Project 2: Simulating Realistic Star Clusters. Comparison between PeTar and nbody6++gpu

Cortese Pietro [2023926],[1]★
[1]Department of Astronomy, University of Padua, Vicolo dell'Osservatorio 3, I-35122 Padova, Italy

15 February 2022

ABSTRACT

Direct N-body simulations are a powerful tool to study the dynamics and the evolution in time of star clusters. As the hardware and software improved during the years, the codes devoted to perform this kind of simulations have increased the number of particles that can be studied without over-increasing the computational time. nbody6++gpu and PeTar are two well known codes that can handle direct N-body simulations of realistic star clusters with a high number of particles and fraction of binary systems. In this report I will present a comparison between the results of a direct N-body simulation of a realistic star cluster performed by the two codes, whose initial conditions have been computed with an hydrodynamic simulation of a molecular cloud, to analyze the differences that arise between them in their time evolution.

Key words: methods: numerical – stars: kinematics and dynamics - star clusters: general

1 INTRODUCTION

N-body simulations have many applications in different scientific fields. In astrophysics they are used to perform numerical integration of the gravity force acting on N particles for a time t; the particles can be gas, stars, dark matter, etc. depending on the problem that is studied. One of the main application in astrophysics of this simulations is the evolutionary study of star clusters, that are self-gravitating systems composed of $\sim 10^2$ - 10^7 stars and have size of some parsecs. In this case the particles studied are the single (or binary) stars and since this kind of systems have a relaxation time, which is a time-scale that defines the importance of the gravitational two-body encounters for the orbital motion (Chandrasekhar 1942), that can be shorter then the evolutionary time of the system (i.e. they have a large density), close encounters may often occur; indeed, the main driver of the stellar clusters dynamics are binary systems and three body encounters (i.e when binaries interact with single stars and exchange energy with them). Therefore, the algorithm that performs the N-body simulation of the stellar cluster must resolve individually the stars and it must integrate the Newton's equations directly (i.e. without any assumption). For this reason the codes of this kind that perform N-body simulations are called *direct N-body codes*. Since this systems are composed by many stars, the number of calculations required to study the interactions between every particle (numerical complexity) at every time-step of the simulation is very high, as it grows as the second power of the number of the particles N^2.

Furthermore, in order to properly describe the close encounters, the code used to perform the N-body simulation must implement an high-order integration scheme and a small time-step; in this way the system is able to conserve it's angular momentum and energy when, for example, a three body encounter occurs. This is crucial in order to describe in a proper way the evolution in time of the star cluster.

However this required conditions (short time-step and high precision) demands a powerful hardware to describe the evolution of a star cluster, and the computational time needed by the code that implements the N-body simulations to run can be very high. In particular, having a short time-step increase a lot the computational time. To overcome this bottleneck, Aarseth (2003) introduced a technique called *block time-steps*, which is frequently used in direct N-body codes; this techniques is based on assigning shorter time-steps to particles that undergo a close encounter and a longer one to the "unperturbed" particles; however, since providing a different time-step for each particle requires a lot of time and the system would loses coherence, the particles are grouped together and their individual time-steps are replaced with a block time-step (usually the time-steps are powers of $1/2$). This allows to assign a short time-step to stars that undergo close encounters in order to study them in a proper way and a longer one to particles that are unperturbed and do not requires short time calculation, to reduce the computational time of the simulation.

As algorithms and hardware have improved in the time, the direct simulations of star clusters could increase the number of stars that can be taken in account while reducing the computational time needed, allowing a more realistic representation of the dynamical evolution of the clusters. One of the main hardware improvement used to compute N-body simulations has been the implementation of GRAPE (GRAvity PiPe), which is a project that uses hardware acceleration to perform Newtonian pair-wise force calculations between particles in self-gravitating N-body systems. This highly specialized hardware has been replaced with the GPUs (Graphics Processing Units), that have been created mainly for heavy graphical tasks, but they were found to be useful also for N-body calculations (Belleman et al. 2008). On the other

★ E-mail: pietro.cortese@studenti.unipd.it (UniPD)

hand, a worth to cite algorithm that has been implemented in the N-body simulations that significantly reduce the numerical complexity is the *Barnes-Hut tree scheme* (Barnes & Hut 1986). This algorithm allows to decrease the numerical complexity from N^2 up to $N \log N$ by calculating directly the Newton's acceleration only for the neighborhood of the particle taken into account, while the distant particles are grouped together in blocks and considered as single particles with mass equal to the total masses of the particles contained in the small block and position set in its center of mass.

The aim of this report is to analyse and compare the outputs of two N-body simulations performed with the codes NBODY6++GPU (Wang et al. 2015) and PETAR (Wang et al. 2020), that are the two widely used direct N-body codes, computed starting from the same realistic initial conditions of a star cluster produced with hydrodynamics simulations (Ballone et al. 2020). This analysis has been done to detect the different behaviors of the codes in computing the time evolution of this star cluster. The report is organized as it follows; in Section 2 I present the initial conditions of the simulation, the NBODY6++GPU and PETAR codes and the scripts used to analyse their outputs. Section 3 describes the results obtained by the analysis. In the Section 4 I discuss the results and finally in Section 5 the conclusions are reported.

2 METHODS

To perform this analysis, two simulations have been run by two different codes, NBODY6++GPU and PETAR[1]. In order to have a fair comparison, the initial conditions of the two simulations were identical. In the following I will focus on how this initial conditions have been computed, the two codes used for the simulations and the scripts that I have used to analyse the outputs.

