# Computational Astrophysics - Assignment 2

Saeid Mahmoudzadeh - s2547341, Yuan Chen - s2562081
Christiaan van Buchem - s1587064

October 23, 2019

## 1 Introduction

In this report we will be presenting the work that we have done for assignment number 2 for the class named Computational Astrophysics. We will begin by giving some background information on the problem at hand. We will then explain our method, present our results, make an analysis and finally draw conclusions. The aim of this assignment is for us to learn how, when and why to make use of a bridge when using the AMUSE software Portegies Zwart et al. (2019). In order to accomplish this aim we will be studying a large system of self-gravitating particles distributed in a virialized Plummer sphere (Plummer, 1911). Due to the size of the simulation we will have to find the best compromise between speed and accuracy when integrating the N-body system. The manner in which this is done will be explained in the next section.

## 2 Method

### 2.1 Ideas of bridge and $m_{\text{cut}}$

We will be using two different integration methods for this exercise. The first is a direct n-body code, named `ph4`, which does a direct $N^2$ integration of the movement of bodies in a system. The second method is a standard tree-code, called `BHTree`, that uses the 'Barnes Hut' algorithm. Deciding which stars will be integrated by which code is done by using the $m_{\text{cut}}$ criterion. Stars with mass $m < m_{\text{cut}}$ will be integrated using the tree-code whilst the stars with mass $m \geq m_{\text{cut}}$ will be integrated using the direct n-body code. The manner in which we went about finding the best compromise was by finding the optimum value for $m_{\text{cut}}$ in terms of accuracy and speed. **The reason that we split the particle set in mass rather than distance from the center-of-mass or as a function of local density is because the mass is a constant value and we therefore only have to split the particles at the very beginning**. If we were to use a value based on the location of the particles, then we would have to re-divide the stars from the cluster after every-time step and pass particles from one integrator to the other. Aside from the complexity that this adds to the code, it is also very likely to result in a very large amount of computing time and therefore offset any advantage that is gained by combining two different methods in the first place. Coupling these two strategies will be done using the bridge feature available in AMUSE. In order to keep the scope of this assignment manageable, we will not be looking at the stellar evolution in this simulation.

### 2.2 Parameters to check

Judging the quality of a simulation is often done using the relative energy error. The energy error is due to small mistakes the program makes (due to round-off errors or due to limitations in the method). A relative energy error is calculated as follows:

$$E_{\text{error}} = \frac{|E(t) - E(0)|}{E(0)} \tag{1}$$

Where $E(t)$ is the total energy of the system at time $t$ and $E(0)$ is the total energy of the system at time $t = 0$. As stated in Portegies Zwart & McMillan (2018), the boundary above which the energy error is acceptable is quite arbitrary and dependent on the context. A rule of thumb, for direct N-body simulations, is that the energy error over the duration of the calculation should remain below 1% of the total energy of the entire system, and below $10^{-4}$ of the total energy of the system over and interval of one dynamical time. In Figure 2.11 of Portegies Zwart & McMillan (2018) we

can see a plot comparing the relative energy error of integration of the N-body integrator `ph4` and the `BHTree` code for different time-step sizes. Seeing as we will be using these same integration methods for this assignment, we can use it in order to judge the bridge time step that we will use for our calculation. We found that the dynamical timescale of the cluster that we generated was about $10^2$ Myr. In the aforementioned figure we can see that once the relative time-step (to the dynamical timescale) is smaller than $10^{-2}$ both integration methods are at their most accurate. We chose to use a bridge timescale of $10^{-1}$ Myr (the same size as the integration time-step) which correlates to a relative time-step of about $10^{-3}$.

Other measures that we use to track the evolution of the cluster over time are the half-mass and core radii. The half-mass radius is calculated using Lagrangian radii which can be directly calculated using AMUSE (where the 50% Lagrangian radius corresponds to the half-mass radius). The core-radius in an N-body simulation is defined as the density-squared weighted root mean square stellar distance from the cluster center. This value is also returned directly by AMUSE. When generating the star cluster we used a power-law mass function from $0.1M_\odot$ to $100M_\odot$ with $\alpha = -2.35$, we set $N = 10^4$ stars, used a virial radius of $r = 3$ pc, assumed a virial mass of $N * M_\odot$, and ran the code for 10 Myr with a time-step of 0.1 Myr. Lastly we also used a fixed seed (using the `numpy.random` package) for generating the cluster in order to be able to carry out the different interactions on the same cluster configuration each time. We will present our results in the next section.

