SMILE reference

Eugene Vasiliev
Lebedev Physical Institute, Moscow, Russia
Rochester Instutute of Technology, Rochester, NY, USA
email: eugvas@lpi.ru

Version $2.5\beta_1$
February 1, 2015

Contents

1 Introduction

SMILE[1] is a software for orbital analysis and Schwarzschild modelling. The scientific reference papers are [1] and [2]; here comes a more technical and practical guide.

As the name suggests, the program is intended to study orbits and self-consistent Schwarzschild models in various potentials, and might also have educational purposes. The primary applications are:

- Exploring orbits in various potentials, either given by analytical formulae or several approximate expansions;
- Studying properties of orbits, from builtin orbit integrator or from external data;
- Constructing equilibrium models of triaxial stellar systems with given density profile by Schwarzschild method.

As of version 2.5, it is not intended for modelling observational data of any kind; however, plans exist to develop an observationally-driven Schwarzschild code, with full account of observational errors and likelihood model search.

The program comes in two versions – GUI interactive tool (Sec. 2) and console program with scripting support (Sec. 3). The GUI version is more suited to "exploratory" and "educational" purposes, since it has many interactive connections between different modules allowing to easily visualize the results. The console one is more appropriate for remote and batch computations, when you know what you're doing.

There are numerous adjustable parameters which are kept in INI file (Sec. 4.1); most of them may be changed in the GUI, and described in the appropriate section of GUI reference; some are not modifiable from GUI and these will be described in the section about INI file.

The architecture of the software is designed to be as flexible and general as possible. Some modules and blocks can be used in external programs (e.g. orbit integration and analysis, generation of equilibrium spherical models, computation of potential and forces), and overall philosophy is to create a layered, modular and extensible design. In particular, the potential module with some support routines forms a separate library, LIBSMILEPOT (Sec. A.1), which has C/C++ and `Python` interfaces and bindings to NEMO, GALPY and AMUSE packages.

The source code and compiled versions for various platforms may be downloaded from `http://td.lpi.ru/~eugvas/smile/`. If you need to compile it from source, refer to Appendix A.

2 GUI – interactive environment

The window is split into three areas – right panel contains the parameters of potential and orbit integration, left side contains one of several tabs depending on the current module (analysis of a single orbit, orbit library or Schwarzschild model); in the top are

[1] Schwarzschild Modelling Interactive expLoratory Environment

the parameters for current module, the rest is occupied by plot area (again depending on the selected task).

2.1 Right panel

Potential: Several potential/density pairs are implemented, having different subsets of parameters.

- Logarithmic: $\Phi(\tilde{r}) = \frac{\sigma^2}{2} \ln(R_0^2 + \tilde{r}^2)$;
- Anisotropic harmonic oscillator: $\Phi(\tilde{r}) = \frac{M}{2}\tilde{r}^2$;
- Dehnen: $\rho(\tilde{r}) = \frac{(3-\gamma)M}{4\pi pq R_0^3} (\tilde{r}/R_0)^{-\gamma}(1+\tilde{r}/R_0)^{-(4-\gamma)}$;
- Miyamoto–Nagai: $\Phi(R,z) = -M \Big/ \sqrt{R^2 + (R_0 + \sqrt{z^2+R_2^2})^2}$
- Ferrers: $\rho(\tilde{r}) = \frac{105\,M}{32\pi pq R_0^3} (1-(\tilde{r}/R_0)^2)^2$;
- Scale-free: $\rho(\tilde{r}) = M\tilde{r}^{-\gamma}$;
- Basis-set expansion (BSE or BSECompact, see below);
- Spline expansion (see below);
- Cylindrical spline expansion (see below);
- Spherical (see below);
- Frozen-N-body (see below).

Here $\tilde{r} \equiv (x^2+y^2/q^2+z^2/p^2)^{1/2}$ is the elliptical radius, and $R \equiv \sqrt{x^2+y^2}$ is the cylindrical radius. The potential parameters are:

- $q = y/x$ and $p = z/x$ – axis ratios, should be $p \le q \le 1$. Define axis ratio for potential (in case of logarithmic and harmonic) or density (in other cases).
- M – total mass of the density model (in case of logarithmic potential the squared asymptotic rotation velocity σ^2 is provided instead of M, and in case of harmonic and scale-free potentials it is just an arbitrary normalization parameter).
- $M_{\rm bh}$ – mass of central black hole (point mass).
- γ (cusp exponent) – index of power-law density profile in the scale-free model or in the inner region of Dehnen model, should be $0 \le \gamma \le 2$.
- R_0 (scale radius) – in log.potential denotes region of constant-density core, may be zero; in other density models is the scale length.
- R_2 (second scale radius) – used in Miyamoto–Nagai and NFW models.

- $N_{\rm radial}$, $N_{\rm angular}$ and $N_{\rm vertical}$ – number of terms in basis-set and spline expansions (the last one is used for Cylindrical spline only).

- ϵ (softening length), θ (tree opening angle) – parameters for frozen-N-body tree-code

- Nbody file – the meaning of this file depends on what is selected in the "Density" drop-down list: an N-body snapshot for initializing the BSE/Spline/frozen-N-body potential, or a text file describing an Ellipsoidal or MGE mass model (Sec. 4.4.2), or a text file with the coefficients of BSE/Spline expansion (Sec. 4.4.4).

BSE (basis-set expansion, [3]), Spline and Cylindrical spline are general-purpose potential expansions which may be used to approximate almost any potential model. The first two (actually, three, since BSE comes with two choices for the basis set – infinite and compact) are based on spherical-harmonic expansion and suitable for density profiles that are not too much flattened; the third is efficient even for very flattened models but is more computationally expensive. The coefficients of expansion are calculated either from an analytic density profile, from a set of N point masses, or from a smooth density profile given by an Ellipsoidal mass model or a Multi-Gaussian expansion. The list of available density profiles includes:

- All finite-mass potential models defined above (Dehnen, Miyamoto–Nagai, Ferrers);

- Plummer: $\rho(\tilde{r}) = \frac{3\,M}{4\pi pqR_0^3}[1+(\tilde{r}/R_0)^2]^{-5/2}$;

- Perfect ellipsoid: $\rho(\tilde{r}) = \frac{M}{\pi^2 pqR_0^3}[1+(\tilde{r}/R_0)^2]^{-2}$;

- Isochrone: $\rho(\tilde{r}) = \frac{M}{4\pi pq}\frac{3(R_0+a)a^2-(R_0+3a)\tilde{r}^2}{a^3(R_0+a)^3}$, $a \equiv \sqrt{R_0^2+\tilde{r}^2}$;

- modified Navarro–Frenk–White (NFW) with an outer cutoff: since the original NFW profile has logarithmically diverging mass, it cannot be used in potential expansion directly; instead, a modification with steeper outer profile is introduced. $\rho(\tilde{r}) = C\,(r/R_0)^{-1}\,(1+r/R_0)^{-2}\,(1+r/r_{\rm cut})^{-1}$ with $r_{\rm cut}$ being computed so that the total mass is equal to the mass of a NFW model with concentration c (sharply cut beyond radius c). $r_{\rm cut}$ is somewhat (but not much) smaller than c because the cutoff is smoother, which is crucial for an efficient potential expansion. The concentration is defined as the ratio between virial radius and core radii (R_2/R_0).

- Sérsic: the deprojected density profile is given by
$\rho(\tilde{r}) = \dfrac{M}{pqR_0^3}\dfrac{b_n^{2n+1}}{2\pi^2 n^2\Gamma(2n)}\displaystyle\int_0^\infty \exp\left[-b_n\left(\frac{\tilde{r}}{R_0}\cosh\eta\right)^{1/n}\right]\left(\frac{\tilde{r}}{R_0}\cosh\eta\right)^{1/n-1} d\eta\,,$
where n is the Sérsic index and $b_n \approx 2n-1/3$ is the root of $\Gamma(2n) = 2\gamma(2n,b)$.

