A quick guide for *PeTar* usage

Camilla Pianta

Contents

List of Tables

List of Figures

1 Introduction

PeTar is a high performance N-body code created by Wang et al. (2020) for modeling massive collisional stellar systems such as globular clusters (GCs), but actually extendable to collisionless ones, like open clusters (OCs). Besides the significant reduction of the calculation cost, which passes from $O(N^2)$ to $O(N\ logN)$ with respect to the predecessor code *Nbody6++GPU*, the most important novelty of *PeTar* is the possibility to perform the output data analysis by means of Python3, which makes it much easier to handle.
In the following, a quick guide for *PeTar* usage will be provided, and the results of a test simulation reported, together with an example of Python3 program for the related output data analysis.

2 Initial conditions set up

The generation of initial conditions is the first, important requirement for *PeTar* to run a simulation. Here two Python3 programs, specifically designed to be compatible with *PeTar* and differing only in the way binary stars are coupled, are presented.

2.1 Stellar masses and binary pairing

Stellar masses are sampled from the Kroupa IMF (Kroupa, 2001) in the interval [m_{min},m_{max}] $M_\odot$, i.e.,

$$f(m) \propto m^{-\alpha}, \text{ with } \begin{cases} \alpha = 0.3, \text{ for } m_{min} \leq m/M_\odot < 0.08, \\ \alpha = 1.3, \text{ for } 0.08 \leq m/M_\odot < 0.5, \\ \alpha = 2.3, \text{ for } 0.5 \leq m/M_\odot \leq m_{max}, \end{cases} \tag{1}$$

where the normalization constants are such to give a matching of the two power laws passing from a mass interval to the adjacent.
In the total number of stars in the system, N_{tot}, the binary fraction is defined as $f_b = N_b/N_{tot}$, where N_b is the number of binary pairs. Consequently, $N_{tot} = N_s + 2N_b$ and $N = N_s + N_b$, where N_s is the number of single stars. In case of random pairing, N_{tot} mass values are extracted from the Kroupa IMF, of which N_s are assigned to single stars and the other $2N_b$ to binaries, hence randomly paired with the most massive member designated as the primary star (m_1) and the lightest as the secondary (m_2). If a power-law mass ratio distribution

$$f(q) \propto q^{-0.4} \tag{2}$$

(Kouwenhoven and de Grijs, 2008), where $q = m_2/m_1$, with extremes $q_{min} = 0.1$ and $q_{max} = 1$, instead, the number of sampled mass values is N, of which N_s are assigned to single stars and the other N_b to binary primary components, so that $m_1 q$ yields the mass of secondaries. Of course, $m_b = m_1 + m_2$ represents the mass of the binary. Finally, in

order for the total mass of the system $M = \sum_{i=1}^{N_{tot}} m_i$ to equal the expected total mass $M_{tot} = N_{tot} m_{mean}$, where m_{mean} is the mean mass in the Kroupa IMF selected interval, the correction M/M_{tot} is applied to all masses.

2.2 Plummer model for positions and velocities

Positions and velocities of both single stars and binary centers of mass are randomly sampled from the Plummer model according to the algorithm proposed by Aarseth et al. (1974).
Radial positions are given by

$$r = \frac{R}{\sqrt{X_1^{-\frac{2}{3}} - 1}} \ , \tag{3}$$

and the corresponding position vector components are

$$\begin{aligned} x &= \sqrt{r^2 - z^2} \ \cos\left(2\pi X_3\right) \ , \\ y &= \sqrt{r^2 - z^2} \ \sin\left(2\pi X_3\right) \ , \\ z &= \left(1 - 2X_2\right) \ r \ , \end{aligned} \tag{4}$$

where X_1, X_2, X_3 are three random numbers in the interval [0,1]. The first N_s radial position vectors have been attributed to single stars ($\mathbf{r}_s$, with components $x_s,\ y_s,\ z_s$), and the remaining N_b ones to binary centers of mass ($\mathbf{r}_b$, with components $x_b,\ y_b,\ z_b$). To obtain the components of the velocity vectors an accept-reject procedure has been employed, respecting the cut to the escape velocity at each position $\mathbf{r}$, i.e.,

$$v_{esc} = \sqrt{2U(r)} \ , \tag{5}$$

where $U(r)$ is the Plummer's potential at a distance r from the center. The velocity components are

$$\begin{aligned} v_x &= \left(1 - 2X_4\right) \ v \ , \\ v_y &= \sqrt{v^2 - v_x^2} \ \sin\left(2\pi X_5\right) \ , \\ v_z &= \sqrt{v^2 - v_x^2} \ \cos\left(2\pi X_5\right) \ , \end{aligned} \tag{6}$$

where X_4, X_5 are two random numbers in the interval [0,1]. Their units are, of course, those picked for the absolute value of the velocity v. Therefore, as in the case of positions, the first N_s radial velocity vectors have been assigned to single stars ($\mathbf{v}_s$, with components $v_{x,s},\ v_{y,s},\ v_{z,s}$), and the other N_b ones to binary centers of mass ($\mathbf{v}_b$, with components $v_{x,b},\ v_{y,b},\ v_{z,b}$). As a final operation, velocities are converted from $\mathrm{kms}^{-1}$to pc/Myr.
By definition, the Plummer model represents a system in virial equilibrium, so that the virial ratio results

$$Q = \frac{T}{|U|} = \frac{\sum_{i=1}^{N_s} \frac{m_{s,i} v_{s,i}^2}{2} + \sum_{i=1}^{N_b} \frac{m_{b,i} v_{b,i}^2}{2}}{\left| -G \sum_{i,j=1,\ i \neq j}^{N_s+N_b} \frac{m_i m_j}{|\mathbf{r}_i - \mathbf{r}_j|} \right|} = \frac{1}{2} \ . \tag{7}$$