# 3 Results

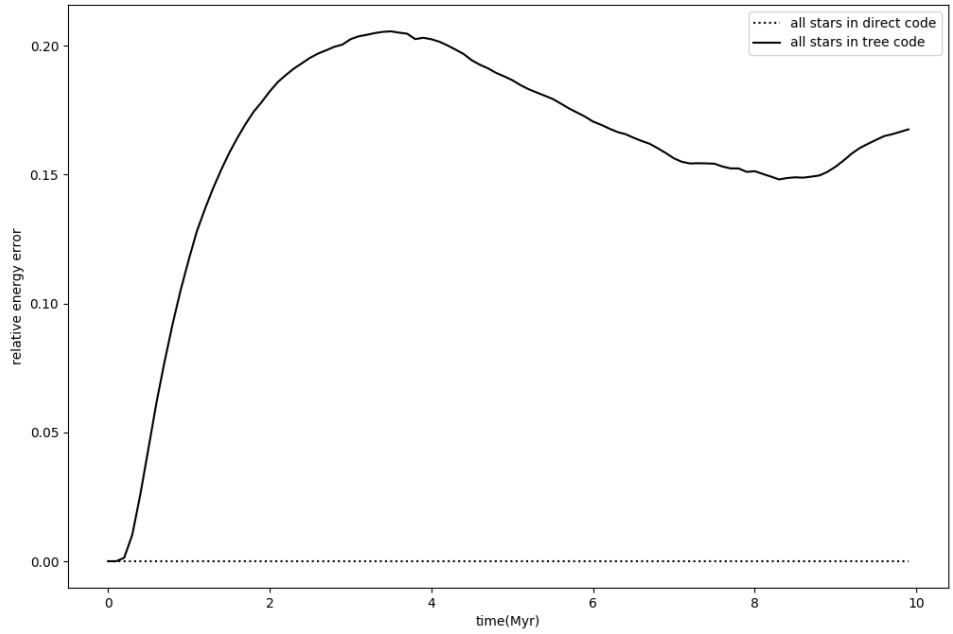
## 3.1 Integrating all particles using one method

First of all, we set our baselines by running the integration using the tree-code for all particles and then the direct N-body simulation. In order to judge the performance of these two integration methods, we compared them directly by plotting the evolution of the relative error and the radii of the cluster. These results are shown in Figure 1 and 2. From these results we can clearly see that the direct N-body code is far more accurate than the Barnes-Hut Tree algorithm (from now on 'tree code'). However, the direct N-body integration for all particles took almost two hours to run, whilst the tree-code took no more than a couple of minutes to finish. Therefore it is clear that we need to find a compromise.