- Exponential disk: $\rho(R,z) = \frac{M}{2\pi R_0^2}\exp(-R/R_0)\,\zeta(z)$, with the vertical profile being either $\zeta(z) = \frac{\exp(-|z|/R_2)}{2R_2}$ or $\zeta(z) = \frac{\mathrm{sech}^2(z/R_2)}{2R_2}$.

- Ellipsoidal mass model is a flexible way to represent arbitrary density profile with arbitrary variation of axis ratios with radius (Sec. 4.4.2). It allows to provide a smooth density model described by user-supplied mass profile from a text file, combining advantages of a smooth density (to avoid statistical fluctuations in expansion coefficients inherent for an N-body initialization of potential expansions) and flexibility in potential description. The disadvantage is a longer initialization time (however, the orbit integration time depends only on the number of expansion coefficients, not the way they were computed; the coefficients are also stored in a text file so the subsequent runs may re-use it).

- Multi-Gaussian expansion (MGE) is another widely-used parametrization of an arbitrary density profile by a sum of gaussian components. The model is defined by a text file, as explained in Sec. 4.4.3.

In both BSE and Spline expansions, the angular dependence of potential and density is represented in spherical harmonics with terms up to $l_{\max} \equiv N_{\text{angular}}$, while the radial dependence is either a sum of small number $N_{\text{radial}} + 1$ of basis functions (in BSE) or a spline interpolation on a grid of N_{radial} points in radius (in Spline). For BSE, there are two sets of basis functions: one, suitable for infinite-extent density profiles such as Dehnen or NFW, is the Zhao(1996) basis set [4], with the parameter α controlling the shape of radial basis functions (0 means auto-detect, values between 1 and 2 are reasonable in most cases), the other (BSECompact) is suitable for finite-extent profiles and is based on spherical Bessel functions [5], with $R_{\max}$ being the radius of compact support of the basis. For the Cylindrical spline expansion, only the azimuthal ($\sin/\cos m\phi$) part of density/potential is expanded in Fourier series, with the other two coordinates in cylindrical basis being represented by a two-dimensional interpolating spline (Sec. B.3.6). This potential is defined in a finite domain ($R < R_{\max}, |z| < z_{\max}$) and extrapolated outside this domain using quadrupole expansion. Finally, there is a Spherical potential which, as hinted by its name, can represent an arbitrary spherically symmetric system given by any of the available density profiles (including generic profiles specified by Ellipsoidal and MGE parametrizations, or N-body snapshots). Its function is pretty much the same as the Spline potential with $l_{\max} = 0$, with the extrapolation to large/small radii performed slightly differently; it simply encapsulates the machinery of spherical mass models (Sec. B.2) into the potential module.

Depending on assumed type of symmetry, only some terms in angular expansion may be used:

- None: all terms with $l \le l_{\max}, -l \le m \le l$ are used;

- Reflection: terms with odd l are zero;

- Triaxial: no terms with odd m or $\sin(m\phi)$ (implementation note: instead of $e^{im\phi}$ we use $\cos(m\phi)$ for $m \ge 0$ and $\sin(|m|\phi)$ for $m < 0$, so this type of symmetry implies that terms with $m < 0$ or $m = 1 \pmod 2$ are zero);

- Axisymmetric: use $m = 0$ only;

- Spherical: $l_{\max} = 0$;

The symmetry type is adjustable when initializing the expansion fron an N-body file, otherwise it is assumed to be triaxial or higher, depending on axis ratios. If initializing from a set of point masses or from an ellipsoidal/MGE model, the potential coefficients are written to a `filename.coef_bse/coef_bsec/coef_spl/coef_cyl` file (Sec. 4.4.4), which may be then used instead of the original N-body/Ellipsoidal/MGE file as an input to BSE/Spline/Cylindrical spline initialization.

Frozen-N-body is a representation of potential of N particles fixed in space; Barnes&Hut tree-code is used for its computation, with tree opening angle θ (the less its value, the more accurate is tree approximation and the longer computation time) and softening length ϵ (may be even set to zero, since the integration uses adaptive timestep based on acceleration, but it is not recommended as it runs terribly slow and is not optimal in terms of bias/variance tradeoff. A better choice is spatially adaptive softening based on local density, which is selected by assigning a negative value to ϵ, so that the actual softening length is $|\epsilon|$ times local mean inter-particle distance).

In the case of generic potential expansions and N-body potential, the file with particle coordinates should be supplied (by pressing the button **Nbody file** and selecting file, or typing the filename and pressing Enter). The file should be in one of the known N-body snapshot formats (Sec. 4.5). As the potential initialization may take a long time, it is only done upon pressing the button "Init potential" or selecting the N-body file, rather than on every change of parameters as for other potentials.

SMILE supports multi-component potential/density models: for example, to create a model of a stellar disk embedded in a spherical halo one provides two potential models. The combobox at the top of right panel determines the choice of the potential component whose parameters are displayed in the panel; one may add or delete components using the same combobox. Depending on the type of potentials, one may need to press "Init potential" after changing any of the parameters and the number of components. If you see this button, the new settings are not yet applied.

Orbit integration: Defines several parameters related to the computation of orbits.
First is the choice of **orbit integrator**: for the N-body potential it is Leapfrog, and for others it can be the default 8th order Runge–Kutta (DOP853) method [6], the 15th order IAS15 method [7], the 4th order Hermite integrator, or – if the program is compiled with the support of ODEINT library [8] – several other Runge–Kutta and the Bulirsch–Stoer methods.
Relative and **absolute error tolerance** are the accuracy parameters for the integrator; for IAS15 there is only one accuracy parameter (and its numerical value is much higher); and for fixed-timestep integrators the parameter is the step size in units of $T_{\rm orb}$.
Relaxation rate regulates the amplitude of velocity perturbations that are added during orbit integration and mimic the effect of two-body relaxation. The numerical value of this coefficient corresponds to $N^{-1} \log \Lambda$, where N is the number of stars in the stellar system to be modelled, and $\log \Lambda \approx \log N$. Zero turns off these perturbations. The relaxation coefficients are computed from a spherically-symmetric approximation to the true density profile under the assumption of isotropic velocity distribution; thus they will not be ade-

quate for a dynamically cold system such as a thin disk.
Lyapunov exponent: If turned on, one may look at the behaviour of deviation vector and finite-time Lyapunov exponent on the "Lyapunov" tab, and use its value in distinguishing regular and chaotic orbits. There are two methods of computing the Lyapunov exponent: integration of a nearby orbit (not always accurate enough; slows down computation approximately twice) and integration of a variational equation (a bit more accurate but also slower, and not available for some potentials). Not applicable for frozen-N-body potential (would give positive values anyway).
Integration time is given in units of $T_{\rm orb}$.
Sampling interval is the time interval between storing points from the trajectory, in units of $T_{\rm orb}$. It affect basically only orbit rendering, but if set too coarse, it may hinder to find higher-frequency spectral lines, so keep it at least ≤ 0.1.
Omega rotation is the angular frequency of figure rotation of the potential.
Dimensions: 2d or 3d – switch regimes; 2d is basically for studying motion in principal planes and for Poincaré section, 3d is for real world.

Initial conditions: May specify either 3 coordinates and 3 velocities, or only energy/Jacobi constant (switch radiobuttons at left). The latter case is primarily used in construction of frequency map, while the former is for studying individual orbits. $T_{\rm orb}$ is the period of x-axis orbit, which is calculated automatically and used as unit of time in integration time and frequency analysis.

Remember that for scale-free and harmonic potential $E > 0$, for log it may be arbitrary, and for other potential models $E < 0$ because they have finite mass.

