Raga reference

Eugene Vasiliev
University of Oxford & Institute of Astronomy, Cambridge
email: eugvas@lpi.ru

Version 3.0
February 1, 2020

Contents

1 Introduction

Raga[1] is a Monte Carlo code for dynamical evolution of self-gravitating non-spherical stellar systems. The method is described in [1]; here comes a more technical and practical guide.

This document describes the version 3 of the program. The main features in this version are the interface to the Amuse framework [3] and the possibility to deal with multimass systems. Version 2 (2017) was a nearly complete rewrite of the first version (2014–2015), although built on the same principles. It is now based on the Agama library for galaxy modelling [2] and included in its distribution. The main features are:

- Simulation of stellar systems with a much smaller number of particles N than the number of stars in the actual system $N_\star$;

[1]Relaxation in Any Geometry

- Representation of an arbitrary non-spherical potential via a Multipole expansion, with coefficients computed from particle trajectories;
- Two-body relaxation modelled by local (position-dependent) velocity diffusion coefficients (as in Spitzer's Monte Carlo formulation); the magnitude of relaxation can be adjusted to the actual number of stars in the target system and is not related to the number of particles in the simulation;
- Particle trajectories are computed independently and in parallel, using a high-accuracy adaptive-timestep integrator; the potential expansion and diffusion coefficients are updated at rather long intervals (possibly comprising many dynamical times, but much shorter than the relaxation time);
- Can model the effect of a central supermassive black hole (SMBH) – capture of low angular momentum stars, and/or a SMBH binary – scattering of stars and hardening of the binary.

In version 3, Raga comes in two flavors: as a standalone simulation program and as a component of the Amuse framework. In the former scenario, it can evolve an N-body system read from a file, optionally storing N-body snapshots and other parameters at regular intervals during the evolution. If used from within Amuse, the user script has more freedom to interact with the code, changing particle properties (stellar mass and radius, the former determines the balance between dynamical friction and heating, and the latter controls the distance of a tidal disruption by the central supermassive black hole). In the latter case, the code may also be used as a general-purpose interface providing gravitational potential and forces to be used in the `bridge` approach.

2 Obtaining and compiling the software

Raga, as part of Agama, is available for download at `https://github.com/GalacticDynamics-Oxford/Agama`. The recommended installation procedure is to run `python setup.py install [--user]` and follow the instructions: the script will attempt to determine automatically all relevant compilation options, but may ask user's permission to download required and optional third-party libraries. Alternatively, one may install the prerequisites manually, edit the file `Makefile.local` and run `make`. The following external libraries are used:

- GSL (C math library).
- optional (but recommended): Eigen library for numerical linear algebra.
- optional: UNSIO library – to enable support for Gadget and Nemo N-body snapshot formats; without it only the text format is available for input/output and Nemo for output.

3 Structure of the algorithm

The simulation progresses in so-called episodes; the duration of each episode can be much longer than the dynamical time, but shorter than the relaxation time. The episode length is constant for all particles. Particle trajectories are computed independently from each other (`openmp`-parallelized) during each episode, using a high-accuracy, adaptive-timestep 8th order Runge–Kutta integrator. The smooth gravitational potential in which all particles move is represented by a spherical-harmonic (multipole) expansion with the order and symmetries chosen by the user. The potential is constructed at the beginning of the simulation and (optionally) updated after each episode using trajectories of particles recording during the episode. During the orbit integration, one or more "tasks" can be attached to each particle, collecting data and/or changing the properties of the orbit. After all particles have been processed, each task is performing its own "finalization" step, possibly changing the global properties of the system, and the entire episode is repeated until the end of simulation time.

The available tasks, and the conditions for them to be used, are described below, in the same order as they are invoked during the simulation (the order matters!).

- *Loss cone treatment*: invoked when there is a single or a binary SMBH at origin with a non-zero capture or tidal disruption radius. The radius of tidal disruption is $r_\mathrm{t} \equiv (M_\bullet/m_\star)^{1/3}\, r_\star$, where $m_\star$ and $r_\star$ are the stellar mass and radius of each particle, and $M_\bullet$ is the mass of the central SMBH (or each of the two components of the binary SMBH). The radius of direct capture is $r_\mathrm{c} \equiv 8GM_\bullet/c^2$, where G is the gravitational constant (equal to unity in N-body units used by the code), and c is the speed of light. The largest of these two radii is chosen as the loss-cone radius r_LC for each particle. To disable the loss cone, one should set stellar radii to zero and c to infinity. When the distance of closest approach of a particle to the SMBH is less than r_LC, the particle is eliminated from the subsequent simulation and its mass (or some fraction of it) is added to the SMBH at the end of episode. The list of captured particles and their properties at the moment of capture are stored in a text file.

- *Binary black hole evolution*: applies when there is a binary SMBH in the center. The orbit of the binary is assumed to be Keplerian, oriented in the $x-y$ plane along the x axis. Time-dependent potential causes the particles that approach the vicinity of the binary to change energy and z-component of angular momentum. These changes are recorded, and at the end of the episode the sum of these changes, weighted by particle masses, is used to adjust the orbital parameters of the binary (semimajor axis and eccentricity), using the conservation laws. When a finite speed of light is specified, these orbital parameters additionally change due to gravitational-wave emission.

