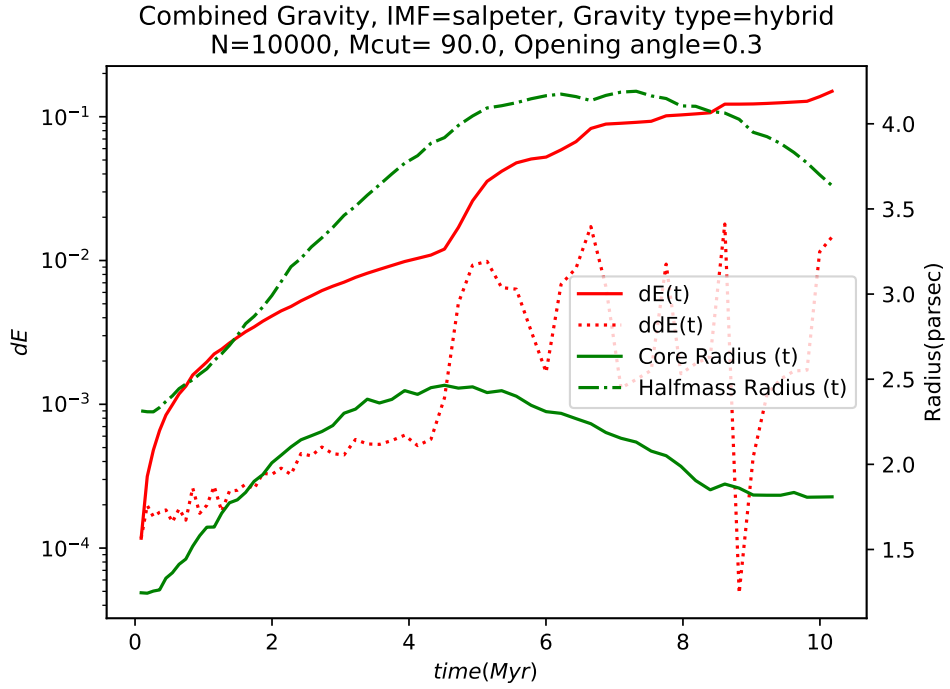

Figure 15: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 90 M_{\odot}$.

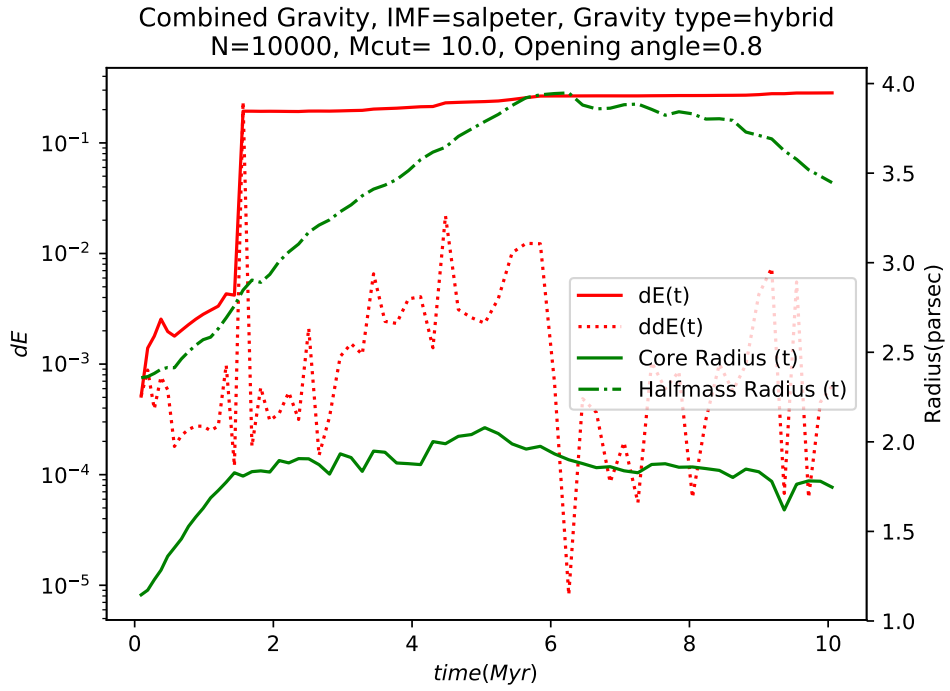

Figure 16: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 10M_{\odot}$, except for an opening angle $\theta = 0.8$ and not 0.3 like in the previous plots. Comparing this plot to figure 7, it can clearly be seen that the tree code is very sensitive to the opening angle parameter, and the closer it is set to zero, the lower the value of the accumulated energy error.