Start buttons: **Start** just does what is does, starts an orbit with given initial conditions (also invoked by pressing Enter in most input lines); **Random** sets arbitrary IC with the same energy and starts orbit integration. This is only available if the energy is within the range of bound orbits; or, in the case of figure rotation, if the Jacobi constant is below the saddle point (L1/L2).
The integration is performed in separate thread, so one may move around GUI during computation.

Part of orbit: Show i'th part out of N – if $N > 1$, split orbit into N equal intervals, and performs frequency analysis and rendering only for given interval.

Save settings on exit: if checked, saves INI file with most of settings, which are automatically retrieved upon launch.

Print: prints current figure in the left panel to a PS or PDF file.

Results text box: this message area contains the results of last operation. For orbit integration – results of orbit classification: leading frequencies, orbit class, minimum distance to center, frequency diffusion rate, Lyapunov exponent (if checked), error in

conservation of energy or Jacobi constant (should tend to 0), wall-clock time for computation. For frequency analysis and Schwarzschild modelling – orbit population, solution of optimization problem, etc.

2.2 Left panel – tabs

2.2.1 Orbit

Orbit plot type: 2d projection of an orbit onto one of principal planes (selected by **2d plane**). In this and other 2d plots in the program the following commands are available: left mousebutton – zoom, Escape or right mousebutton – unzoom to default view, middle mousebutton – pan/shift.
3d line rendering of orbit: left mousebutton – rotation, Ctrl+left – move, mousewheel – zoom;
3d mesh rendering of orbit as a solid body (using Delaunay tesselation performed by an external program `qdelaunay`, in a separate thread – so it is available after some delay upon finishing of integration).
$\boldsymbol{r}_{\rm peri}$**,** $\boldsymbol{L}^2_{\rm peri}$ **and** $\boldsymbol{L}^2$ displays the cumulative distribution of various quantities: pericenter radius, squared angular momentum at pericenter, and squared angular momentum along the entire orbit (sampled at each step). The array of recorded values is sorted and displayed as y values on the plot, while x runs from 0 to 1.
$\mathbf{\Delta E(t)}$ **and** $\mathbf{L(t)}$ show time dependence of energy and angular momentum. These plots are interesting to see in the case when the effect of two-body relaxation was switched on.

3d mesh parameters: Here one may choose to display entire orbit or just a half of it lying above one of principal planes, and also specify the maximal segment length of facets (if it was unlimited, any orbit would look like a convex blob; setting it too large will remove details, too small – create holes; it is automatically selected based on typical segment length of trajectory). To apply changes, press **Refresh** ([re]starts the tesselation thread).

Load/save orbit to a text file (Sec. 4.4.5).

2.2.2 Potential

This panel is used to display the radial profile of the **potential**, **density**, **circular velocity** (rotation curve), **frequencies**, or **surface density**, depending on the choice of the radiobutton. The circular velocity is defined as $v_{\rm circ} \equiv \sqrt{x_i\,\partial\Phi/dx_i}$ for i-th coordinate, which matches the conventional definition for axisymmetric systems if i is x or y. The surface density option displays $\Sigma(x,0) \equiv \rho(x,0,z)\,dz$, analogous quantity for y, or $\mathcal{M}(z) \equiv \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy\,\rho(x,y,z)$ for z coordinates in the abscissa axis. The frequencies option shows the three frequencies: circular (blue), epicyclic (green), and vertical (red). One may use **linear** or **logarithmic** scaling, and choose between three principal **axes** to be used as the abscissa. The radial extent of the plot is determined by the current initial conditions (the radius of x-axis orbit with the same energy as the current orbit; in case

of logarithmic scaling the abscissa covers three decades in radius inward from the outer extent). If the potential consists of more than one **component**, each of them may be displayed on the plot separately or all components together (in particular, a central black hole is included in the potential if all components are selected, but not for individual ones even if there is only one). The potential/density profile can be **exported** to a text file (Sec. 4.4.7).

2.2.3 Poincaré

Poincaré section is an useful tool for studying orbital structure of 2d systems. (It may be used in 3d, but is mostly meaningless). Needs to be turned on by corresponding checkbox. The coordinates used in the surface of section are specified by two drop-down lists: when the first coordinate passes through zero with positive derivative, then the second coordinate and its derivative are added to the surface.

Each orbit integration adds a series of points with a new color to the plot. Regular orbits have these points grouping in one-dimensional cycles; chaotic ones have scattered set of points in two-dimensional regions. Red outer curve marks the equipotential surface.

Plot may be zoomed in by left mousebutton, Ctrl-right zooms out, middle button moves. Right click within the equipotential boundary sets up the initial condition (to integrate the orbit, press Enter thereafter). There is a button to locate the periodic orbit that parents the current one and closes after a specified number of passages through the plane (works intermittently). Changing the energy clears the plot (as does the eponimous button). One may also export the contents of the plot to a text file.

2.2.4 Frequencies

Displays spectra of orbit in three coordinates (blue – x, green – y, red – z), frequencies measured in units of inverse X-axis orbit period. Vertical lines show detected spectral lines (white – by the precise Hunter method, black – by the non-refined Carpintero&Aguilar method which is accurate to within Nyquist frequency; the latter is used when Hunter method produces divergent results, typically for very closely spaced lines). Lines may be turned off by checkbox. Left mousebutton zooms, middle button moves, right click zooms out.

2.2.5 Lyapunov

Displays quantities used to estimate Lyapunov exponent of an orbit, in the case that it is calculated (then the evolution of deviation vector $\mathbf{w}$ is computed along with the orbit integration)

X axis is for time (in $T_{\rm orb}$ time units); left Y axis is for finite-time estimate of Lyapunov exponent $\Lambda = T_{\rm orb}\ \ln(|\mathbf{w}|)/t$ (relative, i.e. normalized to unit frequency), in blue; right Y axis is for deviation vector divided by time, $|\mathbf{w}|/t$, in red; both axes are logarithmic.

For regular (part of) orbit Lyapunov exponent decreases as t^{-1}, and deviation vector grows linearly, so the red line is horizontal. When chaos starts to appear, Λ fluctuates around non-zero value, and $\mathbf{w}$ grows up. (See Fig. 4 in [1] for explanation).

2.2.6 Frequency map

Here one may study an ensemble of orbits by means of frequency map (and other tools).

Frequency map is typically built for given energy (specified in the right panel); however, one may also view orbit library from Schwarzschild model, in this case **View shell** spinbox selects the energy level from Schwarzschild grid (0 means display all orbits).

To create FM, one specifies the **number of points** in the start-spaces:
stationary (initial conditions on the equipotential surface with zero velocity),
principal-plane (on three principal planes),
$Y - \alpha$ (on y axis, with velocity perpendicular to it, as in Schwarzschild 1982),
random (yeah, anything you like! Ergodic within the energy hypersurface).
Note that the first three options are sort of meaningless in the case of figure rotation (the orbits will not have the same value of Jacobi constant); the last option will only work if the value of Jacobi constant is below the Lagrangian point L1.
Alternatively, one may use an existing start space loaded from a file. In this case, setting 0 as the integration time (in the right panel) forces to use the values for each orbit written in the orbits file (otherwise they are overridden with the settings in GUI).

Start button starts the orbit integration in several parallel threads (their number being based on the number of processor cores). Once started, this button serves to terminate prematurely the threads (after each one finishes its current orbit).

Import/Export loads/stores data in the Orbit Library format (Sec. 4.2). The current configuration is kept along with the orbits file in the corresponding `.ini` file and automatically loaded during import.

The main area displays plots based on the selected radiobutton in the top-right array:

- **Frequency map:** each orbit is represented by point which coordinates are ω_y/ω_x and ω_z/ω_x (for 3d) or simply ω_x and ω_x (for 2d), where the ωs are the leading frequencies in each coordinate. 3d map also is decorated with a dozen of most important lines representing resonant or thin orbits (most notably, $(0, 1, -1)$ line for long-axis tubes (LAT) and $(1, -1, 0)$ for short-axis tubes, SAT).