2.3 Binary orbital parameters

The binary orbital parameters, i.e., the semi-major axis a and the eccentricity e, have been determined in the following way.
The generic value of a is extracted from a flat semi-major axis distribution $f(a) \propto 1/a$ in the interval $[a_{min}, a_{max}]$ (Kroupa and Burkert, 2001), and thus obtained as

$$a = \exp\left(n_a X_a\right) + \ln\left(a_{min}\right) \ , \tag{8}$$

where X_a is random number in the interval [0,1] and

$$n_a = \ln\left(\frac{a_{max}}{a_{min}}\right) \tag{9}$$

the normalization factor.
The eccentricity, instead, is taken from a thermal distribution $k(e) = 2e$ (Jeans, 1919), yielding

$$e = \sqrt{n_e X_e + e_{min}^2} \ , \tag{10}$$

where, as above, X_e is a random number in the interval [0,1] and

$$n_e = e_{max}^2 - e_{min}^2 \tag{11}$$

the normalization factor.

2.4 Binary positions and velocities

Both the positions and the velocities of the $2N_b$ binary components are evaluated in the center of mass reference frame and by opting for a configuration in which the secondaries are at the apocenter of the orbit of the binary system they belong to, whereas the primaries are integral with their associated center of mass.
Thereby, given the apocenter radius and the orbital velocity moduli

$$\begin{aligned} r_{apo} &= a(1+e) \ , \\ v_{orb} &= \sqrt{\frac{G m_b}{a}} \ , \end{aligned} \tag{12}$$

the components of the corresponding vectors are calculated by means of a linear transformation to map random numbers from the interval [0,1] to the interval [-1,1]. In this way, the position and velocity vectors of primaries result

$$\begin{aligned} \mathbf{r}_1 &= \mathbf{r}_b + \frac{m_2}{m_b}\,\mathbf{r}_{apo}\ , \\ \mathbf{v}_1 &= \mathbf{v}_b + \frac{m_2}{m_b}\,\mathbf{v}_{orb}\ , \end{aligned} \tag{13}$$

whereas those of secondaries are

$$\begin{aligned} \mathbf{r}_2 &= \mathbf{r}_b - \frac{m_1}{m_b}\,\mathbf{r}_{apo}\ , \\ \mathbf{v}_2 &= \mathbf{v}_b - \frac{m_1}{m_b}\,\mathbf{v}_{orb}\ . \end{aligned} \tag{14}$$

2.5 Initial conditions file

The initial conditions of all the stars in the system, i.e., masses, positions and velocities, must be reported in a file composed by 7 columns and N_{tot} lines. In order, columns contain:

- masses in units of $M_\odot$;
- position components (x, y, z) in units of pc;
- velocity components (v_x, v_y, v_z) in units of pc/Myr.

On the other hand, lines are occupied first by binary components (primary and secondary of each pair), and then by single stars.

3 Initial conditions generation programs

Here is a list of code functions, with the related purpose and returned variables (see Tab. 1 for a summary).

- ***Kroupa IMF***
  It generates stellar masses according to the Kroupa IMF (Eq. 1). It returns *masses* (1D array, float type), i.e., the masses of all the stars in the system in $M_\odot$.
- ***mass ratio***
  It extracts q values from a power-law mass ratio distribution according to Eq. 2, thus returning the mass ratio q (1D array, float type).
- ***stellar masses*** **(random pairing case)**
  It assigns the masses extracted from the Kroupa IMF to single stars and binary systems, which are then randomly paired to generate primary and secondary components. All masses are finally corrected for their sum to match the imposed system's total mass. The function returns, after correction:

1. m_s: masses of single stars in $M_\odot$ (1D array, float type);
2. m_1: masses of primaries in $M_\odot$ (1D array, float type);
3. m_2: masses of secondaries in $M_\odot$ (1D array, float type);
4. m_{cm}: masses of binary centers of mass in $M_\odot$ (1D array, float type);
5. m: masses of single stars and binary centers of mass in $M_\odot$ (1D array, float type);
6. m_{tot}: masses of single stars, primaries and secondaries in $M_\odot$ (1D array, float type);
7. M: system's total mass in $M_\odot$ (float type variable).