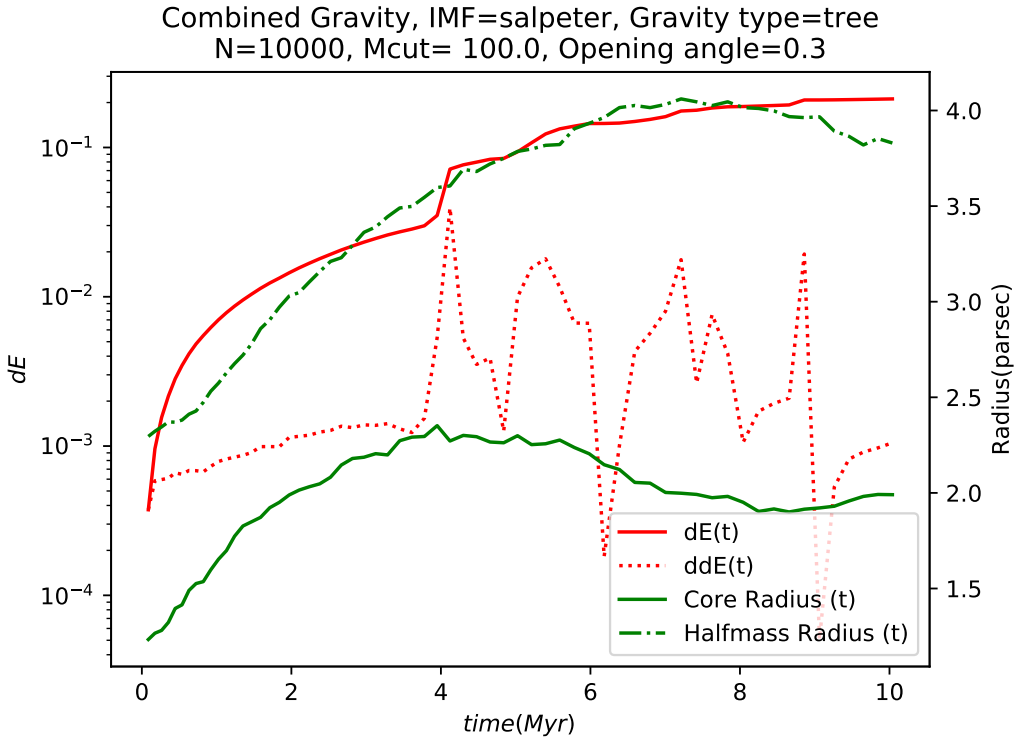

Figure 17: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 100M_{\odot}$ i.e. all stars in the **tree code**.