- **Histogram:** cumulative distribution function of either of two chaos indicators.

- **Start-space** (stationary, principal-plane, and $Y - \alpha$) – points from the corresponding start space are plotted in 2d projection.

The points in the plot are colored in blue (regular) or red (chaotic), based on the criteria in the **chaos criterion** section: an orbit is termed to be chaotic if it has either the frequency diffusion rate $\delta\omega$ larger than the threshold given, or if its Lyapunov exponent Λ is larger than the threshold (usually 0, since all orbits with positive exponents are chaotic; if it was not computed, this has no effect).

The coloring depends on only one of these two criteria (select appropriate radiobutton), but the labelling of an orbit as chaotic happens if either of them is satisfied. This labelling is important in Schwarzschild modelling, and in calculation of fraction of chaotic orbits.

View shell setting enables to filter out only orbits whose initial conditions place them in a particular energy shell of the Schwarzschild model (0 shows all orbits).

Only nonzero weight checkbox filters out only the orbits with nonzero weights in the Schwarzschild model (and for the cumulative distribution histograms, the weight of each orbit is also taken from the model). If in addition **view shell** is set nonzero, only the orbits from the corresponding energy shell are shown.

In the frequency map and spart-space chart one may right-click on the plot and select the nearest orbit, which set the initial conditions and displays some information about the orbit. Pressing Enter one may then re-integrate the orbit. Additionally, zoom, move and unzoom on frequency map is the same as in Poincaré plot.

2.2.7 Schwarzschild model

Orbit library tab specifies the number of orbits in the entire model and the method of **initial conditions generation** (Sec. B.8). All of these methods can use uneven sampling in energy space, if **bias factor in energy** is non-zero, then it specifies the relative difference in weight between the most bound and the least bound orbits (i.e. if this parameter equals 9, then the ratio will be 10, thus increasing the number of orbits in the central parts of the model). The choice of **Eddington** IC generator has additionally a **bias factor in angular momentum**, which may be used in a similar way to increase the number of low angular momentum orbits (and correspondingly decrease their average weight). **Spherical Jeans** IC generator takes additional **velocity anisotropy** parameter (commonly denoted as β); and **Axisymmetric Jeans** has two parameters – **meridional velocity anisotropy** (β_m) and **degree of rotation** (k). Finally, one may use already loaded orbit library as initial conditions.

Number of sampling points for Nbody model specifies how many points drawn randomly from each orbit trajectory will be stored in `.smpl` file for subsequent creation of N-body initial conditions. If you do not plan to create N-body model, may set it to zero.

Import/export *SM* buttons does the same as for Frequency map, but in addition `.schw` and `.smpl` binary files are stored (Sec. 4.3). **Start** button initializes *SM*, creates initial conditions (if IC generator is not set to **use existing points**) and begins orbit integration in parallel threads.

Create N-body snapshot button generates a N-body representation of the Schwarzschild model, where particles are drawn from sampled points, their number proportional to orbit weight. If there are insufficient points for a particular orbit (that is, if its weight w is greater than $N_{\text{sampling points}}/N_{\text{bodies}}$), this orbit is set to be reintegrated for the same time interval, but collecting more sampling points. The reintegration is supposed to recreate the same orbit, but sometimes it may turn out to be different (e.g. if using Lyapunov exponent calculation, with initial deviation vector generated randomly), so it's best to avoid such situation. On average, one needs to have at least $10\,N_{\text{bodies}}/N_{\text{orbits}}$ sampling points per orbit.

There is an option to create N-body model with unequal mass particles ("mass refinement"), so that the innermost particles are lighter. If the refinement factor $Rf > 0$, the orbits are sorted in energy and binned into $Rf + 1$ bins, yielding approximately the same number (not mass) of particles each. Particle masses in each bin differ by a factor of two. NB: for rotating models the energy of initial conditions in the inertial frame is taken as

the sorting value, not the Jacobi constant which is actually the conserved quantity, but is not monotonic with radius.

The N-body model is exported in one of the supported snapshot formats (Sec. 4.5). Then some statistical quantities are calculated, including virial ratio and average position and angular momentum (for the entire model, and for particles within half-mass radius shown in brackets).

Model parameters tab contains the choice of *SM* variant, which should be made prior to starting orbit library integration itself:

- **Classic** Schwarzschild model: partitioning configuration space into a number of cells, compute fraction of time each orbit spends in each cell. Parameters: **Number of radial coefs** (shells), number of **lines** splitting each of three segments of each shell (so the number of cells in a shell is $3n_{\rm lines}^2$, and total number of constraints is $3\,n_{\rm lines}^2\,n_{\rm radial}$).

- **Cylindrical**: similar to Classic, but with the spatial grid covering a cylindrical region with base in $x-y$ plane and extending in z direction up to some $z_{\rm max}$. The grid consists of $n_{\rm radial}$ concentric shells in cylindrical radius, $n_{\rm angular}$ divisions in the azimuthal (ϕ) direction, and $n_{\rm vertical}$ slices in z direction.

- **SHGrid**: evaluating coefficients of spherical-harmonic expansion of the potential at a number of radial grid points (conceptually similar to Spline expansion, except that no splines are constructed, just the values at grid nodes are used). Parameters: number of radial shells, number of **angular coefs** ($= l_{\rm max}$, so that the number of terms in each shell is $(l_{\rm max}/2+1)(l_{\rm max}/2+2)/2$ since only terms with even l are used).

- **SHBSE**: evaluating coefficients of spherical-harmonic BSE expansion of every orbit. Parameters: number of angular coefs (same as above), radial coefs (equivalent to $n_{\rm radial}$ in BSE), α parameter of BSE (may set 0 for auto-detect). Total number of constraints is $(n_{\rm radial}+1)\,(l_{\rm max}/2+1)(l_{\rm max}/2+2)/2$.

Note that the *SM* variant chosen here doesn't need to be related to the potential used in orbit integration. However, if they are related (SHGrid *SM* and Spline potential, or BSE *SM* and BSE potential with the same value of α), this is used to speed up initialization of model constraints (often very substantially).

Regardless of the selected variant, there is still a radial grid used for recording kinematic data (that is, radial and tangential velocity dispersion of each orbit in each radial shell), which may be used in optimization to constrain velocity anisotropy. The size of this grid is given by the parameter $n_{\rm shells}$ and is not related to the number of radial coefficients in the density model.

Inner and outer shell mass specify which fraction of total model mass (which should be finite) is contained in the innermost and outermose radial grid node; 0 makes the default choice of $M_{\rm outer}=1-1/(N_{\rm shell}+1)$, $M_{\rm inner}=M_{\rm outer}/N_{\rm shell}$. If the requested inner shell mass is smaller than $1/N_{\rm shell}$ of the total mass of *SM* (which is $M_{\rm outer}$ times the total mass of this density model), then a non-uniform, exponentially spaced grid in shell

masses is constructed. This parameter applies to the kinematic grid and Classic density model.
The total weight of all orbits is required to match the total model mass, not the mass within the outermost grid shell. Initial conditions for particles cover the energy range from the lowest possible energy (unbounded for models with a black hole) to the binding energy at a radius containing 0.999 of total mass; thus some orbits are deliberately assigned to lie (mostly) beyond the grid, thus ensuring that the total model mass could be more than $M_{\rm outer}$.

For a multi-component *SM* (which necessarily involves a composite potential), **density components** enumerates the components of the potential whose combined density is used as the target of the model (empty line means all).

Note that all variants of *SM* imply at least triaxial symmetry of density profile, even if the underlying potential is less symmetric (no warning is given).

Optimization tab contains several parameters controlling the solution of optimization problem (see Sec. B.6 for the formulation of this problem).