- ***stellar masses*** **(pairing through mass ratio distribution case)**
  It assigns the masses extracted from the Kroupa IMF to single stars and primaries, which are then coupled with a given mass rati to generate secondaries. All masses are finally corrected for their sum to match the imposed system's total mass. The function returns, after correction:
  1. m_s: masses of single stars in $M_\odot$ (1D array, float type);
  2. m_1: masses of primaries in $M_\odot$ (1D array, float type);
  3. m_2: masses of secondaries in $M_\odot$ (1D array, float type);
  4. m_{cm}: masses of binary centers of mass in $M_\odot$ (1D array, float type);
  5. m: masses of single stars and binary centers of mass in $M_\odot$ (1D array, float type);
  6. m_{tot}: masses of single stars, primaries and secondaries in $M_\odot$ (1D array, float type);
  7. M: system's total mass in $M_\odot$ (float type variable).

- ***positions***
  It determines the positions of single stars and binary centers of mass according to Eq. 3, and returns:
  1. r: position vectors' magnitudes in pc (1D array, float type);
  2. X: position vectors' components in pc ($N \times 3$ matrix, each line referring to a star and each column to a position component, i.e., x, y, z).

- ***single positions***
  It assigns the extracted positions to single stars; therefore, it returns single stellar position vectors' components in pc (X_s: $N_s \times 3$ matrix, each line referring to a star and each column to a position component, i.e., x_s, y_s, z_s).

- ***binary positions***
  It determines the positions of both binary centers of mass and binary components, and returns:

1. X_1: position vectors' components of primaries in pc ($N_b \times 3$ matrix, each line referring to a primary star and each column to a position component, i.e., x_1, y_1, z_1);
2. X_2 : position vectors' components of secondaries in pc ($N_b \times 3$ matrix, each line referring to a secondary star and each column to a position component, i.e., x_2, y_2, z_2).

- ***velocities***
  It extracts the velocities of both single stars and binary centers of mass according to Eq. 6, and returns:
  1. v: velocity vectors' magnitudes in pc/Myr (1D array, float type);
  2. V: velocity vectors' components in pc/Myr ($N \times 3$ matrix, each line referring to a star and each column to a position component, i.e., v_x, v_y, v_z).

- ***single velocities***
  It assigns the extracted velocities to single stars and returns single stellar velocity vectors' components in km/s (V_s: $N_s \times 3$ matrix, each line referring to a star and each column to a velocity component, i.e., $v_{s,x}$, $v_{s,y}$, $v_{s,z}$).

- ***binary velocities***
  It assigns the remaining extracted velocities to binary centers of mass and determines the velocity of primaries and secondaries by accounting for the additional orbital motion. The function returns:
  1. V_1: velocity vectors' components of primaries in pc/Myr ($N_b \times 3$ matrix, each line referring to a star and each column to a velocity component, i.e., $v_{1,x}$, $v_{1,y}$, $v_{1,z}$);
  2. V_2: velocity vectors' components of primaries in pc/Myr ($N_b \times 3$ matrix, each line referring to a star and each column to a velocity component, i.e., $v_{2,x}$, $v_{2,y}$, $v_{2,z}$).

- ***virial ratio***
  It calculates the kinetic energy T and potential energy U of the system, which are used to determine the virial ratio Q (float type). The purpose of the function is checking that the virial equilibrium is effectively obtained, i.e., that $Q = 1/2$ (Eq. 3).

- ***semi-major axis***
  It extracts binary semi-major axes from a flat distribution according to Eq. 8 in the desired range of values [$a_{au,min}$,$a_{au,max}$], and returns:
  1. a_{au}: semi-major axes in AU (1D array, float type);
  2. a: semi-major axes in pc (1D array, float type).

- ***eccentricity***
  It samples eccentricities from a thermal distribution according to Eq. 10 in the desired range of values [e_{min},e_{max}], hence returning the eccentricity e (1D array, float type).

- ***plot distributions*** **(random pairing case)**
  It displays the trend of the previously implemented distribution functions from sampled data. In order, panels show:

  1. the Kroupa IMF $f(m)$;
  2. the semi-major axis distribution $f(a)$;
  3. the eccentricity distribution $f(e)$.

- ***plot distributions*** **(pairing through mass ratio distribution case)**
  It displays the trend of the previously implemented distribution functions from sampled data. In order, panels show:

  1. the Kroupa IMF $f(m)$;
  2. the mass ratio distribution $f(q)$;
  3. the semi-major axis distribution $f(a)$;
  4. the eccentricity distribution $f(e)$.

- ***main program***
  It initializes all the functions in the program once the free parameters R, N_{tot}, f_b, m_{min}, m_{max}, m_{mean}, $a_{au,min}$, $a_{au,max}$, e_{min} and e_{max} are specified. Then, it generates a dataframe for *PeTar*'s initial conditions file.

Function name	Return
Kroupa IMF	*masses*
mass ratio	q
stellar masses	m_s, m_1, m_2, m_{cm}, m, m_{tot}, M
positions	r, X
single positions	X_s
binary positions	X_1, X_2
velocities	v, V
single velocities	V_s
binary velocities	V_1, V_2
virial ratio	Q
semi-major axis	a_{au}, a
eccentricity	e
plot distributions	plots of $f(m)$, $f(a)$, $f(e)$, $f(q)$
main program	initial conditions file

Table 1: Code functions and returned variables.