- *Potential recomputation*: switched on by the corresponding flag in the INI file (**off** by default). Collect sampling points from each particle's orbit during the episode and use them to update the total potential at the end of the episode. In the simplest case, only the position at the end of episode is used, but more than one sampling point per particle is possible by setting the `numSamplesPerEpisode` parameter – this reduces the discreteness noise in the potential.

- *Relaxation*: turned on by a non-zero value of the `coulombLog` parameter. The Spitzer's Monte Carlo approach for simulation two-body relaxation consists of adding perturbations to particle velocities during orbit integration, after each internal timestep of the ODE solver. These perturbations are computed from the local (position-dependent) drift and diffusion coefficients, which in turn depend on the distribution function (DF) of scatterers. The latter is identified with the entire population of particles in the simulation, but approximated by a spherically-symmetric isotropic DF $f(E)$.
  An important point is the distinction between the *gravitational* mass of a particle, which determines its contribution to the total potential and the total DF, and the *stellar* mass, which determines the relaxation rate and the balance between the energy loss due to dynamical friction and energy gain due to heating. For instance, a simulated $10^7\,M_\odot$ cluster may be represented by 10^5 particles with gravitational masses $10^2\,M_\odot$ and stellar masses $0.5\,M_\odot$, each particle corresponding to 200 equal-mass stars. The relaxation rate will still correspond to that of a 2×10^7-star system. Moreover, stellar masses of particles need not be equal, and heavier stars experience stronger dynamical friction and gradually sink to the center of the system. Initially, stellar and gravitational masses of each particles may be assigned arbitrarily and independently from each other, but if the stellar mass is updated in the course of evolution, the gravitational mass is adjusted by the same amount, to keep the ratio between them (i.e., the number of stars represented by each particle) constant.
  When this task is activated, samples of particle energies are recorded at regular intervals during the episode and used to recompute the DF and the diffusion coefficients (in the same way as for the potential, possibly using more than one sample per particle).

- *Trajectory output*: if the output interval is assigned, store the particle positions, velocities and masses in an N-body snapshot file. This is done at most once per episode, at its end.

Input and output snapshot files may be provided in any of the supported formats (text format by default, but also Nemo and Gadget if available; however, the latter does not support all particle attributes). In text files, the first 6 columns contain position and velocity in Cartesian coordinates, the next column is the *gravitational* mass, and two more optional columns are the *stellar* mass (if not provided, assumed equal to the gravitational mass) and the stellar radius (if not provided, assumed zero; it is only needed for the loss-cone task). In Nemo files, these additional attributes are stored in `Aux` (stellar mass) and `Eps` (stellar radius) fields.

4 INI parameters

For the standalone program, all parameters are set in the INI file, which should be passed as the one and only command-line argument. The file should contain a section named `[Raga]` with the parameters described below (case-insensitive, default values are given in brackets), using N-body units ($G = 1$). For the Amuse interface, these parameters should be provided in the constructor of the `Agama` worker object; dimensional parameters should have the appropriate dimension.

- `type`, `scaleRadius`, `mass` and other parameters describing a particular built-in gravitational potential can be used to explicitly specify the initial potential of the system. If `type` is not provided (default), the initial Multipole potential is computed from the initial snapshot. When potential update is disabled, this initial potential will remain fixed throughout the evolution (even though the DF will still be updated after each episode if relaxation rate is nonzero).
  In an AMUSE script, it is possible to create a given potential (whether analytic or computed from particles) and use it simply as an external potential in a `bridge` scheme, without invoking any dynamical evolution provided by the RAGA code.

- `particles` – the initial snapshot in the AMUSE script (optional: particles can also be loaded from a file or assigned at a later stage). If provided, it should be an instance of `amuse.datamodel.Particles`, optionally with an additional attribute `gravitating_-mass` (the `mass` attribute in AMUSE corresponds to stellar mass).

- `fileInput` – the input N-body snapshot (required only for the standalone program). It may be in any of the formats supported by UNSIO library (e.g., NEMO or GADGET), or – even without this library – a simple text file with at least 7 columns, as described in the previous section. The central SMBH should not be included in the snapshot, and given by the `Mbh` parameter instead.

- `fileLog` (`fileInput.log`) – the name of a text file where the diagnostic information will be written.

- `timeTotal` – the total simulation time (required for the standalone program, optional for AMUSE, in which the user script prescribes the evolution time).

- `timeInit` (`0`) – initial time, i.e., an offset added to all internal timestamps (useful if continuing a previous simulation).

- `episodeLength` – duration of one episode; if none provided, this means that the entire simulation is performed in a single go. Typically it should be considerably shorter than the timescale on which the system evolves (the central two-body relaxation time, the binary black hole hardening timescale, or the stellar evolution timescale if run from within AMUSE), but may well be longer than the characteristic dynamical time. In an AMUSE script, one may manually advance the evolution by any length of time, so this parameter is not necessary.