Constraint penalty linear/quadratic define the contribution to the objective function penalizing the constraint violation. If both are zero, the constraints must be satisfied exactly (not recommended). If the linear penalty is positive, it corresponds to the factor α_1 in the penalty function (64); if negative, its absolute value gives the relative tolerance range for the constraints α_0 (65). The quadratic penalty α_2 may be zero or positive. *Note* that if the linear penalty is positive (default situation), then the optimization problem is always feasible, even if the constraint values are severely mismatched; therefore, a message "Optimal solution found" from the solver doesn't mean that the actual *SM* is feasible.

Regularization is the factor λ in the quadratic part of the penalty function (66), which drives the orbits towards a more uniform weight distribution.

Chaotic penalty is the factor μ in (67) which enhances (if > 0) or reduces (if < 0) contribution from regular orbits in the solution; its magnitude is in principle unlimited, but if set too high, the cost of adding an unwanted orbit will overweight the cost of constraint violation. Keeping its absolute value ≤ 1 is typically enough. Setting it to zero produces a model without any preferences about orbit properties.

Maximal orbit weight may be used to limit the weight assigned to each orbit by the solver (setting it too low will, of course, result in infeasibility of solution, so it only makes sense to adjust it in the case that a few clear outliers are seen, particularly in linear optimization). 0 turns off this constraint.

Constrain anisotropy option adds constraints to optimization problem, forcing the average velocity anisotropy coefficient β in each shell to equal a predefined value. Its value is linearly interpolated with the enclosed mass from $\beta_{\rm in}$ in the first shell to $\beta_{\rm out}$ in the last shell.

The optimization problem may be solved either by **linear** or **quadratic** programming using the interior-point method. There are several solvers available: GLPK library (LP only), Python-based CVXOPT module (LP/QP), or BPMPD, external solver available from Cs.Mészáros (LP/QP). In general, QP is a preferred method since it tries to make the distribution of orbit weights more uniform, penalizing "outliers" with high weights.

When the solver has processed the optimization problem, its results are displayed in

info box in the right panel, including the number of infeasible constraints (if the problem cannot be solved), and a few quantities indicating the quality of solution (entropy – a measure of non-uniformity of orbit weights, ratio of max to average orbit weight, and fraction of orbits comprising 50% of total model mass).

Export grid button creates a text file with statistics about the solution (after it was obtained). This file format is described in Sec. 4.4.6.

View options switch between various plots:
Grid cell displays the model constraints (blue for feasible, red for infeasible, for which the difference from the required value is $> 1\%$). The meaning of coefficients depends on the variant of *SM*. For Classic and Cylindrical *SM*, it is mass in corresponding spatial cell, which are displayed in a projection on the 2d plane similarly to stationary start-space points – for each radial shell separately (only for Classic model), or together. For SHGrid and BSE variants, these are coefficients in expansion, arranged in 2d array so that each line shows spherical-harmonic coefficients for given radius (in SHGrid) or index of radial basis function (for BSE); they are further separated in groups of $1, 2, 3, ...$ coefs for $l = 0, 2, 4, ...$, each group having $l/2 + 1$ coefficients for $m = 0, 2, ..l$. Right-click on a cell displays some information in the message area.
Anisotropy shows the value of β for each radial shell (horizontal axis is the shell number).
Orbit weights are shown with the horizontal axis being the orbit number; regular and chaotic orbits are colored blue and red. Right-click on a point does the same as in frequency map plot.
Weight histogram displays these weights as a cumulative distribution function. If it has only a small percent of orbits at the right end of the plot, this signals that the model is over-constrained or even infeasible.
For Classical *SM* one may display any particular **grid shell** or the entire model (in the latter case all cell centers at all radii are simultaneously projected on the plot, which may be confusing but allows to quickly identify infeasible ones). For the other three variants all constraints are displayed simultaneously on a two-dimensional plot.

3 Console scripting

The console variant may perform the same set of operations either using interactive commands entered in the command prompt, or feeding them as a script from a text file (by running `smilec scriptfile`). Below follows the command reference (spelling is case-insensitive).

`Exit`. Obvious (not necessary in a script file).

`ReadIni("file.ini")` (or `LoadIni`, `ReadConfig`, `LoadConfig`) normally should be the first command in a session (unless one is satisfied with default parameters loaded from `smile.ini`).

`ImportOrbits("orbitsfile")` (or `LoadOrbits`) loads orbit library from text file; `ExportOrbits("orbitsfile")` (or `SaveOrbits`) stores orbit library (after building frequency map, etc.). Equivalent to the **Import/Export** buttons on the *Frequency map* page of the GUI version, with the difference that the "Export" command does not auto-

matically store configuration in accompanying `orbitsfile.ini` file (since the configuration anyway has to be loaded from an `ini` file prior to computations). Loading an orbit library, however, includes an attempt to load a corresponding `ini` file.

`ImportModel("orbitsfile")`, `ExportModel("orbitsfile")` – same as above, but also load/stores binary files `orbitsfile.schw` (with *SM* data arrays for each orbit) and `orbitsfile.smpl` (with sampling points from trajectory), see Sec. 4.3. Equivalent to **Import/Export** buttons on *Schwarzschild* page, again without saving an `ini` file on export.

`ExportPotential("potentialfile")` creates a text file with potential data described in Sec. 4.4.7. If an orbit library was loaded, the potential/forces/density is also sampled at the location of points of the library and stored in a separate file `<potentialfile>.points`, and if the potential is BSE or Spline, its coefficients are stored in `<potentialfile>.coefs`

`ExportNbody(NumberOfBodies, "nbodyfile"[, "Format"[, RefineFactor]])` creates the N-body representation of *SM* and writes a snapshot file in the given format (variants include "Text", "Nemo" and "Gadget", the latter is available only if the program was compiled with the UNSIO library; see Sec. 4.5). Refine factor may be used to create a model with unequal particle masses, having a finer mass resolution in the centre. Default is the text format and a zero refine factor.

`RenderDensity(NumberOfPoints, "nbodyfile"[, "format"])` creates a visual representation of density in the model, by sampling it with the required number of points (same as the standalone program RenderDensity, Sec. C.2, with the difference that it always operates on the entire potential and not on the density model that might have been used in computing potential expansion coefficients).

`BuildFreqMap` creates an orbit library with the number of points specified in the `Frequency_map` section of INI file, for the energy or Jacobi constant given in the `Orbit` section. `BuildFreqMapExist` integrates the orbits with the initial conditions loaded earlier from an orbit library file.

`BuildSchw` (or `BuildModel`) and `BuildSchwExist` do the same except that first a model instance is created (with the grid parameters specified in the `Schwarzschild_model` section of INI file), and the *SM* data (such as velocity dispersion, cell mass fraction or potential expansion coefs, depending on *SM* variant) and sampling points are also recorded. They later can be saved by `ExportModel` command.

`SchwLinear`, `SchwQuadratic` start the corresponding solver routine, after the model has been loaded by `ImportOrbitsSchw` or created by `BuildSchw`.

`ExportStats("statsfile")` writes out the text file with Schwarzschild model statistics (Sec. 4.4.6).

4 File formats

Non-bulky data is kept in text files (with tab- or space-separated values). The most important is the orbit library file, which contains initial conditions and results of analysis for a set of orbits. It is accompanied by a configuration (INI) file which contains all the necessary information about this orbit library (potential, integration parameters, chaos criteria, etc.). Some text file formats are only for export purposes.

4.1 INI parameters

Here is the list of all options and parameters in the configuration file `smile.ini` [with alternative names] (and their default values). The parameters are split into several `[sections]`.

`[Potential]` – properties of potential/density model. If there are more than one components in a composite model, then the parameters for the additional components are listed under sections `[Potential1]`, `[Potential2]`, etc., while the first component has no zero in the section name.