4 Output data analysis program

In the following, a program for *PeTar*'s output data analysis will be outlined in two steps: first the processing of data files concerning the values of single and binary stars' physical parameters per timestep, and then the visualization of the time evolution of the system's most relevant global quantities.

4.1 Analysis of *data.single.timestep* and *data.binary.timestep* output files

Hereafter is a list of code functions for the analysis of *data.single.timestep* and *data.binary.timestep* output files, together with the associated returned variables (see Tab. 2 for a summary).

- ***single parameters***
  This function is intended to read output files related to single stars for a given timestep. Files are labeled as *data.binary.timestep* and are organized in such a way that each line represents a star and each column a different parameter. The function returns:
  1. N_s: umber of single stars (int type);
  2. m_s: masses of single stars in $M_\odot$ (1D array, float type);
  3. X_s: position components of single stars in pc (matrix, each line representing a star and each column a position component, i.e., x_s, y_s, z_s);
  4. V_s: velocity components of single stars in pc/Myr (matrix, each line representing a star and each column a velocity component, i.e., $v_{s,x}$, $v_{s,y}$, $v_{s,z}$);

5. L_s: luminosities of single stars in $L_\odot$ (1D array, float type);
6. R_s: radii of single stars in $R_\odot$ (1D array, float type);
7. $type_s$: stellar evolutionary type (1D array, int type).

Note that the function works only if stellar evolution (i.e., *interrupt mode*) is active in the simulation with the options *bse* or *mobse*.

- ***binary parameters***
This function is intended to read output files related to binary stars for a given timestep. Files are labeled as *data.timestep.binary*, and are organized in such a way that lines contain binary components of each pair (i.e., primary and secondary of each pair) and columns the parameters associated to the barycenter and the components of each pair. In particular, *binary.p1* allows to access the parameters of primaries, whereas *binary.p2* those of secondaries. The function returns:

1. N_b: number of binaries (int type);
2. X_{cm}: position components of binary centers of mass in pc (matrix, each line representing a star and each column a position component, i.e., x_{cm}, y_{cm}, z_{cm});
3. V_{cm}: velocity components of binary centers of mass in pc/Myr (matrix, each line representing a star and each column a velocity component, i.e., $v_{cm,x}$, $v_{cm,y}$, $v_{cm,z}$);
4. a: semi-major axes in pc (1D array, float type);
5. e: eccentricities (1D array, float type);
6. m_1: masses of primaries in $M_\odot$ (1D array, float type);
7. m_2: masses of secondaries in $M_\odot$ (1D array, float type);
8. X_1: position components of primaries in pc (matrix, each line representing a star and each column a velocity component, i.e., x_1, y_1, z_1);
9. X_2: position components of secondaries in pc (matrix, each line representing a star and each column a velocity component, i.e., x_2, y_2, z_2);
10. V_1: velocity components of primaries in pc/Myr (matrix, each line representing a star and each column a velocity component, i.e., $v_{1,x}$, $v_{1,y}$, $v_{1,z}$);
11. V_2: velocity components of secondaries in pc/Myr (matrix, each line representing a star and each column a velocity component, i.e., $v_{2,x}$, $v_{2,y}$, $v_{2,z}$);
12. L_1: luminosities of primaries in $L_\odot$ (1D array, float type);
13. L_2: luminosities of secondaries in $L_\odot$ (1D array, float type);
14. R_1: radii of primaries in $R_\odot$ (1D array, float type);
15. R_2: radii of secondaries in $R_\odot$ (1D array, float type);
16. $type_1$: stellar evolutionary type of primaries (1D array, int type);

17. $type_2$: stellar evolutionary type of secondaries (1D array, int type).

Note that the function works only if stellar evolution (i.e., *interrupt mode*) is active in the simulation with the options *bse* or *mobse*.

- ***stellar numbers***
  This function calculates the number of stars in the system and the binary fraction at a given timestep. Hence it returns:
  1. N: number of single stars and binary systems (int type);
  2. N_{tot}: total number of stars, i.e., number of single stars and binary components (int type);
  3. $f_{b,timestep}$: binary fraction (float type).

- ***HRD***
  Function returning, at a given timestep:
  1. L: luminosity of all the stars in the system in $L_\odot$ (1D array, float type);
  2. L_{tot}: total luminosity of the system in $L_\odot$ (float type);
  3. T: temperature of all the stars in the system in K (1D array, float type).

  The function also displays the Hertzsprung Russel Diagram (HRD) of the system if *plot=True*.

- ***get data versus time***
  This function calculates the quantities of interest per timestep; by way of example, in this case the returned variable is the binary fraction per timestep f_b (1D array, float type).

Function name	**Return**
single parameters	N_s, m_s, X_s, V_s, L_s, R_s, $type_s$
binary parameters	N_b, m_{cm}, X_{cm}, V_{cm}, a, e, m_1, m_2, X_1, X_2 V_1, V_2, L_1, L_2, R_1, R_2, $type_1$, $type_2$
stellar numbers	N, N_{tot}, $f_{b,timestep}$
HRD	L, L_{tot}, T
get data versus time	f_b

Table 2: Functions for the analysis of *data.single.timestep* and *data.binary.timestep* output files, and related returned variables.