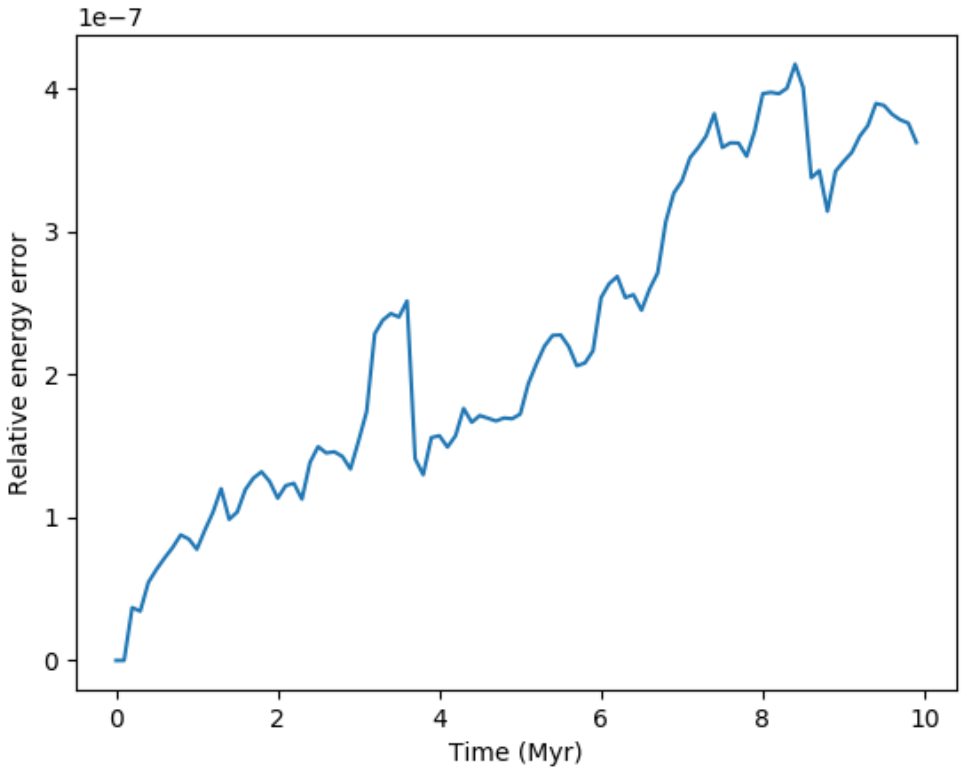
## 3.2 Relative energy and radius as a function of time

In Figure 4 we can see that for $m_{\rm cut} = 0.8M_\odot$ and $10.0M_\odot$, there is an obvious 'jump' in the relative energy error at some time during the 10-Myr evolution, which immediately makes the simulation 'unreliable'. We are not sure about the cause, but there is high possibility that it results from a quick ejection of massive stars due to close encounters. Considering the fact that we do not simulate collisions and that our time-step may be too large for these cases, our simulation is no longer accurate in these scenarios. We tried plotting out the virial ratio of the cluster system ($Q = -T/U$, which equals to 0.5 initially) along with the evolution time, and found that $Q$ would jump every time when the energy error jumps. The boost of $Q$ implies that the kinetic energy of the cluster system quickly increases or the (absolute value of) potential energy suddenly decreases. This is very likely to be caused by an ejection. We also tried animating the evolution with direct integration ($m_{\rm cut} = 100M_\odot$), and actually saw the occurrence of ejection. (But in that case the ejections are calculated properly by the direct N-body code, and does not lead to a jump in the relative energy error.)

Now the problem is that we cannot find a rule of when (i.e. with what value of $m_{\rm cut}$) the jump will happen. If we expect that the larger the $m_{\rm cut}$ is, the less accurate the simulation is (since less star are classified into the high-mass group (Figure 3) and integrated with the direct N-body code), there should be more 'severe' jumps after a threshold of $m_{\rm cut}$. However, this seems not to be the case, since from Figure 4 we can see that the energy error with $m_{\rm cut} = 0.8M_\odot$ boosts more heavily than that with $m_{\rm cut} = 1.0M_\odot$. From Figure 7 we can also notice that even when the $m_{\rm cut}$ is larger than 5 $M_\odot$, the final energy error could be small in some cases, i.e. there is no significant deviation during the whole evolution. We ran the simulation with large values of $m_{\rm cut}$ under the same random seed for several times, and got similar results in terms of energy error and wall-time. Therefore we suspect that these 'jumps' are caused by either the bridge of a direct N-body code with a less accurate tree code, or the choice of time-step. It may also has some relation with the random seed which determines the initial mass and spatial distribution of the cluster members, but this is hard to verify. As for reducing the time-step of integration, we only looked into this point




(a) Relative energy error as a function of time for the direct integration method (dotted line) and for the 'Barnes Hut' tree method (solid line).




(b) Zoomed in image of the relative error when all particles are directly integrated.

Figure 1: Here we can see the comparison between the two integration methods. In Figure 1a we can clearly see that the relative energy of the direct integration method is negligibly small compared to the 'Barnes Hutt' tree integration method. In Figure 1b we can see a more zoomed in view of the error of the direct method. Here it is clear that there is some increase in the error over time, most likely due to round-off errors.