2.1 Initial conditions of the simulations

To define the initial conditions of a star cluster described by N particles, the main quantities that must be assigned are the position coordinates (x, y, z), velocity components (v_x, v_y, v_z) and masses (m) of each particle. In some cases other parameters can (or must) be added (e.g. pressure, temperature or luminosity of the star). One way to assign this quantities is to sample a distribution function that describes the quantity I want to define with Monte Carlo technique (e.g. generate particles randomly as to be representative of the distribution function of the quantity). An example of distribution function is the Plummer sphere (Plummer 1911), that is a spherical density distribution described for the first time by Plummer in 1911 to fit observations of globular clusters. However the initial conditions described by distribution functions like this one are idealized. Star clusters form from molecular clouds and do not have a spherical or smooth shape, but are irregular and may present multiple density peaks. To perform proper theoretical studies on this systems, a more realistic description of their initial conditions is needed. Indeed, there are stronger and stronger indications that the observed proprieties of the star clusters could be imprints of their formation process. Furthermore, a non-monolithic formation could explain the differences

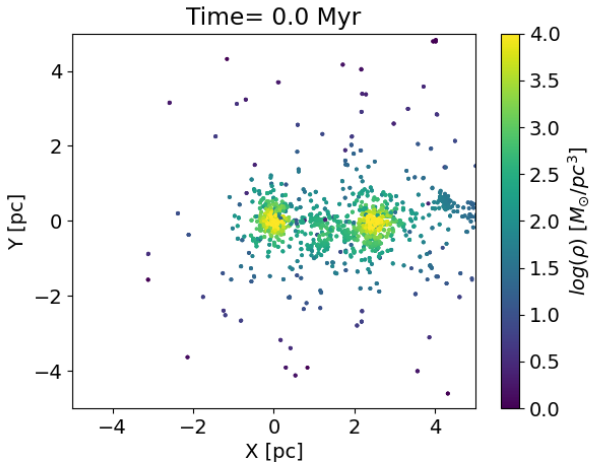

Figure 1. Projection on the $x - y$ plane of the star cluster's particles at $t = 0$ Myr. The color correspond to the density at the particle position expressed in $\log(M_\odot/pc^3)$

in the different chemical proprieties, kinematics and segregation observed in globular clusters (Ballone et al. 2020, 2021; Fujii et al. 2021).

For this work I have used the output of one hydrodynamical simulation of a turbolent molecular cloud performed with the SPH code GASOLINE (Wadsley et al. 2004) obtained by Ballone et al. (2020). The cloud had initial uniform density of $n = 250$ cm^{-3} and temperature $T = 10$ K. Moreover the simulation had a fixed number of 10^7 gas particle with a gravitational softening of $\epsilon = 10^{-4}$ pc and the initial mass of the molecular cloud was of $M_{mc} = 10^4$ M$_\odot$. A projection on the $x - y$ plane of the stars initial positions can be found in Figure 1.

2.2 The codes

The output of the hydrodynamical simulation presented in the previous paragraph has been provided as initial conditions for two different codes that performs direct N-body simulation, NBODY6++GPU and PETAR. In the following I will present a brief explanation about this codes.

2.2.1 *NBODY6++GPU*

The NBODY6++GPU[2] code (Wang et al. 2015) is an optimized version of NBODY6++, that is in turn the extended version designed for large particle number simulations by supercomputers of NBODY6 (Aarseth 2003). This code implements the hybrid parallelization methods to accelerate direct N-body simulations with a large number of particles, and in particular to solve the million-body problem. The NBODY6 and his successors use several algorithms to enhance the computing speed and accuracy, especially for close encounters that arise from a large fraction of binaries and relatively short relaxation time. For the Newton's equations integration it uses the fourth-order Hermite integration method and the hierarchical block time-steps, that adjusts the time-steps of the

[1] The two simulations have been run by Dr. Sara Rastello (PETAR) and Dr. Stefano Torniamenti (NBODY6++GPU)

[2] The NBODY6++GPU code can be found in the following GitHub repository: https://github.com/nbodyx/Nbody6ppGPU.git

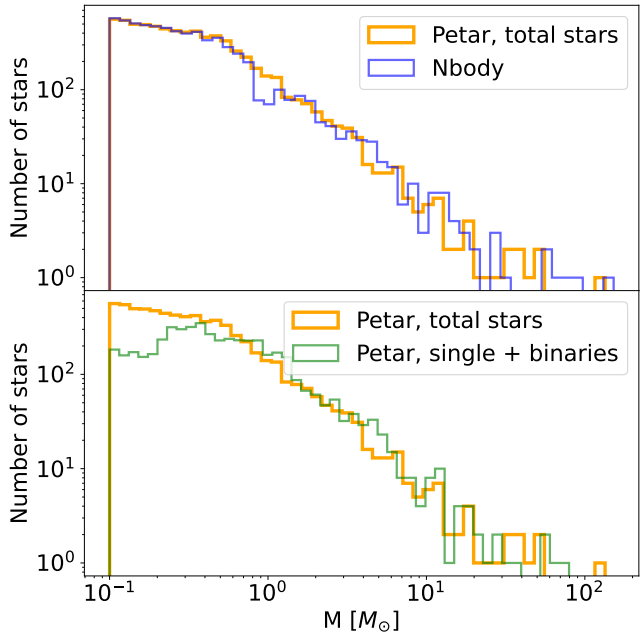

Figure 2. Initial mass distribution of the stars: In the top panel for NBODY6++GPU (blue) and for PETAR total stars (orange). In the bottom panel comparison between the PETAR output file 1 (orange) and 2 (green)

particles to quantized values (usually an integer power of 0.5). The main implementation of the NBODY6++GPU code is the GPU acceleration, that is efficient in reducing the computational time of the long-range gravitational forces calculation and in predicting the neighbour forces, as reported in the paper by Wang et al. 2015. Moreover, the NBODY6++GPU code can implements also the stellar evolution in the calculation (i.e. it can be coupled with a population synthesis code like MOBSE (Giacobbo & Mapelli 2018)).