4.2 Analysis of *data.lagr*, *data.status*, *data.core* and *data.tidal* output files

The functions listed below serve as a tool to analyze the *data.lagr*, *data.status*, *data.core* and *data.tidal* output files, which contain information about the simulated system's

global parameters and their temporal evolution (see Tab. 3 for a summary of the returned variables).

- ***evolutionary time***
  Function to read the output file *data.status*, where the values of the time in Myr, corresponding to each timestep, are stored, and returning the system evolutionary time $time_{myr}$ (1D array, float type).

- ***dynamical timescales***
  Function to compute both the crossing and the relaxation time in Myr per timestep starting from the file *data.lagr*, and returning:

  1. t_{cr}: crossing time in Myr (1D array, float type);
  2. t_{rh}: relaxation time in Myr (1D array, float type).

  Note that these are half-mass dynamical timescales.

- ***lagrangian radii***
  Function to calculate and display (if $plot = True$) the Lagrangian radii as a function of time at varying mass fraction, whose values are stored in the variable f_m (1D array, float type). The information about Lagrangian radii per timestep is contained in the output file *data.lagr*, and the attribute *.all* means that all the stars in the system are considered in the computation. The function returns:

  1. r_{lagr}: Lagrangian radii at varying mass fraction (2D array: the first dimension represents the values of the Lagrangian radii in pc per timestep, whereas the second one the mass fraction; float type);
  2. r_{hm}: half-mass radius per timestep (1D array, float type). Note that the half-mass radius corresponds to the Lagrangian radius relative to the mass fraction 0.5;
  3. r_{core}: core radius per timestep (1D array, float type).

- ***virial ratio***
  Function to calculate and display (if $plot = True$) the virial ratio as function of time, starting from the output file *data.lagr*. It hence returns the value of virial ratio Q per timestep (1D array, float type), which is computed by considering all stars inside the system's half-mass radius.

- ***energy***
  Function to calculate and display (if $plot = True$) as a function of time both the total energy of the system and the normalized difference between this and its initial value. In particular, the potential energy E_{pot} is obtained from the output file *data.lagr* and computed by considering all stars inside the system's half-mass radius, as well as the virial ratio. Also, here the user can choose between the potential energy of both the stars and the Galaxy, and the potential energy of single stars only (variable E_{pot} in the program). Thus the function returns:

1. E: total energy of the system per timestep in erg (1D array, float type);
2. ΔE_{norm}: normalized energy difference per timestep (1D array, float type).

- ***total mass***
  Function to calculate and display (if $plot = True$) the total mass of the system as a function of time, starting from the output file *data.lagr*. Therefore, it returns M (1D array, float type), i.e., the total mass of the system per timestep in $M_\odot$.

- ***center coordinates***
  Function to calculate the position and velocity coordinates of the system's center by using the output file *data.core*, and returning:
  1. X: system's center position coordinates in pc per timestep (1D array, float type);
  2. V: system's center velocity coordinates in pc/Myr per timestep (1D array, float type).

- ***tidal***
  Function to display (if $plot = True$) the evolution of the number of bound stars, as well as that of the bound mass, as a function of time, and to calculate the expected time for the system to dissolve by using a linear interpolation. It hence returns:
  1. r_{tid}: tidal radius in pc per timestep (1D array, float type);
  2. m_{bound}: total bound mass of the system in $M_\odot$ (1D array, float type);
  3. n_{bound}: number of bound stars in the system (1D array, float type);
  4. $time_{myr,end}$: expected time for system dissolution (float type).

  Note that the output file *data.tidal* exists only if *external mode* = *galpy*.

- ***binary fraction***
  Function to display (if $plot = True$) the binary fraction f_b (obtained from the *get data versus time* function) as a function of time.

Function name	Data file	Return
evolutionary time	*data.status*	$time_{myr}$
dynamical timescales	*data.lagr*	t_{cr}, t_{rh}
lagrangian radii	*data.lagr*	r_{lagr}, r_{hm}, r_{core}
virial ratio	*data.lagr*	Q
energy	*data.lagr*	E, ΔE_{norm}
total mass	*data.lagr*	M
center coordinates	*data.core*	X_{center}, V_{center}
tidal	*data.tidal*	r_{tid}, n_{bound}, m_{bound}, $time_{myr,end}$
binary fraction	None	None

Table 3: Functions to analyze the *data.lagr*, *data.status*, *data.core* and *data.tidal* output files, and related returned variables.

5 Test simulation

A test simulation have been realized considering a Milky Way (MW) OC composed by $N_{tot} = 10^3$ stars, with binaries in fraction $f_b = 0.15$ (number of binaries $N_b = 150$). The system has been modeled as a Plummer sphere of radius $R = 1$ pc, and stellar masses have been sampled from the Kroupa IMF in the interval [0.1,100] $M_\odot$, so that the mean system's mass results $m_{mean} = 0.64$ $M_\odot$. Binary stars have been paired by using Eq. 2, and their orbital parameters, i.e., semi-major axis and eccentricity, have been extracted from a flat distribution (Eq. 8) in the interval [0.2,100] AU and a thermal distribution (Eq. 10), respectively. See Tab. 4 for athe complete list of the simulation input parameters.