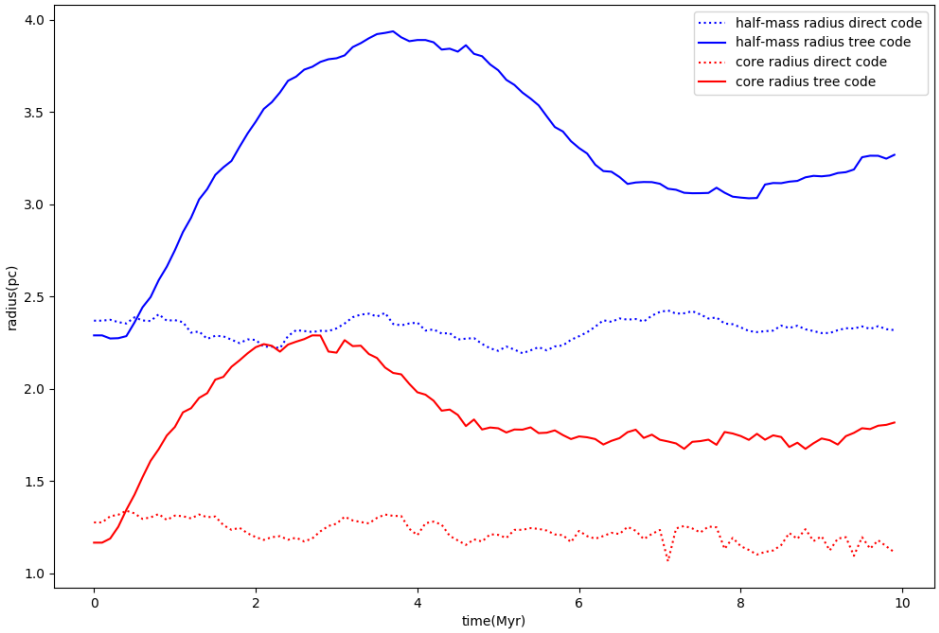



Figure 2: Half-mass and core radii as a function of time for the two different integration methods. As in Figure 1a the dotted line represents the direct integration method whilst the solid line represents the 'Barnes-Hut' tree integration method. Note the possible correlation between the increase in relative error seen in Figure 1 and the increase in radii for the `BHTree` method.

in the case of $m_{\rm cut} = 10M_\odot$ (Figure5). We only reduced the time-step by a factor of two (from 0.1Myr to 0.05 Myr) and the relative energy error already shows a great improvement. But this comes at a price that the computation time will multiply. So it is a trade-off in choosing suitable values for $m_{\rm cut}$ and the time-step.

At least there is one thing that we are sure about: if the $m_{\rm cut}$ is small enough, with which the cluster has more (or roughly the same) high-mass stars than the low-mass ones ($m_{\rm cut} = 0.3 - 0.5M_\odot$, see Figure 3), the jumps will not occur and the relative energy error will keep constrained throughout the evolution.

We plotted the half-mass and core radii for $m_{\rm cut} = 1M_\odot$. The results can be seen in Figure6. Both the half-mass radius and the core radius can be seen to be growing over time. As noted in Portegies Zwart & McMillan (2018), clusters can in the long run evolve to a point where core collapse occurs. Although this could explain the increase in half-mass radius, this would also require a decrease in core radius and is therefore not the cause of the observed behaviour. It is more likely that the change the growth in the radii is due to inaccuracies in the simulation as similar to those visible in Figure 2 for the tree code.

### 3.3 Final energy error and wall-clock time as a function of $m_{\rm cut}$

Finally we also made some plots comparing the final energy error and the wall-clock time for different $m_{\rm cut}$ values. These can be found in Figure 7 and 8 respectively. In Figure 7 we also decided to include the 'fail point', which is to say, the point at which the relative energy error reaches 1. The lower this point, the less trustworthy the simulation. In general we can see that the simulations with $m_{\rm cut}$ around $0.5M_\odot$ are most accurate with no real visible trend afterwards. In Figure 8 we can see that the run-time becomes manageable around an $m_{\rm cut}$ of about $1M_\odot$. This comes to no surprise when looking at Figure 3 where we can see that this coincides with the point where the number of high-mass stars becomes significantly less than the number of low-mass stars.