`Type` – variants: Logarithmic, Harmonic (default), Dehnen, Scale-free, Scale-free SH, Miyamoto-Nagai, Ferrers, BSE, BSECompact, Spline, CylSpline, Spherical, Nbody.

`Symmetry` – None, Reflection, Triaxial (default), Axisymmetric, Spherical.

`Density` [or `DensityType`] – for BSE/BSECompact/Spline/CylSpline/Spherical potentials, this is the underlying density model used to compute coefficients. Variants: Dehnen, Ferrers, Miyamoto-Nagai, Plummer, Perfect Ellipsoid, Isochrone, NFW, Sersic, ExpDisk, and three other options which load or compute coefs from a text file given in `NbodyFile`: `Coefs` means that pre-computed coefficients are loaded from that text file (Sec. 4.4.4), `Nbody` – coefs are calculated from a set of point masses stored in that file (Sec. 4.4.1), and `Ellipsoidal` or `MGE` – calculated from an Ellipsoidal mass model or a Multi-Gaussian Expansion specified in the text file (Sec. 4.4.2 and 4.4.3).

`NbodyFile` [or `File`] – for treecode N-body potential this is the name of the file with the set of point masses, for BSE/Spline potentials it may be any of the above described three kinds of text files.

`q_YtoX, p_ZtoX` [or simply `q, p`] `(1)` – axis ratios y/x and z/x, must be $p \le q \le 1$.

`Mbh` [or `BHmass`] `(0)` – central point mass (supermassive black hole); for a composite modelonly the value from the first component is used.

`Mass (1)` – total mass of the density model.

`scalerad` [or `Rscale`] `(1)` – scale radius of the density model; its meaning depends on the model type. For instance, for the logarithmic potential it is the core radius, while for the Dehnen profile it is the scale radius.

`scalerad2` [or `Rscale2`] `(1)` – second scale radius of the density model (used for the Miyamoto–Nagai and NFW profiles).

`Gamma (1)` – index of power-law cusp for Dehnen and scale-free potentials.

`SersicIndex (4)` – n parameter of the Sérsic density profile.

`Ncoefs_radial, Ncoefs_angular (20, 6)` – number of radial and angular terms in BSE, Spline and Scale-free SH potential expansions (for analytic density models which have at least triaxial symmetry, the number of angular coefs $l_{\max}$ is even; 0 means spherical model only).

`Ncoefs_vertical (20)` – for the Cylindrical spline expansion, the number of nodes in z direction.

`Alpha (0)` – shape parameter in the BSE potential; 0 means auto-detect, allowed range is $0.5 - \infty$, preferred values are $1 - 2$.

`Rmax (1)` – radius of compact support for the BSECompact potential.

`treecodeEps (-2)` – ϵ, softening length used in frozen-N-body integration. Negative means adaptive softening based on local interparticle distance.

`treecodeTheta (0.5)` – tree opening angle for N-body potential.

`splineSmoothFactor (1)` – value of ΔAIC (57) determining the amount of smoothing of spline coefficients initialized from an N-body snapshot.

`splineRMin`, `splineRMax (0)` – determines the extent of the radial grid in Spline and Cylindrical spline potentials; 0 means compute by default.

`splineZMin`, `splineZMax (0)` – determines the extent of the grid in z direction for the Cylindrical spline potential; 0 means compute by default.

`[Orbit]` – parameters of orbit integration.

`x, y, z, vx, vy, vz` – initial conditions (IC).

`E` – IC energy or Jacobi constant (in case of rotating frame).

`useE (false)` – whether to use energy or coordinates (in the former case, x is initialized to be long-axis radius for given E. Makes sense for building frequency map at given E).

`N_dim (3)` – 2 or 3.

`Omega (0)` – angular frequency of figure rotation of the potential (it is listed in the orbit integration section because it has no effect on the properties of the potential itself, only on the equations of motion in the rotating frame).

`integratorType (default)` – choice of the ODE integrator: default is the 8th order Runge–Kutta (`DOP853`) for all potentials except N-body, and the `Leapfrog` for the latter. Another general-purpose integrator for smooth potentials is `IAS15`, a 15th order adaptive-timestep integrator which can easily be accurate to machine precision and is comparable in performance to `DOP853`, while having better accuracy in most cases. Yet another option is the 4th order `Hermite` integrator, which uses the information about force derivatives in a predictor–corrector scheme with only two force evaluations per timestep. It uses adaptive timesteps and in general is quite well suited for potentials that are not infinitely smooth (i.e. spline expansions). Other variants are possible if compiled using the ODEINT library: `DormandPrince5`, `CashKarp5` and `RungeKutta3a` are adaptive-timestep Runge–Kutta integrators (of order 5 and 3), `RungeKutta4` and `SymplecticRK4` are fixed-timestep 4th order Runge–Kutta methods (the latter is symplectic), and `BulirschStoer` is the eponimous adaptive-timestep, variable-order method. Of these, `DOP853`, `DormandPrince5` and `BulirschStoer` have comparable efficiencies, `IAS15` is the best choice for high accuracy in infinitely smooth potentials, `Hermite` also does a good job at intermediate accuracy for Spline potentials, and other methods are largely for demonstration purposes.

`accuracyRelative, accuracyAbsolute (1e-9)` – accuracy parameters for variable-timestep ODE integrators (except the Leapfrog integrator for the N-body potential). The values are the upper bounds on estimated error per integrator timestep; usually the energy conservation error per one *dynamical time* $T_{\rm orb}$ (not per internal timestep) appears to be of the order of these accuracy parameters. For Hermite the mathematical meaning of the accuracy parameter is somewhat different, but it yields comparable errors for the same settings as for other integrators. Note that a nonzero `accuracyAbsolute` is breaking the scale-invariance of the integrator (for instance, orbits that come to very small radii can

have an absolute error in position, say, 10^{-9}, but relative error for $r = 10^{-5}$ will be only 10^{-4}) – therefore it may be better to set the absolute error tolerance to zero..

`accuracyIAS15 (1e-4)` – accuracy parameter for the IAS15 integrator; its meaning is less straightforward, but a rule of thumb is that 10^{-3} gives an energy conservation error less than 10^{-8} per 100 $T_{\rm orb}$, and 10^{-4} is close to machine precision, while being only 20% slower. Also for higher accuracy the accumulation of energy error is not monotonic, as for other integrators, but rather a random walk, therefore it is better suited for long ($\gg 10^3\,T_{\rm orb}$) integration intervals. However, keep in mind that for the Spline potentials the accuracy of all high-order integrators is limited by discontinuities of the 3rd derivative of the potential, therefore one shouldn't expect the energy error to be smaller than $10^{-7}..10^{-6}$.

`accuracyFixedTimestep (0.005)` – for fixed-timestep ODE integrators gives the timestep in units of $T_{\rm orb}$.

`accuracyTreeCode (0.25)` – η, factor in timestep selection for the Leapfrog integrator used for the N-body tree-code potential. The timestep is taken to be $\tau = \eta \times \min(l/v, \sqrt{l/a})$, where l – distance to nearest particle, softened (i.e. it is $\sqrt{l_{\rm true}^2 + \epsilon^2}$), v – particle velocity, a – acceleration.

`treecodeSymmetrizeTimestep (false)` – whether to use Hut et al.(1995) [11] algorithm for time-symmetrizing adaptive timestep, to improve energy conservation at the expense of almost twice as slower integration.

`intTime (100)` – integration time in units of long-axis period ($T_{\rm orb}$). Zero value means using per-orbit data stored in orbit library file, when integrating orbit library with pre-loaded initial conditions.

`samplingInterval (0.02)` – timestep for recording the sampling points from the orbit, in units of $T_{\rm orb}$: determines the maximum orbit frequency which can be detected, and the smoothness of rendered orbit. Has nothing to do with integration timestep.