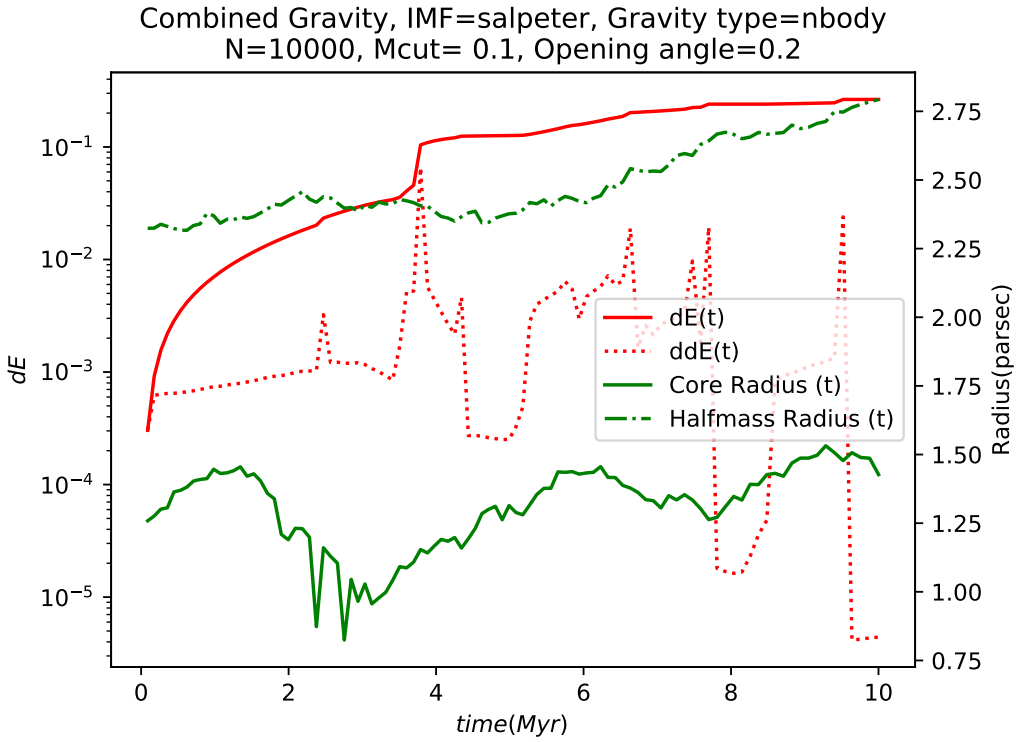

Figure 18: Relative energy error dE and energy error between the time steps ddE (left axis) and core and halfmass radii (right axis) as a function of evolution time for $m_{cut} = 0.1 M_{\odot}$ i.e. all stars in the **N-body code**.

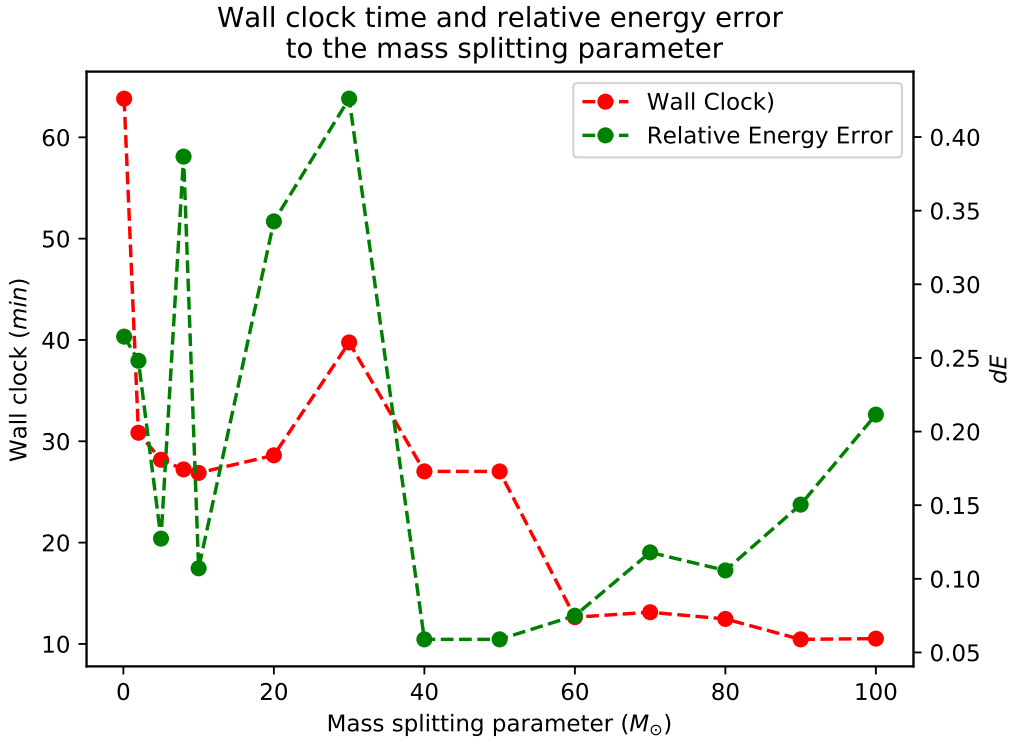

Figure 19: Wall-clock time (left axis) and relative energy error dE (right axis) a as a function of the mass splitting parameter m_{cut}.