For the simulation ran with NBODY6++GPU the metallicity of all the stars has been set to $[Fe/H] = 0.002$ and the gravitational tidal field was not taken into account. The output data files have been printed every $t = 0.201$ Myr, and contain the mass, position coordinates (x, y, z) and velocity components (v_x, v_y, v_z) of each particle at every time-step. The total number of stars is $N = 6812$, with 341 binary stars provided by the user. This binaries are automatically merged in a single particle with total mass $m_{bin} = m_1 + m_2$, position centered in the center of mass of the system and with velocity equal at the one of the center of mass. At the first time-step computed by the code the stars in binary systems (not provided by the user) are automatically detected and merged together in a single particle.

2.2.2 *PETAR*

NBODY6++GPU made it possible to directly simulate million-body models for star clusters. However this models have relatively lower densities compared to the ones observed in globular clusters; it is difficult to simulate high density systems because of the short relaxation time scale that lead to a significantly increase of the computational time. Moreover the orbital integration of binary stars is not parallelized in NBODY6++GPU, so it must take into account a smaller fraction of binaries compared to the one observed in the globular clusters in order to not increase too much the computational time.

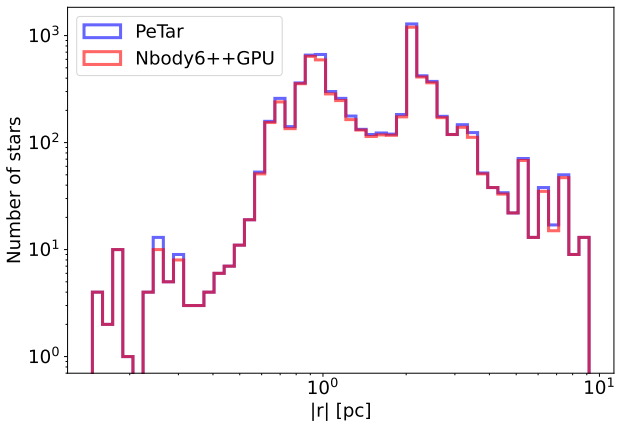

Figure 3. Distribution of the distance from the center module at the initial time = 0 Myr, computed from the NBODY6++GPU (red line) and PETAR output file 1 (blue line).

PETAR[3] (Wang et al. 2020) has been developed to overcome this bottlenecks in order to handle an arbitrary number of multiple systems while keeping high performance. This code implements the hybrid parallelization and combines the methods of Hermite fourth order integration with block time-steps, the Barnes-Hut tree and the slow-down algorithmic regularization via the P^3T (Particle Particle Particle Tree) (Oshino et al. 2011). Furthermore, PETAR mixes the N-body integration and the stellar evolution (it can be coupled with a population synthesis code like MOBSE (Giacobbo & Mapelli 2018)) in a less complex way with respect to NBODY6++GPU.

For the simulation ran with PETAR the best tree time-step used in the simulation is automatically detected by the PETAR tool `petar.find.dt`, and it is $t_{pt} = 2.048 \times 10^{-5}$ Myr, the metallicity of all the stars has been set to $[Fe/H] = 0.002$ and the gravitational tidal field was not taken in account. The code printed the output files every 1 Myr, that contains the position coordinates, velocity coordinated and mass of each star (the stars in binary systems are taken separately); from now on I will refer to this files as **PETAR output files 1**. Then I have used the PETAR tool `petar.data.process`, that uses the PETAR output files to detect binaries and calculate Lagrangian, core radii, averaged mass and velocity dispersion of the system. The data of the single and binary stars [4] (that this tool merge together in a single particle) are stored for each snapshot in two additional data files with the suffix ".single" and ".binary". From now on I will refer to this two output files as **PETAR output files 2**.

2.3 Scripts

To analyze the outputs of the two simulations I have compiled some PYTHON3 scripts.[5] The analysis was focused on the the differences between the codes in the initial conditions, the time evolution and the presence of mass segregation in the star cluster. In the following the

[3] The PETAR code can be found in the following GitHub repository `https://github.com/lwang-astro/PeTar.git`

[4] An additional data file for the triple and quadruple systems can be created but that was not the case

[5] All the scripts used for the analysis can be found in the following GitHub repository: `https://github.com/cortese98/computational_astro_codes.git`.

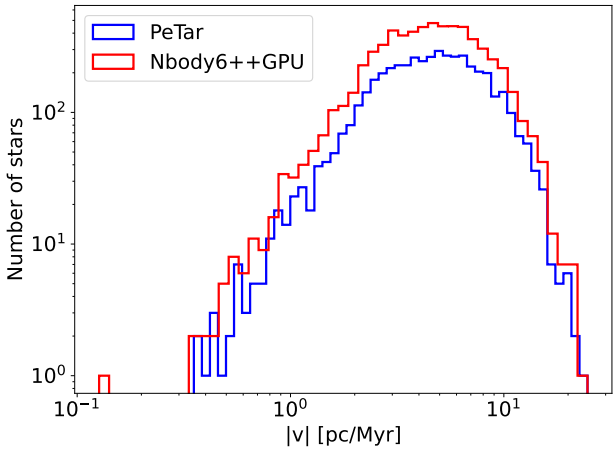

Figure 4. Initial distribution of the velocity modules, computed from the NBODY6++GPU (red line) and PETAR output file 2 (blue line).

scripts used are described. The results of the analysis are reported in the section 3.