`calcLyapunovMethod (0)` – whether and how to compute Lyapunov exponent. 0 means no computation, 1 is the integration of a nearby orbit, 2 is the variational equation. The latter option might be slower but potentially more reliable.

`relaxationRate (0)` – amplitude of velocity perturbation that mimics the effect of two-body relaxation. Its numerical value corresponds to $N^{-1} \ln\Lambda$, where N is the number of stars in the stellar system and $\ln\Lambda \approx \ln N$ is the Coulomb logarithm.

`usePS (false)` – whether to use Poincaré surface of section (makes sense only in 2d).

`intTimeMax (0)` – maximum integration time (in $T_{\rm orb}$) if Pfenniger's adaptive method [14] is used (only in the context of Schwarzschild modelling; 0 to disable). [currently unused].

`intTimeMinAbs (0)` – minimum orbit integration time in global N-body units (not in energy-dependent orbital periods). This could be used if the dynamical times vary enormously throughout the model, to ensure that even the most tightly bound orbits are integrated for a minimum amount of physical time. The most apparent application is for the models with central massive black holes, when we want to make sure that all orbits perform at least several precession cycles (so that the minimum integration time is a few times longer than $T_{\rm orb}$ at the influence radius of the black hole).

`adaptiveTimeThreshold (0.05)` – controls the Pfenniger's method: if difference in SM constraint values between two halves of integration time exceeds this threshold, continue integration further until this difference becomes less or the maximum integration

time is reached. [unused]

`[Frequency_map]`
`numOrbitsStationary, numOrbitsPrincipalPlane, numOrbitsYalpha, numOrbitsRandom` – number of points in corresponding start spaces.

`[Schwarzschild_model]`
`densityModelType` – `(Classic), Cylindrical, SHBSE, SHGrid`, specifies the type of density model.

`densityComponents (all)` – list of potential components (if more than one) that are used for the density profile in a multi-component Schwarzschild model (Sec. B.7). For instance, if there are three [Potential*] groups, for the disk, bar and halo, then one might set this parameter to `0,1` to use the combined disk and bar potentials as the target density for the model.

`numOrbitsRandom (1000)` – number of orbits in the entire model.

`ICgenerator` – `(Eddington), JeansSph, JeansAxi`, specifies the method for generating initial conditions for the orbit library (Sec. B.8).

`ICbeta (0)` – velocity anisotropy parameter for JeansSph IC generator.

`ICbetaAxi (0), ICkAxi (1)` – meridional velocity anisotropy parameter and rotation parameter for JeansAxi IC generator.

`weightSkewFactorE (0)` – if positive, create more orbits at high binding energies (near the center); the value is the ratio of maximum to minimum orbit weight priors minus 1, i.e. if it equals 9, then the outermost orbits will have 10 times larger weight (on average), and will be the same 10 times less numerous.

`weightSkewFactorL (0)` – same mechanism applied to the distribution of orbits in relative angular momentum squared (normalized to the angular momentum of a circular orbit with the same energy). Applies to Eddington IC generator; if positive, create more orbits with low angular momentum.

`numRadialCoefs (20)` – in BSE *SM*, number of radial coefs; in SHGrid, Classic and Cylindrical *SM*, number of grid points in radius (cylindrical radius for the latter).

`numAngularCoefs (6)` – in BSE and SHGrid *SM*, number of angular coefs (should be even); in Cylindrical *SM*, number of grid cells in angular direction.

`numVerticalCoefs (15)` – in Cylindrical *SM*, number of grid cells in z direction.

`linesPerSegment (2)` – in Classic *SM*, split each shell into $3 \times (\text{linesPerSegment})^2$ cells.

`SMAlpha (0)` – in BSE model, the shape factor in the basis set. 0 means the same as in the potential if the latter is also BSE, or autodetect based on the potential's inner density slope.

`numShells (20)` – number of radial shells to store the velocity dispersion information in all variants of *SM*.

`innerShellMass, outerShellMass (0, 0)` – fraction of mass enclosed by the innermost and the outermost shells in radius, for the Classic *SM* and for the velocity dispersion grid. Zero values mean default 1/(numShells+1) and numShells/(numShells+1).

`smGridYtoX, smGridZtoX (1, 1)` – flattening (axis ratio) of the spatial grid in Classic *SM*.

`chaoticMinFreqDiff (1e-3)` – threshold in frequency diffusion rate ($\Delta\omega$) separating regular from chaotic orbits. Used to plot them in different colors on the frequency map, and to increase or reduce fraction of chaotic orbits in Schwarzschild model.

`chaoticMinLambda (0)` – same threshold in Lyapunov exponent (Λ). If it is not calculated, this has no effect; if it is, then the default value of 0 just separates orbits with detected signs of chaos ($\Lambda > 0$) from those for which chaotic behaviour was not detected (the latter are assigned $\Lambda = 0$).

`chaoticPenalty (0)` – if positive, penalize usage of chaotic orbits; negative – prefer them. May take a continuous spectrum of values, although most of the effect is felt when this factor is of order ± 1. For larger values, the penalty for wrong type of orbits may outweight that of violating *SM* constraints, so the model becomes infeasible.

`constraintPenaltyLin, constraintPenaltyQuad (1, 1)` – penalty factors for constraint violation (applies for both density and kinematic constraints, if the latter are used). See Sec. 2.2.7 or B.6 for the description.

`regularization (0.1)` – λ factor (66) which drives the distribution of orbit weights towards a more uniform one.

`maxWeight (0)` – max.weight of a single orbit. 0 means no restriction.

`constrainBeta (false)` – whether to constrain velocity anisotropy coefficient β in the solution.

`betaIn, betaOut (0, 0)` – values of β for the inner and the outer radial shells (linearly interpolated in enclosed mass between these two values).

`constrainAngMom (false)` – whether to constrain angular momentum distribution; if yes, then the solver tries to achieve the discretized distribution of relative angular momentum which corresponds to the given anisotropy coefficient β, using the same inner/outer values as above, but interpolating them in the energy space. This is similar in effect to constraining β directly, but operates in the energy space, not in configuration space, and imposes more demanding constraints if the number of angular momentum bins is more than two.

`numAngMomBins (4)` – number of equal-width bins in relative angular momentum $L/L_{\rm circ}(E)$.

`numSamplingPoints (0)` – number of sampling points from each orbit that are stored in `.smpl` file. They are drawn randomly from the trajectory after orbit integration, and later used in creating N-body model from Schwarzschild model, so if one doesn't need this feature, this number may be set to zero.

`[Common]` is the section for global parameters.

`WorkDir` – default working directory name (for GUI file open/save dialogs).

`TempDir` – directory for temporary files used to exchange data with external programs (qdelaunay and bpmpd).

`MaxThreads` – number of parallel threads in orbit library integration (0 is default, equal to the number of processor cores).

`useBPMPD (true)` – whether to use external solver `bpmpd.exe` for linear/quadratic optimization problem. It is a lot faster on large problems than GLPK, and may handle quadratic problems (preferred mode), but the publicly available version is limited to small problems (approx.250 orbits). You may ask the author, Csaba Mészáros, for the

unrestricted version (as did I:-). If this option is turned off, or if `bpmpd.exe` executable is not present in the application dir, then GLPK library is used as the linear solver and CVXOPT, written in Python, as the quadratic solver (if the program was compiled with its support; to use CVXOPT also for linear problems pass 0 as `regularization` parameter in the quadratic solver).