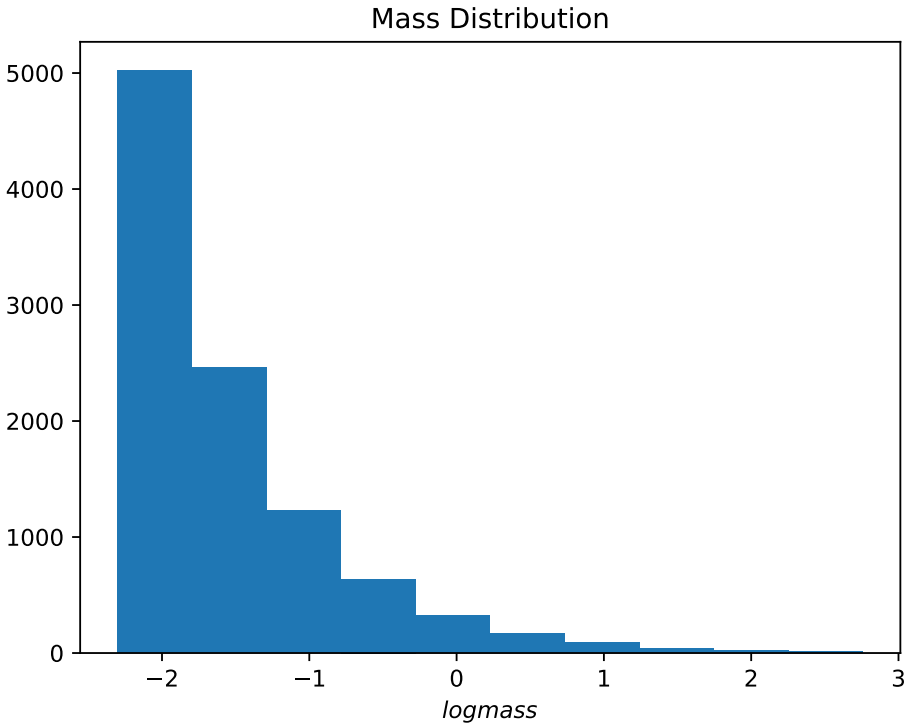

Figure 20: Histogram of the mass distribution in the star cluster using a **salpeter** IMF.

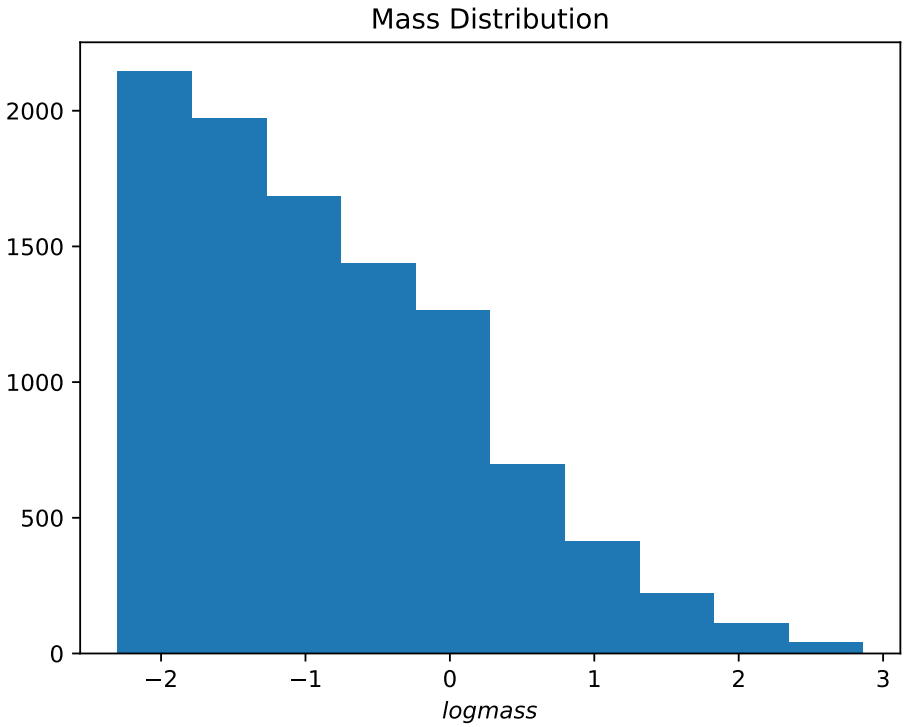

Figure 21: Histogram of the mass distribution in the star cluster using a **miller-scalo** IMF.