2.3.1 *Initial conditions comparison*

To analyze if there are systematic differences between the initial conditions taken into account by PETAR and NBODY6++GPU I have plotted the initial mass, position and velocity distributions of all the particles (the scripts that I used are `mass_distribution.py`, `positions_distribution.py`, `velocity_distribution.py`).

The plot of the particles mass distribution can be found in Figure 2. In that case I have compared the mass distribution from the PETAR output file 1 with the NBODY6++GPU one (top panel) and then I have compared the masses of the particles from the output files 1 and 2 of PETAR (bottom panel).

The plot of the stars position distribution can be found in the Figure 3. The distribution refers to the module of the distance from the center $|r|$, computed for every particle as:

$$|r_i| = \sqrt{x_i^2 + y_i^2 + z_i^2},$$

where x_i, y_i and z_i are the position coordinates of the i particle. To compute this distribution I have used the PETAR output file 1.

The plot of the particles velocity distribution can be found in the Figure 4. The distribution refers to the module of the total velocity of each particle $|v|$, computed as:

$$|v_i| = \sqrt{v_{x_i}^2 + v_{y_i}^2 + v_{z_i}^2},$$

where v_{x_i}, v_{y_i} and v_{z_i} are the velocity components of the i particle. For PETAR the output file 2 has been used, in order to consider for the binary systems the velocity of the center of mass and not the velocity of the single stars with respect to it. The velocities of NBODY6++GPU have been multiplied by a conversion factor $c = 1.02271216504569$. This is done to uniform the units of measure of the two codes.

2.3.2 *Density center*

Since the star cluster is not monolithic, but it presents two initial sub-clusters (Figure 1), the position and velocity coordinates of the

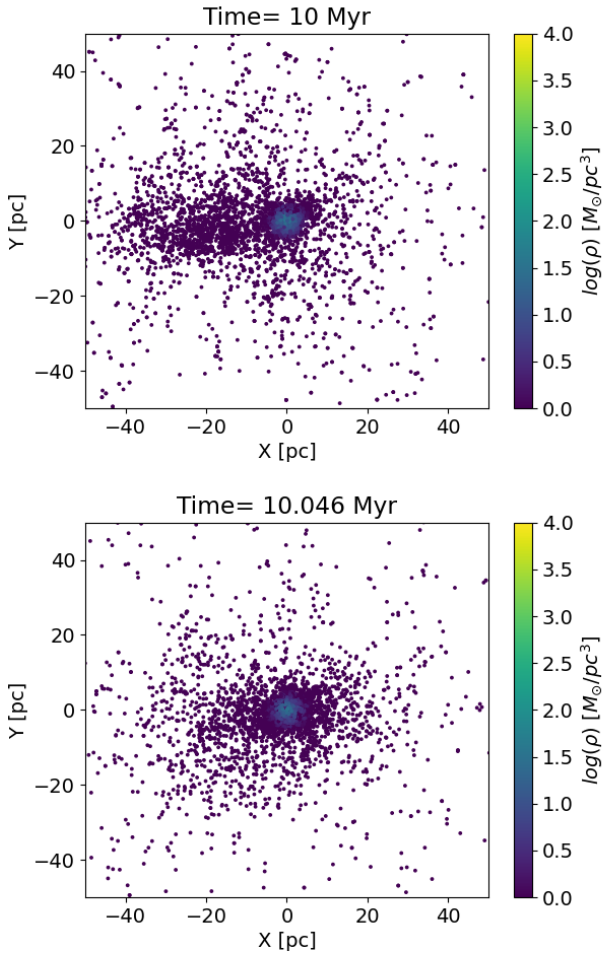

Figure 5. Positions of the particles in the $x-y$ plane (both in parsec) obtained with PETAR output files 1 (top panel) and NBODY6++GPU (bottom panel) at the time $t = 10$ Myr. The colors correspond to the density calculated at the particle's position neighborhood, expressed in $\log(M_\odot/pc^3)$

.

particles must be centered with respect to the density center of the cluster. To do that I have used two functions provided by Dr. Stefano Torniamenti (that can be found in the file `modules.py`). The first function evaluates the density around each particle, considering the volume of a sphere that contains the closest 500 particles to the one taken into account and the total mass contained in that sphere. Then, with the density obtained, the second function calculates the center of density of the cluster and rescales the positions and velocities of the particles with respect to it.

This two functions have been applied to all the scripts that will be presented in the next paragraphs. Therefore from now on all the positions and velocities are referred to the ones rescaled with respect to the density center of the star cluster.

2.3.3 *Positions and Velocities*

To visualize the time evolution of the star cluster computed by the two N-body codes, the positions on the $x-y$ plane and the velocities on the $v_x - v_y$ plane have been plotted for every time-step with four python scripts (`positions_Petar.py`, `positions_Nbody.py`, `velocity_PeTar`, `velocity_Nbody.py`). For the positions plots

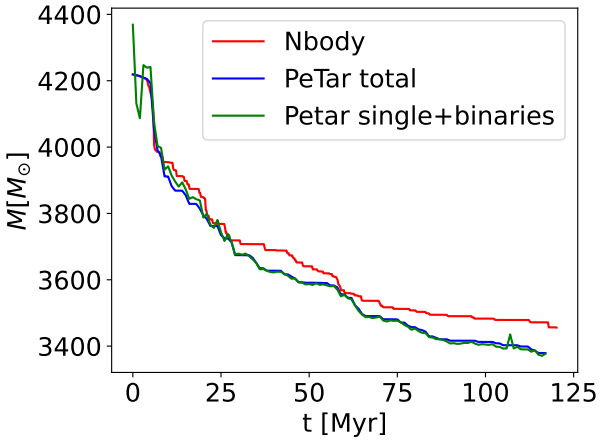

Figure 6. Total mass of the star cluster (expressed in solar masses) in function of the time computed by NBODY6++GPU (green line) and PETAR from the output files 1 (blue) and the output files 2 (green)