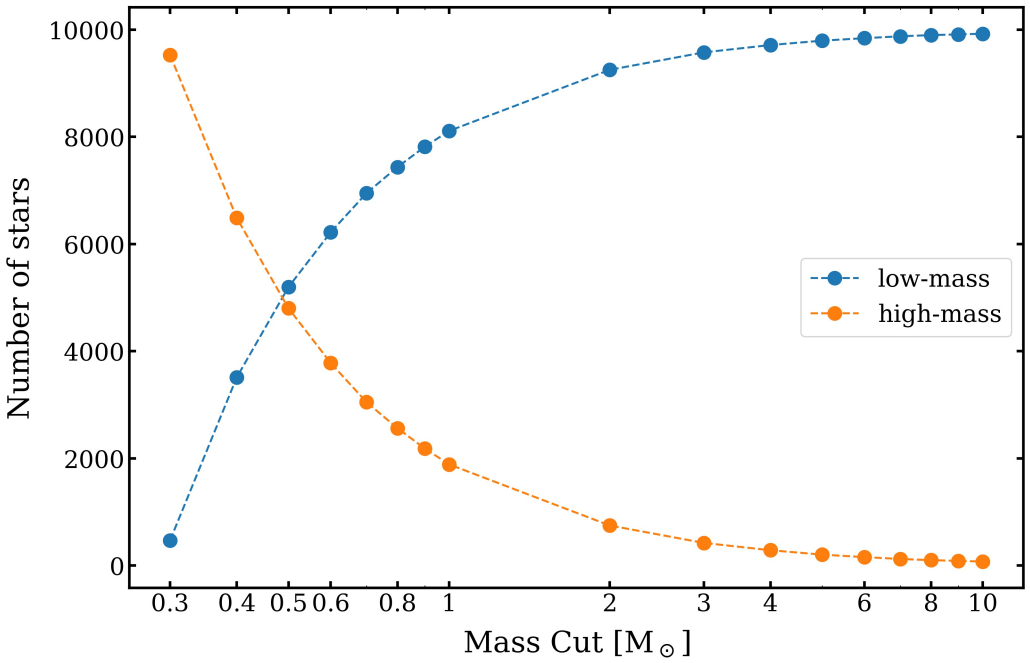



Figure 3: The number of low-mass stars and high-mass stars that are classified by $m_{\rm cut}$