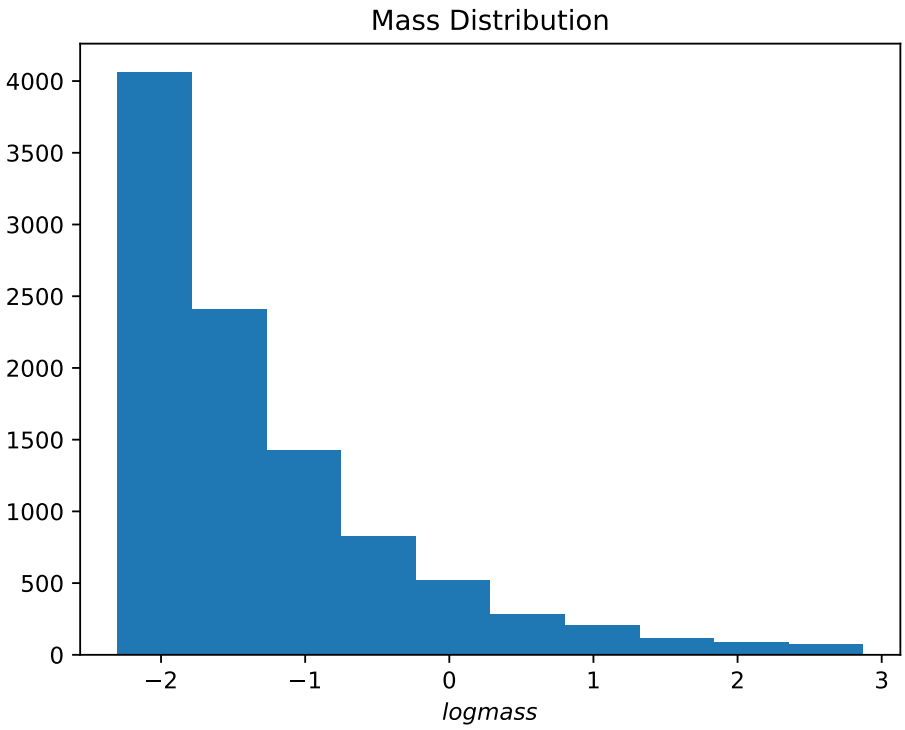

Figure 22: Histogram of the mass distribution in the star cluster using a power-law function **otherexp** with $\alpha = -2.0$ as an IMF.

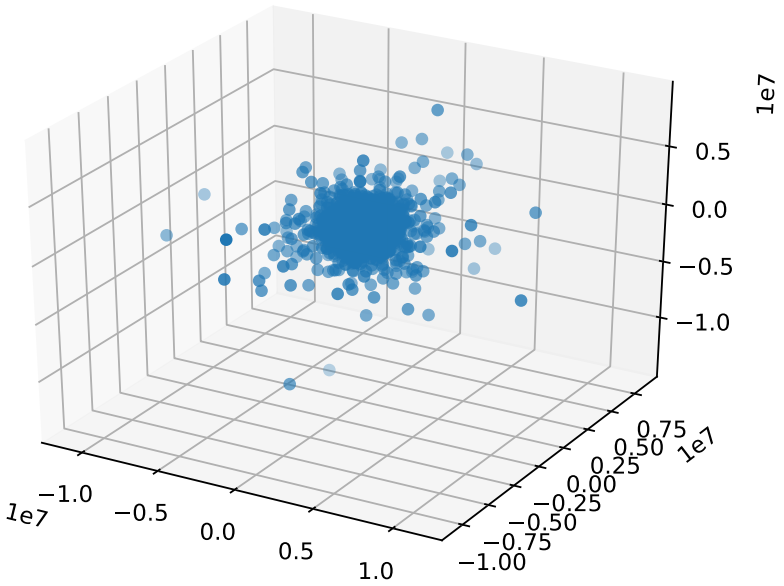

Figure 23: An example scatter plot of the star cluster at a certain point in time. The stars are around the (0,0,0) centre of the cluster, and as the core radius suggests, they are clumped near it.