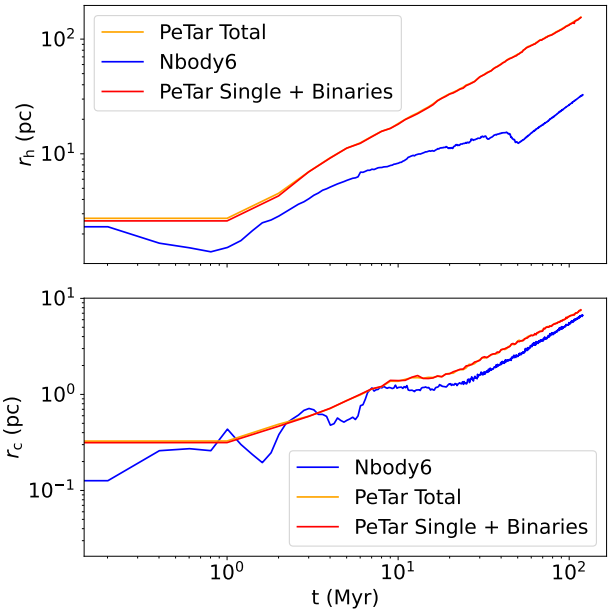

Figure 7. Time evolution of the half-mass (top panel) and the core radius (bottom panel). The blue lines correspond to the radii obtained with the outputs of NBODY6++GPU, the orange lines from the output files 1 of PETAR and the red ones from the output files 2.

the color of the particles has been associated to the local density in their neighborhood, while for the velocities plots the color associated corresponds to the mass of the particle. The plots of the positions obtained at $t = 10$ Myr can be found in the Figure 5. Then with the python module MOVIEPY all the images relative to the positions and the velocities at each time-step have been merged together in movies [6]. For the positions I have produced the same plots and movies both in a region of 50 pc $\times$ 50 pc and a more zoomed region of 5 pc $\times$ 5 pc, to visualize better what happens in the central region of the cluster.

2.3.4 *Time evolution*

To compare the time evolution between the two codes I have firstly plotted the total mass of the cluster in function of the time (the scripts that I used are `total_mass_Nbody.py`, `total_mass_PeTar.py`, `total_mass_compare.py`). For PETAR the total mass has been computed both with the output files 1 and 2. The resulting plot can be found in the Figure 6.

Then, the evolution in time of the half-mass and core radius have been analysed (the scripts used are `analysis_radii_Nbody6.py`, `analysis_radii_PeTar.py`, `compare_radii.py`). These radii have been derived by calculating at every time-step the lagrangian radius of the cluster, that is the radius of an ideal sphere centered in the density center of the cluster that contains a fixed proportion of its total mass, and fixing the amount of mass at 10% of the cluster total mass for the core radius and 50% for the half-mass radius respectively. The function that contains the calculation of the lagrangian radius has been provided by Dr. Stefano Torniamenti and it can be found in the file `modules.py`. This analysis has been done both for the output data of NBODY6++GPU and from PETAR. Moreover, this calculation has been done with both the output files 1 and 2 of PETAR to compare the results. At the end the evolution of the two radii have been plotted and it can be found in the Figure 7.

[6] The movies can be found in folder *Positions_Velocities_movie* of the scripts github repository.

In addition to that, I have plotted the evolution in time of the half-mass and the core radius using the tool implemented in the PETAR code to make this kind of analysis (the script used is `rh_rc_Petar_tool.py`). The resulting plot is reported in the Figure A1 of the appendix. However, this plot has not been considered in the results because the data used by this tool are rescaled for the density center of the cluster, so they are have not been taken into account in the results.

2.3.5 *Mass segregation*

I have analysed also if the simulations ran with the two different direct N-body codes show mass segregation, a phenomena caused by the dynamical friction. This friction is the effect of a body with high mass M that travels with velocity V_0 in a sea of low mass bodies; this bodies are gravitationally attracted by the heavy one, but when the light particles approach it, the heavy body already moved away, causing a local overdensity behind it; this overdensity attracts the heavy body with a force that slows it down. The timescale of this dynamical friction is directly proportional to the relaxation timescale (Chandrasekhar 1942) because the driver is the same: the gravitational encounters. One of the effects of the dynamical friction is that the most massive stars slow down and sink to the center of the star cluster (mass segregation).

To study if the mass segregation is present in this cluster, at every time-step I have calculated from the NBODY6++GPU and PETAR output files (for PETAR I have used the output files 1), the 20% lagrangian radius (i.e. the lagrangian radius that contain the 20% of the total mass considered) of the stars with mass lower then 10 $M_\odot$ and of the ones with mass higher then 10 $M_\odot$. I have plotted then the time evolution of this radii and it can be found in the Figure 8 (the scripts that I have used are `mass_segregation_Petar_TOTAL.py`,