- `symmetry` (`triaxial`) – the type of potential symmetry that determines the choice of non-trivial coefficients in the Multipole expansion. Possible values: `spherical`, `axisymmetric`, `triaxial`, `reflection`, `none`, or a numerical code (see `coords.h`); only the first letter is important.

- `lmax` (`0`) – the order of angular expansion (should be an even value, 0 implies spherical symmetry).

- `GridSizeR` (`25`) – size of the radial grid in the Multipole potential (rarely needs to be adjusted)

- `rmin, rmax` (`0`) – min/max grid radii; if not provided (0), they are assigned automatically based on the particle distribution.
- `Mbh` (`0`) – mass of a central black hole, or the combined mass of a binary black hole. Its center of mass is fixed at origin, and it adds a Newtonian contribution to the stellar potential. Warning: the input snapshot should not contain the black hole particle.
- `binary_sma` (`0`) – semimajor axis of a binary black hole (0 means no binary, while a positive value turns on the task of evolving the binary orbit parameters).
- `binary_q` (`0`) – mass ratio of a binary black hole (components have masses $M_{\rm bh}/(1+q)$ and $M_{\tt bh}\,q/(1+q)$, 0 means no binary).
- `binary_ecc` (`0`) – eccentricity of a binary black hole; its orbit is assumed to lie in $x-y$ plane oriented along x axis.
- `updatePotential` (`false`) – whether the stellar potential is recomputed after each episode.
- `coulombLog` (`0`) – the value of Coulomb logarithm $\ln\Lambda$, which sets the amplitude of velocity perturbations that mimic the effect of two-body relaxation. As explained above, the relaxation rate is determined by *stellar* masses assigned to particles, not their *gravitational* masses (the latter determine the density profile of the system). A typical value of $\ln\Lambda \simeq \ln N_\star$ or $\ln M_\bullet/m_\star$ is 10–15, and setting it to zero turns off the two-body relaxation. One should keep in mind that even with the relaxation rate set to zero, the recomputation of potential from particles leads to unavoidable discreteness noise, which is, however, much lower than the level of numerical relaxation in conventional N-body simulations: both the long interval between updates (episode length) and the use of more than one sample per particle greatly suppress this noise.
- `numSamplesPerEpisode` (`1`) – number of sample points taken from the orbit of each particle during one episode and used in recomputation of the potential and the distribution function; a value > 1 reduces the discreteness noise (a few dozen is a reasonable value).
- `gridSizeDF` (`25`) – size of the grid in energy space used for representing the distribution function.
- `captureMassFraction` (`1`) – fraction of mass of captured particles that is added to the mass of the black hole.
- `speedOfLight` (∞) – specifies the speed of light in N-body velocity units. It is responsible for the loss of energy and angular momentum of a binary SMBH due to gravitational-wave emission, and also determines the lower limit on the capture radius of a single SMBH (4 times the Schwarzschild radius).
- `outputInterval` (`0`) – interval between writing out N-body snapshots, potential expansion coefficients, and relaxation model (should be an integer multiple of episode length, 0 disables this).

- `fileOutput` – file for writing N-body snapshots (if the format is Nemo, all snapshots are stored in a single file, otherwise each one goes into a separate file with the timestamp appended to the file name).
- `fileOutputFormat` (`text`) – output format for snapshots (text/Nemo/Gadget, the latter is available only with the UNSIO library; only the first letter matters).
- `fileOutputPotential` – base filename for writing the coefficients of Multipole expansion (timestamp appended to the name).
- `fileOutputRelaxation` – base filename for writing the table with parameters of the spherical model used to compute diffusion coefficients (timestamp appended to the name).
- `fileOutputLosscone` – file for storing the list and parameters of particles captured by the black hole(s).
- `fileOutputBinary` – file for storing the orbital parameters of the binary black hole (semimajor axis and eccentricity) after each episode.
- `accuracy` (10^{-8}) – accuracy parameter of the orbit integrator; roughly corresponds to the relative energy error per orbital period. The default value is good enough for most cases, but keep in mind that very tightly bound particles close to the SMBH may complete millions of orbits per episode!
- `maxNumSteps` (10^{8}) – upper limit on the number of time integration steps per episode for each orbit; there are typically a few dozen or more steps per orbital period.
- `number_of_workers` (Amuse only) – the number of OpenMP threads used by the simulation (*not* the number of independent MPI processes, unlike some other codes). If not provided, Raga *will use all available cores on the machine!* For the standalone program, a similar effect is achieved by setting the environment variable `OMP_NUM_THREADS`.

References

[1] Vasiliev E., 2015, MNRAS, 446, 3150

[2] Vasiliev E., 2019, MNRAS, 482, 1525

[3] Portegies Zwart, S., McMillan, S., van Elteren, E., Pelupessy, I., de Vries, N. 2013, Comp.Phys.Comm., 184, 456