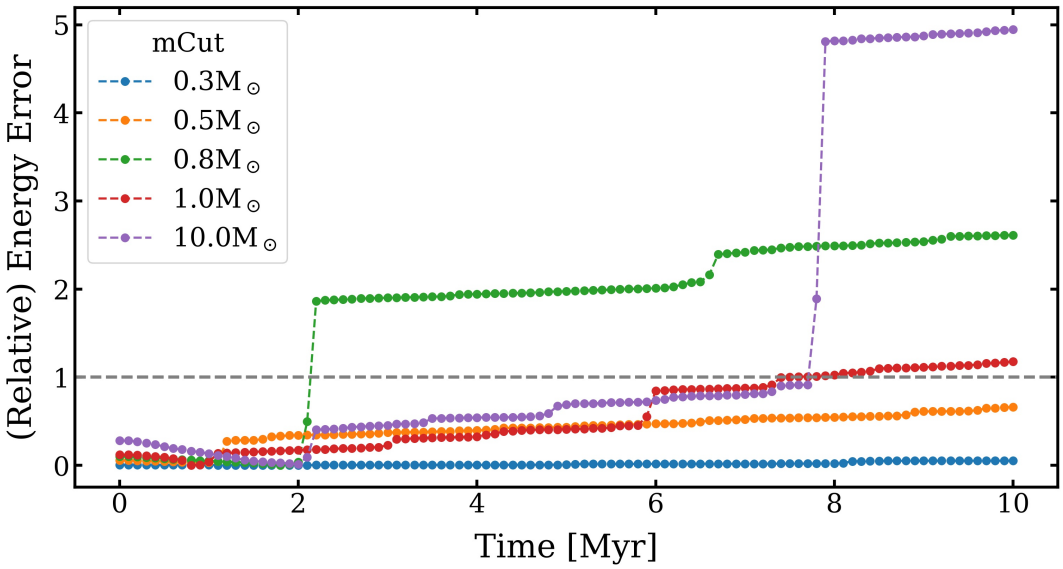



Figure 4: Relative energy error versus evolution time for four different $m_{\rm cut}$: 0.3, 0.5, 0.8, 1.0, and 10.0 $M_\odot$. A threshold of 1 of the relative energy error is labeled by a horizontal grey dashed line.

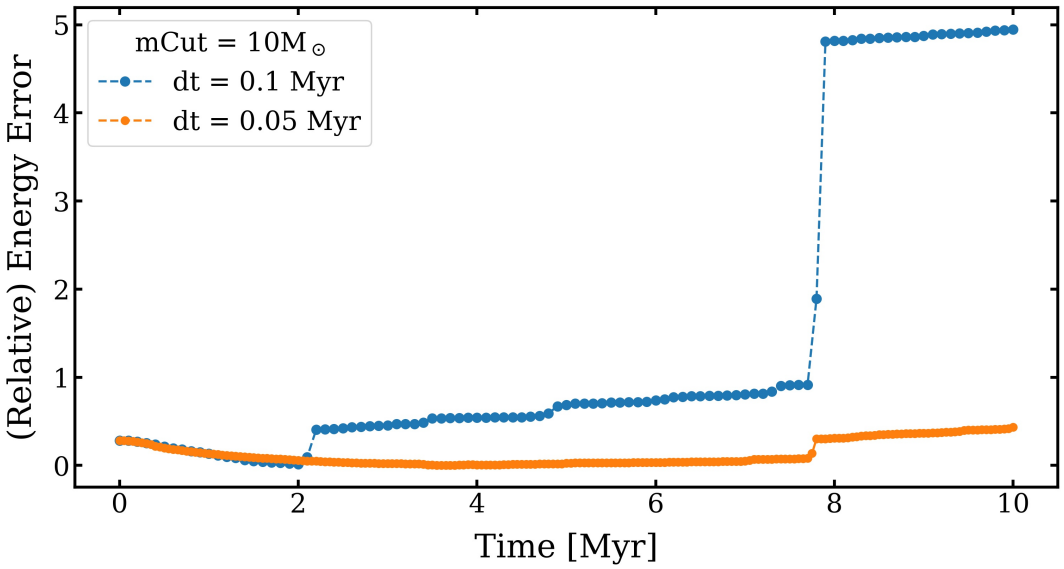
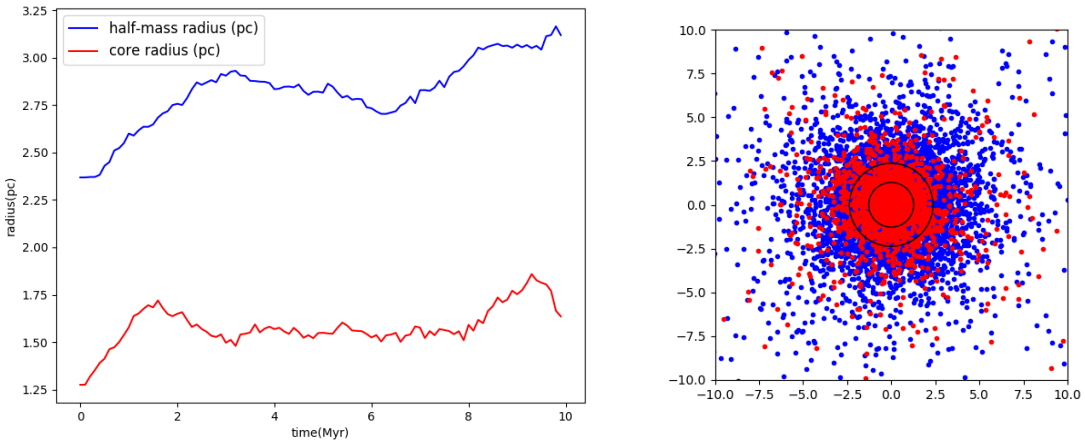



Figure 5: Relative energy error under the same mass cut but with two different time-steps. Only by reducing the time-step by a factor of two, the integration gets rid of ejections and remains its reliability throughout 10 Myr – a smaller time-step really helps increase the integration quality!