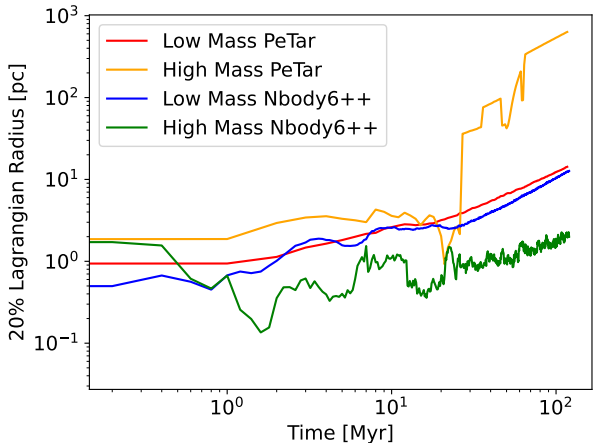

Figure 8. Comparison between the mass segregation obtained from PeTar (output files 1) and nbody6++gpu outputs. The 20% of lagrangian radius is plotted in function of the time. The blue and the red lines are the radii obtained with the low mass stars ($M < 10$ M$_\odot$) of PeTar and nbody6++gpu respectively. The orange and green lines are the radii obtained with the the high mass stars ($M \geq 10$ M$_\odot$) of PeTar and nbody6++gpu one respectively.

`mass_segregation_Nbody.py`, `segregation_compare.py`).

I have also compared the mass segregation obtained with PeTar output files 1 and 2. The resulting plot can be found in Figure A2 (the scripts that I have used are `mass_segregation_Petar_TOTAL.py`, `mass_segregation_Petar.py`, `segregation_compare_Petar.py`).

3 RESULTS

In this section I am going to report the results of the analysis done with the python scripts to compare the outputs of the simulations ran with the two direct N-body codes.

3.1 Initial conditions

As it can be seen from the particles distributions (Figures 2, 3 and 4), the initial conditions of the stellar cluster used by the two codes do not show systematic differences and are compatible between them. The differences in the masses distribution plotted in the top panel of the Figure 2 are present because PeTar considers the total stars as single particles, while nbody6++gpu merge together the 341 user-provided binaries; small differences can be caused also by a different unit of measure used for the mass. Moreover, the initial total mass of the cluster is the same for the two codes, as can be seen in the Figure 6. However, in this plot it can be seen that there is a difference between the initial total mass computed by PeTar output file 1 (blue line) and the one obtained from the output file 2. This difference is $|\Delta_M| \sim 150$ M$_\odot$. Furthermore, it can be seen that in the masses distribution plot of PeTar (bottom panel of Figure 2), there are no bins for mass higher then 10^2 M$_\odot$ in the distribution of the stars of output file 2, while it is present both for nbody6++gpu and for the PeTar output file 1.

3.2 Time evolution

The evolution of the cluster total mass is different between the simulation ran with the two codes, as it can be seen in Figure 6. Indeed, with PeTar the cluster has a final mass at $t = 117$ Myr of $M_{f_P} = 3379$ M$_\odot$, while with nbody6++gpu the final total mass is $M_{f_N} = 3472$ M$_\odot$. The total mass evolution is also different between the two outputs of PeTar. The difference $\Delta_{m_{1-2}}$ between the two kind of outputs varies over the time and it assumes both positive and negative values. This discrepancy will be discussed in the Section 4.

By looking at the movies that show the evolution of the particles positions in the $x - y$ plane, and from the time snap at $t = 10$ Myr reported in the Figure 5, it can be seen that the two initial dense clumps merge together into a single one. However, while in the nbody6++gpu simulation this central cluster evolve as a single object, in the PeTar a low-density sub-clump of stars detaches from the density center and it breaks down after about $t \sim 10$ Myr.

The evolution of the core radius (bottom panel of Figure 7 is almost the same between the two codes (the difference at $t = 117$ Myr is $\Delta_{r_c} \simeq 1$ pc); the difference between the starting points of the core radius evolution is probably caused by the different amount of binary stars detected at $t = 0$ by the two codes. Furthermore, this may explain also the differences at $t = 0$ of the half mass radius (top panel of figure 7) and of the lagrangian radii in the mass segregation plots (Figures 8 and A2). The evolution of the half mass radius (top panel of the Figure 7) is very different between nbody6++gpu and PeTar. The difference between the two half-mass radii at $t = 117$ Myr is $\Delta_{r_h} \simeq 120$ pc. This difference will be discussed in the Section 4.

3.3 Mass segregation

The resulting plot with the results of the mass segregation analysis is reported in the Figure 8. While the 20 % of the lagrangian radius is almost the same for the low mass stars ($M < 10$ M$_\odot$), for the high mass stars ($M \geq 10$ M$_\odot$) the evolution is completely different. The radius computed from the nbody6++gpu simulation output drops at $t \sim 1$ Myr and evolves towards values that are lower with respect to the radius of the low mass stars; on the other hand for the one computed from the PeTar output files 1, after a drop below the radius of the low mass stars, it diverges up to a value of $r \sim 650$ pc. This trend is the same (but with some differences) considering both the output files 1 and 2 of PeTar, as it can be seen in Figure A2.

4 DISCUSSIONS

In this section I am going to discuss the main differences found in the analysis of the outputs of the simulation ran with the two codes.

4.1 Total mass of the star cluster

The initial total mass (Figure 6) and the initial particles mass distribution (Figure 2) computed from the output of the two codes are consistent. However, the evolution of the cluster total mass is different between the two, as it can be seen from the Figure 6; this discrepancy suggests that the two codes have computed the star cluster evolution in different ways.