Parameter	Value
R	1 pc
f_b	0.15
N_{tot}	10^3
m_{min}	0.1 $M_\odot$
m_{max}	100 $M_\odot$
m_{mean}	0.64 $M_\odot$
$a_{au,min}$	0.2 AU
$a_{au,max}$	100 AU
e_{min}	0
e_{max}	1

Table 4: Input parameters for the test simulation.

The file thus created, which will be named *initial conditions* for simplicity, must stored in the same directory where the simulation is launched. Here, to this end, a

specific sequence of commands, depending on the chosen *PeTar* tools, must be given. For instance, if the *mobse* stellar evolution option is used and the duration time of the simulation is 1 Gyr, the user must:

1. generate an input file from the *initial conditions* file:

   petar.init -s mobse -f input initial conditions

2. start the simulation:

   petar - -mobse-metallicity 0.02 -u 1 -b 300 -t 1000 -o 10 input

   where *-u* denotes the units to be used, *-b* the total number of binary components, *-t* the finishing time in Myr and *-o* the output time interval in Myr;

3. activate Python3 and gather the output data when the simulation successfully ends:

   petar.data.gether data

4. process the output data to create the corresponding output files of the simulation:

   petar.data.process -i mobse - - e escape 0 - -r escape 100 - -calc energy data.snap.lst

   where *- - r escape* denotes the value of the system's escape radius in pc.

Some results of the test simulation are reported in Fig. 1, 2, 3 and 4.

References

[1] S. J. Aarseth, M. Henon, and R. Wielen. A comparison of numerical methods for the study of star cluster dynamics. , 37(1):183–187, December 1974.

[2] J. H. Jeans. The origin of binary systems. , 79:408, April 1919.

[3] M. B. N. Kouwenhoven and R. de Grijs. The effect of binaries on the dynamical mass determination of star clusters. , 480(1):103–114, March 2008.

[4] P. Kroupa and A. Burkert. On the origin of the distribution of binary star periods. , 555(2):945, jul 2001.

[5] Pavel Kroupa. On the variation of the initial mass function. , 322(2):231–246, April 2001.

[6] Long Wang, Masaki Iwasawa, Keigo Nitadori, and Junichiro Makino. PETAR: a high-performance N-body code for modelling massive collisional stellar systems. , 497(1):536–555, September 2020.