(a)

(b)

Figure 6: In Figure (a) half-mass and core radii are plotted as a function of time when $m_{\rm cut} = 1M_{\odot}$. In (b) we see a plot stars plotted according to their coordinates with the high mass stars represented as red dots and the low mass stars as blue dots. The inner circle represents the core and the outer circle the half-mass-radius.

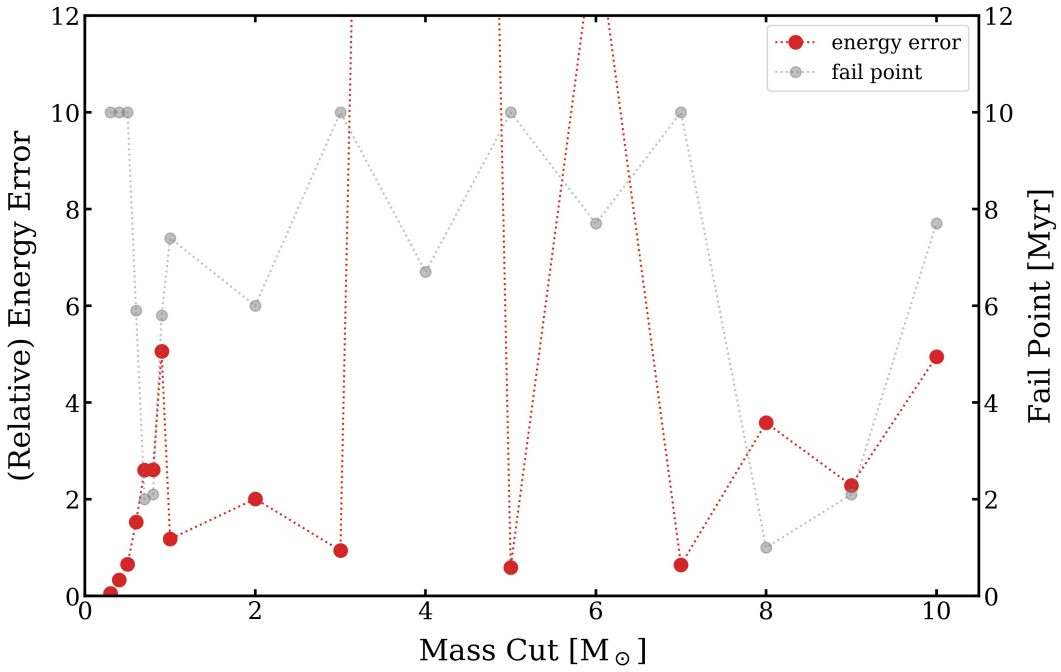



Figure 7: The final relative energy error and the 'fail point' versus $m_{\rm cut}$. The fail point means the time stamp right before the relative energy error reaches 1, which is a threshold beyond which the simulation is not reliable any more. When the fail point is low, the corresponding energy error loses its eligibility of reflecting the accuracy of this simulation. Actually we can see a negative relation between these two results, i.e. when the fail point is low, the final energy error tends to be large.