On the other hands the differences between the PeTar output files 1 and 2 are relevant. As reported in the Section 3.2, there are discrepancies between the total mass computed by the two output

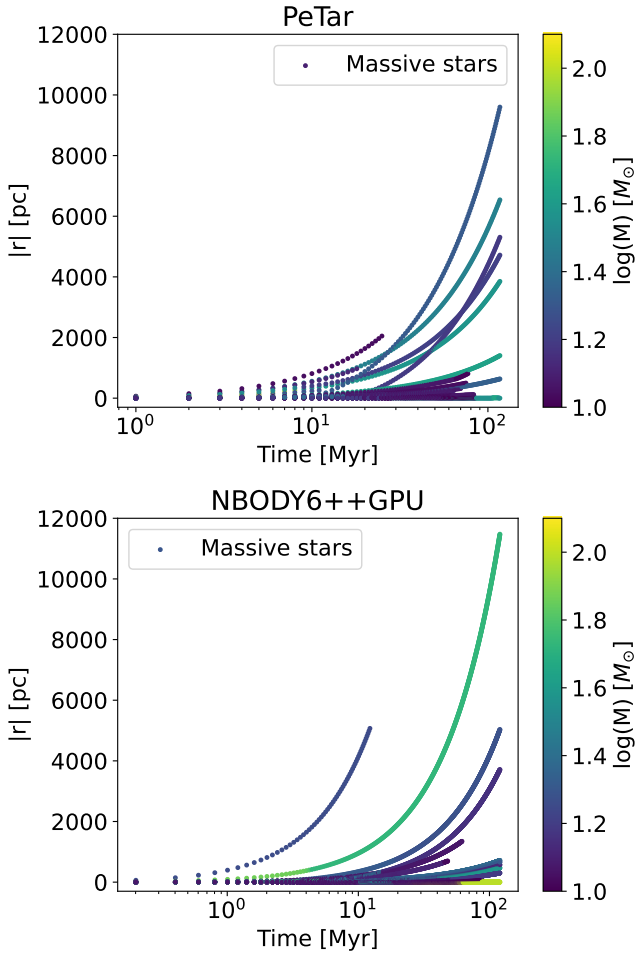

Figure 9. Evolution in time of the massive stars ($M \geq 10\ M_\odot$) distance from the density center of the cluster $|r|$ computed for the NBODY6++GPU (bottom panel) and PETAR (top panel). The color of the points correspond to the mass of the star expressed in $M_\odot$. For PETAR the output file 2 has been used.

files. This differences may be caused by some errors in the tool `petar.data.process`, that splits the stars into single stars and binaries. To support this hypothesis I have plotted with a python script (`particles_Petar.py`) the evolution in time of the number of stars contained in the PETAR output files 1 and 2. The resulting plot can be found in Figure 10. As it can be seen, the number of particles in the output files 2 is always higher with respect to the one in the output files 1, but the total mass from this output files is not always the lower one. This contradiction between the number of stars and the total mass of PETAR should be further investigated.

4.2 Differences in half-mass radius evolution and mass segregation

The evolution of the half-mass radius (top panel of Figure 7) and the high-mass stars 20% lagrangian radius (Figure 8) computed from the outputs of the two codes are very different. The high values found with PETAR may be caused by some massive stars that escapes at very high distance from the center of the cluster. To support that explanation and visualize it I have used two scripts (`massive_stars_tracks_Nbody.py`, `massive_stars_tracks_Petar.py`) that take from the outputs of

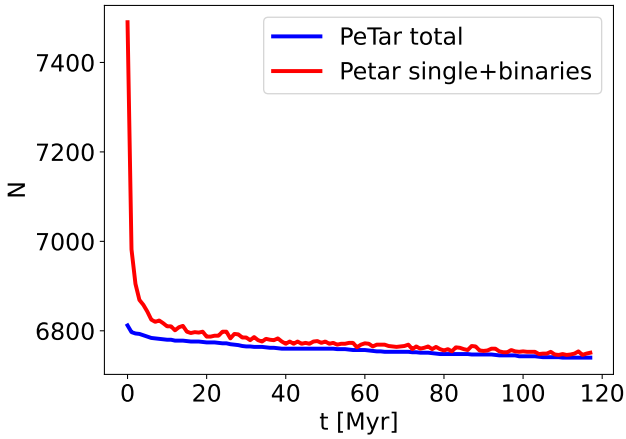

Figure 10. Time evolution of the number of particles contained in the PETAR output files 1 (blue line) and 2 (green line)

the two codes (for PETAR uses the output files 2 but the result is the same also with the output files 1) the stars with $M \geq 10\ M_\odot$ and plot their distance from the density center of the cluster $|r|$ in function of the time. The resulting plots can be found in the Figure 9. While for NBODY6++GPU at the final time there are only three stars with $|r| > 10^3$ pc, for PETAR there are more stars with distance from the center higher then 10^3 pc. This explain the different evolution of the two radii. Furthermore, the value of the half-mass radius (and the one of the high-mass stars 20% lagrangain radius) obtained with PETAR is way higher with respect to the typical values observed in star clusters (Portegies Zwart et al. 2010). To understand the reasons that led to the escape from the cluster of many high mass stars require further investigations, computing for example other simulations with the two codes to compare more outputs.

5 CONCLUSIONS

Direct N-body simulations contribute significantly to theoretically understand the star clusters evolution and dynamics. The hardware and software improvement have made it possible to increase the number of particles that can be studied and reduce the computational time required to perform the simulation while keeping an high precision in the calculations. In this report I have compared the results of two simulations performed with NBODY6++GPU and PETAR, two widely used codes that perform direct N-body simulations, starting from the same realistic initial conditions obtained from an hydrodynamic simulation of a molecular cloud. The analysis shows that the time evolution of the cluster computed with the two codes is different, as it can be seen in the analysis done on the total mass, the half-mass radius and the mass segregation (Sections 3.2 and 3.3), despite the fact that the initial conditions were the same for the two codes, as confirmed by the analysis done (Section 3.1). In Particular, the evolution of the radii mentioned before computed by PETAR shows values that are higher with respect to the typical ones observed in stellar clusters (Portegies Zwart et al. 2010); the reason for that is the escape of some massive stars from the density center of the cluster. However, further investigations are needed to understand why this happened.