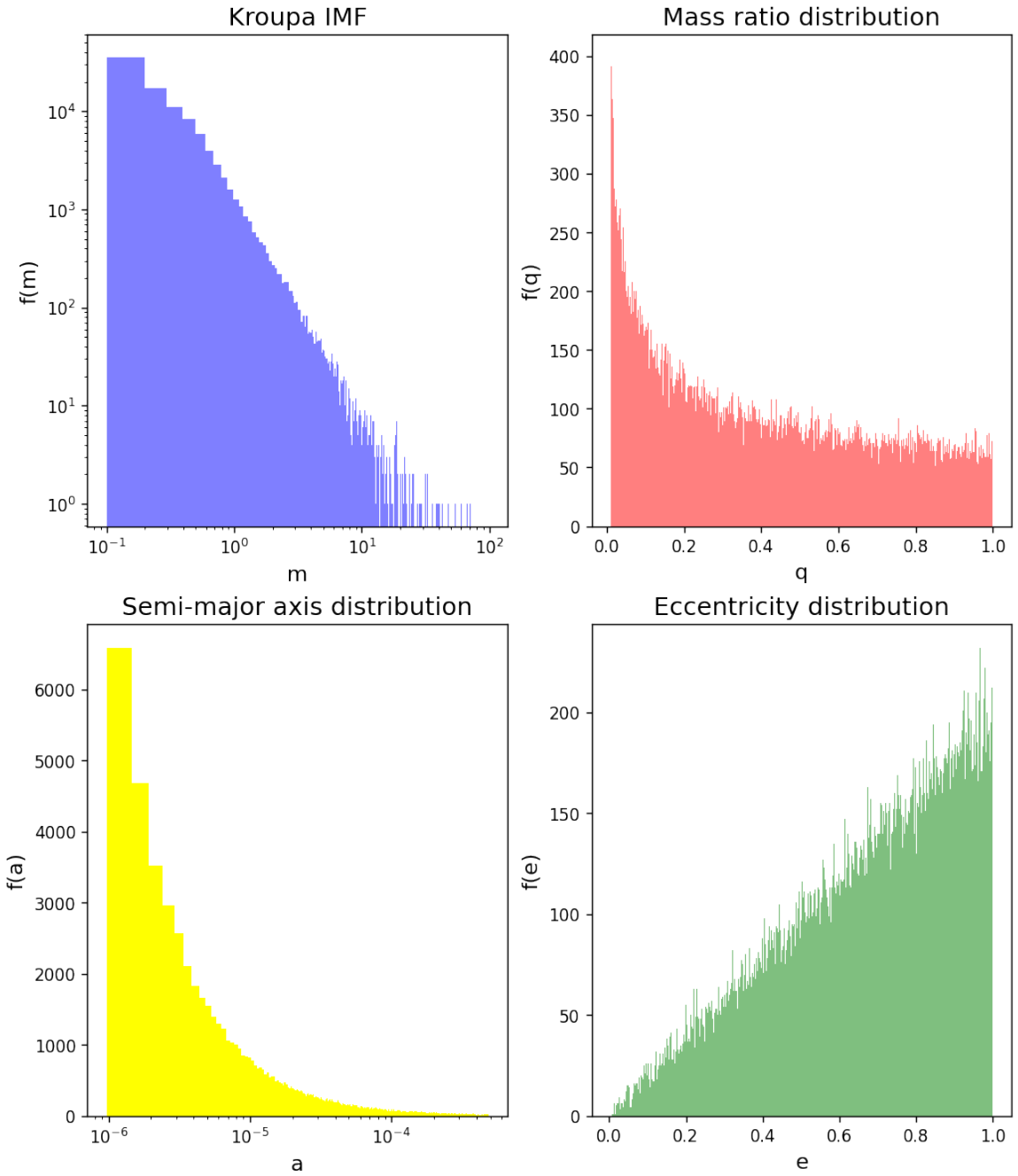

Figure 1: Distribution functions computed by using the function *plot distributions* in the initial conditions generation program: Kroupa IMF (top-left panel), mass ratio distribution (top-right panel), semi-major axis distribution (bottom-left panel) and eccentricity distribution (bottom-right panel).

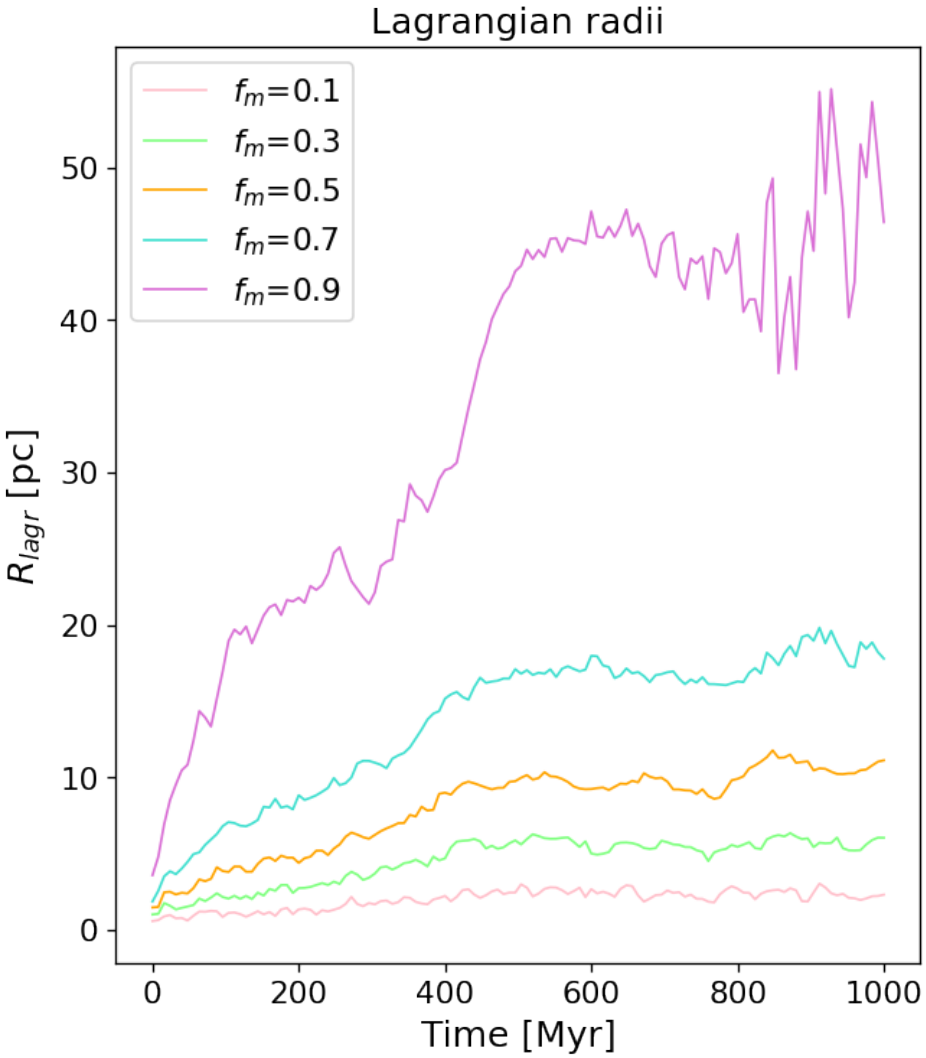

Figure 2: Lagrangian radii computed by using the function *lagrangian radii* in the output analysis program.