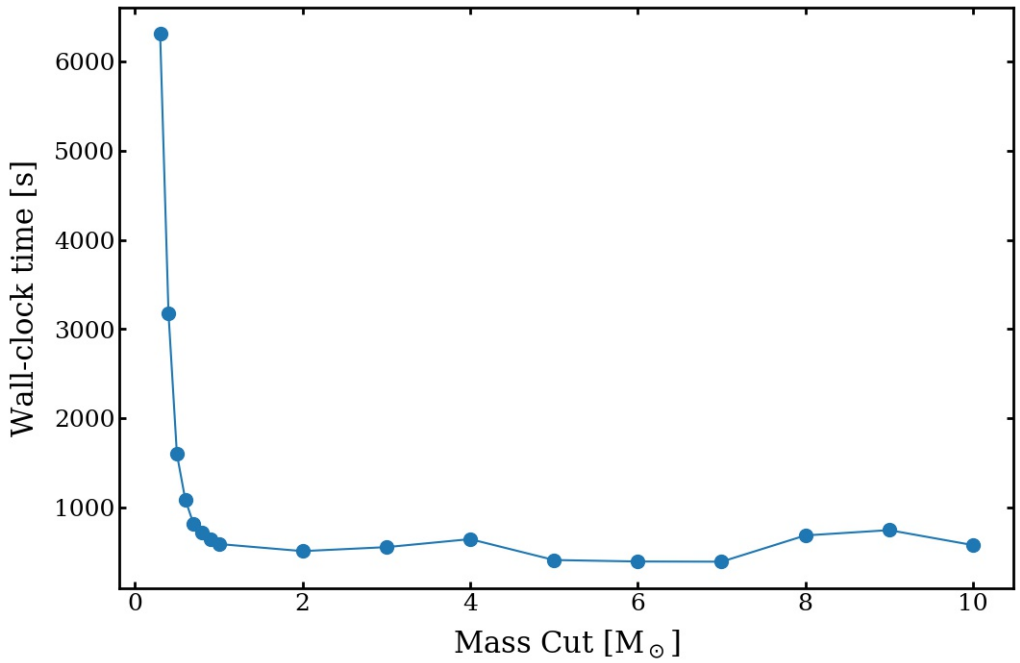



Figure 8: Wall-clock time versus $m_{\rm cut}$. The general trend is that the time cost decreases when the mass cut increases. The slight 'bump' in the tail of this curve is probably caused by big ejections during the simulation. Since if some cluster stars are ejected very far away from the center of mass, the integration tends to cost more time in each step.

# 4 Discussion and conclusion

It appears that integrating the cluster using two different methods that are bridged is effective up to a certain point. A very significant amount of time is saved (up to 12 times faster) when dividing the particles between the two methods instead of directly integrating the cluster using N-body code. It is harder to draw clear conclusions from the final relative energy graph (Fig 7). It is clear that using the `BHTree` algorithm makes for a far less accurate integration than using only the `ph4` integrator, but no clear trend can be discerned. We believe that this is caused by the inaccuracies surrounding close encounters. In order to investigate this further, an average final relative energy error should be calculated for a large number of different cluster initialization, though apparently we do not afford that amount of effort here. However, for low $m_{\text{cut}}$ values, the simulation is already greatly improved: it remains relatively accurate whilst saving a lot of time. When taking these factors in consideration, we believe that with the current set of initial conditions the most advantageous $m_{\text{cut}}$ value is $0.5M_{\odot}$, since it is significantly faster (6 times) than using the full N-body code and in the meantime still remains fairly accurate.

If we were to continue further with this project we would look into finding a way to avoid the inaccuracies caused by close encounters. One possible way in which we could investigate this is by using the distance of the nearest star as a dividing criterion for the two integration methods, with all close encounters utilizing the N-body integrator whilst the non-close encounters would be integrated using the tree code. The trouble with this of course would be the fact that particles would have to be passed from one integrator to the other throughout the simulation which might mitigate the computation time decrease. Another factor that we could potentially look into is decreasing the time step taken by one or both of the integration methods. Some of our experiments have already suggested that this may indeed lead to a lower final energy error (Figure 5) and would therefore allow us to use a high $m_{\text{cut}}$ value and hence cost less computation time.

# References


Plummer H. C., 1911, Monthly Notices of the Royal Astronomical Society, 71, 460

Portegies Zwart S., McMillan S., 2018, Astrophysical Recipes: The art of AMUSE. IOP Publishing, Bristol, doi:10.1088/978-0-7503-1320-9

Portegies Zwart S., et al., 2019, AMUSE: the Astrophysical Multipurpose Software Environment, doi:10.5281/ZENODO.3260650, https://zenodo.org/record/3260650#.XZMXFeYzaAk