Moreover, PETAR has a different way with respect to NBODY6++GPU to recognize the binary systems. However, the split-

ting of the stars in the output files "data.single" and "data.binary" performed by PeTar presents some errors (Figures 6 and 10) and this aspect should further investigated too.

More simulations with the same initial conditions must be performed to investigate the nature of the differences between the codes to improve the tools for the theoretical analysis of the stellar clusters. Understanding the nature of this differences is crucial in order to being able to improve the efficiency of the codes and their ability to simulate the evolution of stellar clusters; this will lead to a better comprehension on their dynamics, evolution and formation.

REFERENCES

Aarseth S. J., 1985, in Goodman J., Hut P., eds, Vol. 113, Dynamics of Star Clusters. pp 251–258

Aarseth S. J., 2003, Gravitational N-Body Simulations

Ballone A., Mapelli M., Di Carlo U. N., Torniamenti S., Spera M., Rastello S., 2020, MNRAS, 496, 49

Ballone A., Torniamenti S., Mapelli M., Di Carlo U. N., Spera M., Rastello S., Gaspari N., Iorio G., 2021, MNRAS, 501, 2920

Barnes J., Hut P., 1986, Nature, 324, 446

Belleman R. G., Bédorf J., Portegies Zwart S. F., 2008, New Astron., 13, 103

Chandrasekhar S., 1942, Principles of stellar dynamics

Fujii M. S., Wang L., Hirai Y., Shimajiri Y., Saitoh T., 2021, arXiv e-prints, p. arXiv:2111.15154

Giacobbo N., Mapelli M., 2018, MNRAS, 480, 2011

Oshino S., Funato Y., Makino J., 2011, PASJ, 63, 881

Plummer H. C., 1911, MNRAS, 71, 460

Portegies Zwart S. F., McMillan S. L. W., Gieles M., 2010, ARA&A, 48, 431

Torniamenti S., Ballone A., Mapelli M., Gaspari N., Di Carlo U. N., Rastello S., Giacobbo N., Pasquato M., 2021, MNRAS, 507, 2253

Wadsley J. W., Stadel J., Quinn T., 2004, New Astron., 9, 137

Wang L., Spurzem R., Aarseth S., Nitadori K., Berczik P., Kouwenhoven M. B. N., Naab T., 2015, MNRAS, 450, 4070

Wang L., Iwasawa M., Nitadori K., Makino J., 2020, MNRAS, 497, 536

APPENDIX A: EXTRA PLOTS

In this appendix some extra plots produced in the analysis of the outputs of the two codes are reported (Figures A2, A1).

This paper has been typeset from a TEX/LATEX file prepared by the author.

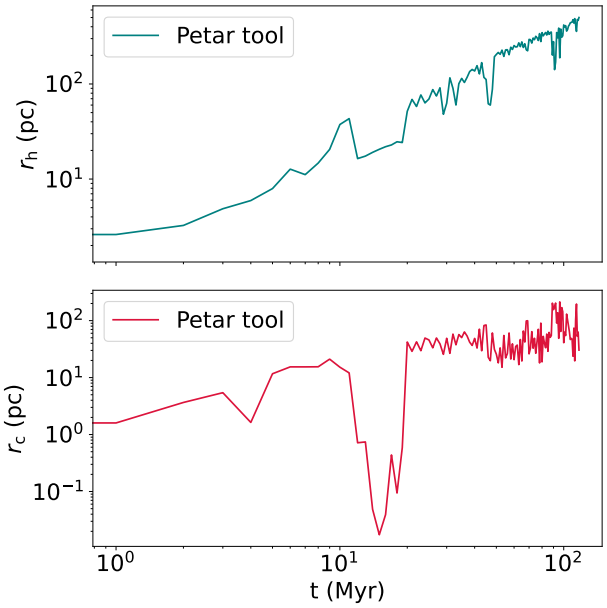

Figure A1. Plot of the time evolution of half mass (top panel) and core radius (bottom panel) computed using the tool implemented in PeTar. The analysis is done without rescaling the positions of the particles with respect to the density center of the cluster

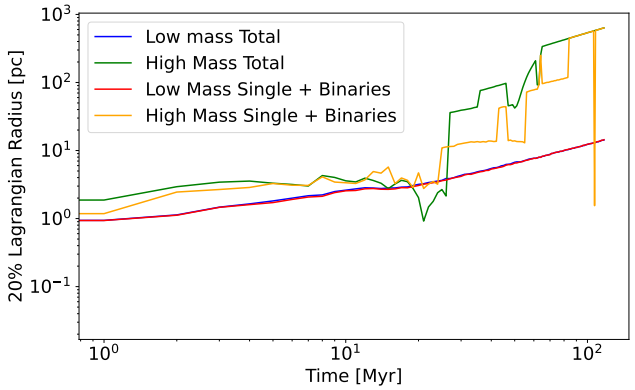

Figure A2. Comparison between the mass segregation obtained from PeTar outputs. The 20% of lagrangian radius is plotted in function of the time. The blue and the red lines are the radii of the low mass ($M < 10$ M$_\odot$) total stars and the combination of single plus binary stars respectively. The orange and green lines are the radii of the high mass ($M \geq 10$ M$_\odot$) total stars and the combination of single plus binary stars respectively.