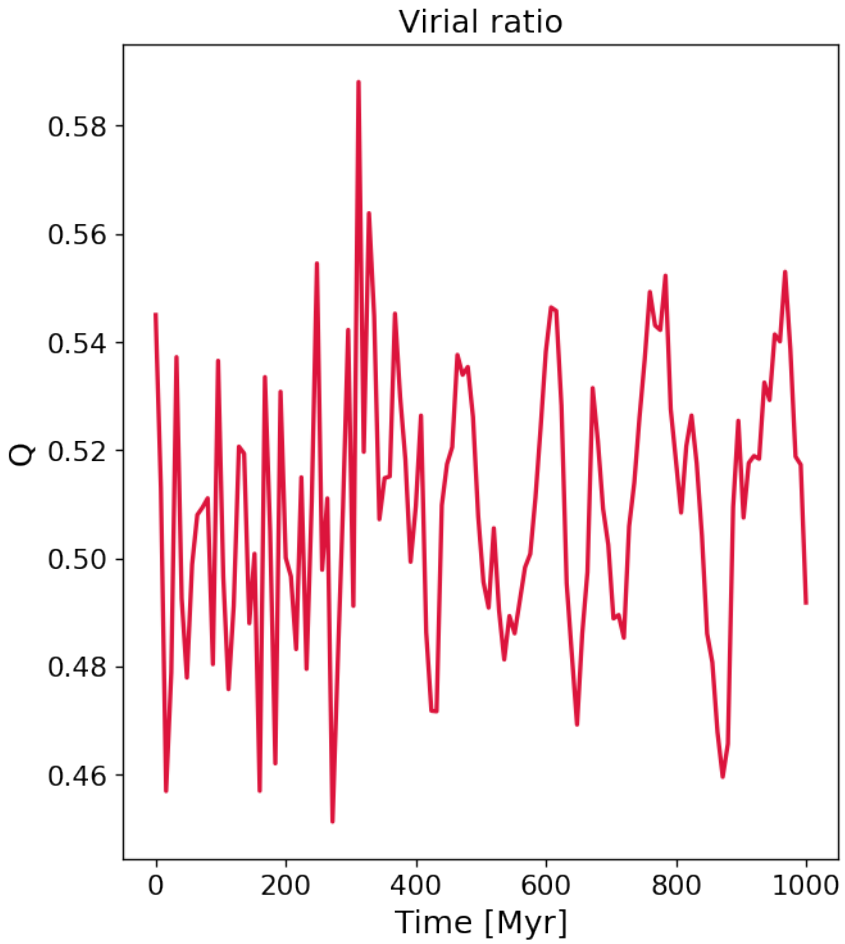

Figure 3: Virial ratio computed by using the function *virial ratio* in the output analysis program.

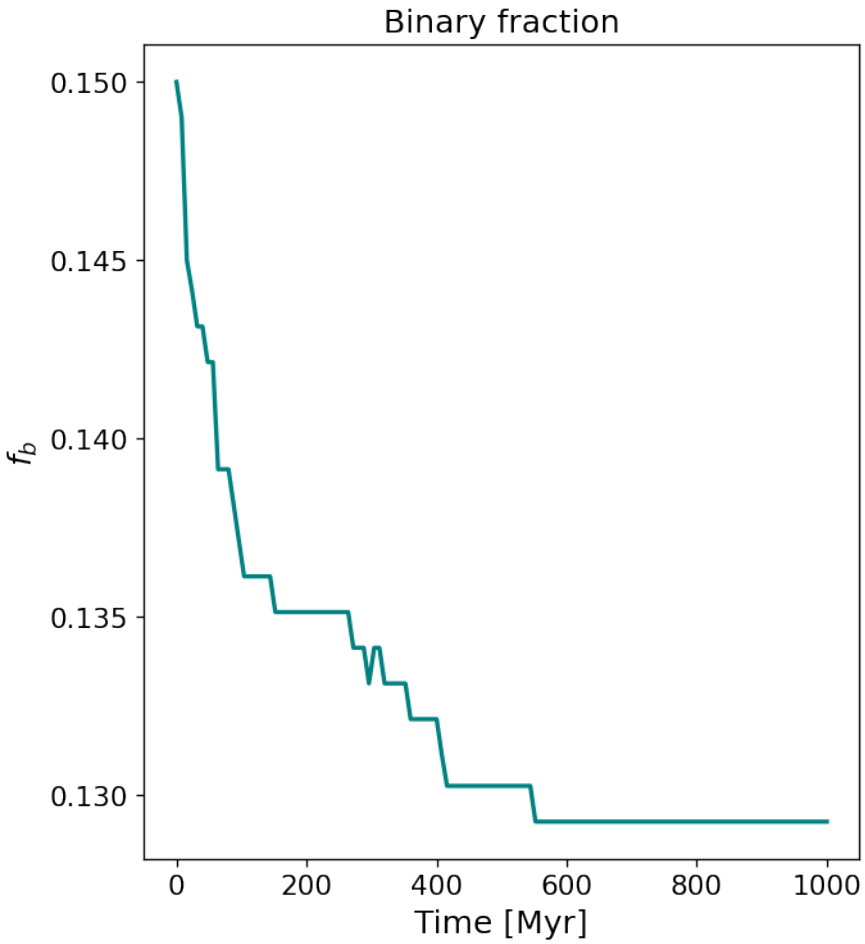

Figure 4: Binary fraction computed by using the function *get data versus time* in the output analysis program.