4.2 Orbit library file

This is the main exchange format used for keeping orbit initial conditions and integration/classification results. This file type is also used to export data to N-body model (if this is requested in addition to creation of binary NEMO snapshot file), and to load particles representing N-body or BSE potential (in this case only 7 first fields are used). Each line contains the following data:

`x y z vx vy vz weight weightprior inttime maxtime timeunit...`
`... energy ediff lfx lfy lfz lfdiff lambda description ...`
`... inertx inerty inertz Lxavg Lxvar Lyavg Lyvar Lzavg Lzvar ...`
`... L2min L2slope fitscatter fitsignificance L2circ`

First 7 fields are orbit initial conditions and mass (which means orbit weight in Schwarzschild model, so it may be zero), in the same order as in the simple N-body interchange file (Sec. 4.4.1). This is, generally speaking, sufficient to load any text file containing this data as initial conditions for orbit library, although if no `timeunit` is provided, it will be calculated on-the-fly in the corresponding potential, which takes some time if the number of orbits is large. For 2d orbits z and v_z are zero. The other fields are either directly derived from initial conditions or are results of orbit integration and analysis.

`weightprior` is the "expected value" of orbit weight if all orbits were equally probable. Makes sense if the initial conditions used non-uniform sampling (if weightSkewFactorE or weightSkewFactorL are non-zero). If it is omitted then all orbits are assigned a uniform prior.

`inttime` is the integration time (in common time units, not in periods), and `maxtime` is upper limit on adaptive integration time (!temporarily defunct!).

`timeunit` is the dynamical time (period of long-axis orbit with the same energy) which serves as the unit of time and frequency for this orbit (same as $T_{\rm orb}$ in GUI); `energy` is (initial) total orbit energy (or Jacobi constant in the case of figure rotation), and `ediff` is energy conservation error.

`lfx`, `lfy` and `lfz` are the leading frequencies in three coordinates, and `lfdiff` is the Frequency Diffusion coefficient.

`lambda` is the Lyapunov exponent (if it was calculated, otherwise -1).

`description` is the text string containing orbit class and possibly "chaotic" attribute (based on analysis of spectrum, not a reliable estimate, see Sec. 6). This text line has underscores instead of spaces, in accordance with the requirement that any space- or tab-separated file may be loaded.

`inert{x/y/z}` are the mean-square values of each coordinate, basically these quantities measure the extent of an orbit in each direction (average, not maximal).

`L*avg` and `L*var` are average value and mean-square scatter of angular momentum about

each axis.
`L2min` and `L2slope` are coefficients in the linear regression fitting the distribution of squared angular momenta at pericenter passages. If `L2min=0` this means that the orbit is centrophilic. The next two parameters, `fitscatter` and `fitsignificance`, assess the quality of fit (see Sec. B.5 for the details of the algorithm). `L2circ` is an approximate squared angular momentum of a circular orbit with this energy (useful scaling parameter for L2min).

The orbit data make sense only in conjunction with corresponding potential, so each orbit library file is accompanied by INI file. Upon import, first INI file (if exists) is read, the potential is initialized, then the orbit library is loaded. Exporting orbit library also creates INI file.

4.3 Binary files

There are two kinds of binary files used in Schwarzschild modelling module: one (`.schw`) contains the data for *SM* (its content depends on the variant of *SM* chosen), the other (`.smpl`) contains sample points from trajectory of each orbit, used to generate an N-body snapshot from a *SM*.

Two alternatives for binary data storage are implemented: the first is a structured HDF5 file, the second is a simple binary array dump (the code can be compiled with only one of these variants). HDF5 is a preferred storage model as it is more portable and commonly used.

If HDF5 is used as the back-end for data storage, the format is the following. For the Schwarzschild data file, orbits are stored in the dataset `/Schw` of compound records, in which each data block is represented as a fixed-length array of necessary size with the name of this data object. That is, for a BSE density model with 20 radial terms and $l_{\max} = 4$ (i.e. 6 angular terms), it will be (a) an array of 20×6 floats with the name `SHBSE`, (b) for the shell-kinematic data with 25 radial shells, another array of 25×3 floats with the name `KinematicShell` (the 3 numbers being the time spent in the shell, and radial and tangential velocity dispersions), and (c) for the angular momentum distribution with 4 bins, and array of 4 floats with the name `AngularMomentum`. All orbits are required to have the same set of Schwarzschild data objects. In the future, additional types of Schwarzschild data objects may be stored using the same named scheme. For the trajectory sample file, the dataset `/Traj` contains a variable-length array for each orbit, the elements of this array being a 6-float compound record (3 for `Pos` and 3 for `Vel`); number of sampling points may vary between orbits. Both `.schw` and `.smpl` files have additional extension `.h5`.

In the alternative case of a "proprietary" data storage model, files have the following format. In the Schwarzschild data file, for each orbit the number of data objects is written as 4-byte integer, then for each object its type and size are stored as 4-byte integers, followed by the bulk data array of floats. In the trajectory sample file, for each orbit the length of sample $n_{\rm points}$ is written (4-byte integer), then the array of $6 \times n_{\rm points}$ floats representing the position and velocity points is stored.

4.4 Auxiliary text files

All text files should be tab- or space-separated. Lines starting with symbols `#` `%` are ignored.

4.4.1 Point file

This is a simple text file for loading/storing point mass sets: each line contains
`x y z vx vy vz m`

It is used as an input data for initializing BSE/Spline/treecode N-body potentials, for exporting N-body model in a text format, and also may be used to load initial conditions for orbit library / Schwarzschild model, since it is a truncated version of orbitlib file (Sec. 4.2).

4.4.2 Ellipsoidal or Spherical model file

The Ellipsoidal mass model, which may be used as an input to BSE or Spline potential, is a very generic representation of arbitrary density profile with arbitrarily varying axis ratios. It is given by a text file containing pairs of r $M(r)$ values describing dependence of enclosed mass on radius; each line may also contain two more values which are taken for axis ratio at a given radius. The density of a spherically symmetric model is calculated as a spline interpolation of density profile from the first two columns (log scaled in both r and ρ). Axis ratios are also spline interpolated in log radius between the provided radial points, and assumed to be constant below the first and above the last radii with provided values. If only one line with axis ratio values is present, they are assumed to be constant. If no values are provided the model is considered to be spherical. The first line of the file should contain the text `Ellipsoidal`.
To compute the actual density at a given x, y, z, first the spherical radius is computed; then the interpolated values of q, p are calculated and the three coordinates are scaled accordingly, so that the product of three scaling coefficients is unity. In other words, $s \equiv (qp)^{-1/3}$ is the common scaling factor, and the elliptical radius is computed as $\tilde{r}^2 = (x/s)^2 + (y/sq)^2 + (z/sp)^2$. The density is then given by spline interpolated $\rho(\tilde{r})$. By construction, the total model mass is not exactly equal to the mass of spherical model given in the last line of the text file, but is typically very close to it.

Spherical mass model is a simplified case of Ellipsoidal where q and p are unity and are not provided in the file. It corresponds to a Spherical potential (which behaves similarly to Spline potential with $l_{\max} = 0$). The file extension should be `.mass` and its first line does not need to contain anything special (in fact, all lines not starting with a number are ignored).

4.4.3 Multi-Gaussian expansion file

Another generic representation of an arbitrary density profile is the Multi-Gaussian expansion, in which density is given by a sum of $N_{\rm comp}$ components, each being a triaxial Gaussian with a given scale length and normalization. This file may be used as an input to the BSE or Spline potential expansion. The first line of the text file should contain

the word `MGE`, and each line contains the data for each component: r_s M and optionally q and p, where r_s is the scale radius, M is the mass in this component, and the other two values give the axis ratios. The density profile of the component is thus given by

$$\rho(x,y,z) = \frac{M}{(2\pi)^{3/2} pq\, r_s^3} \exp\left[-\frac{x^2+(y/q)^2+(z/p)^2}{2r_s^2}\right] \quad (1)$$

If one has a MGE in the observer's units (central luminocity density Σ in $L_\odot/\mathrm{pc}^2$, width of the component σ in arcsec, and the projected flattening q'), then the mass of each component is given by $2\pi r_s^2 q' \Sigma \Upsilon$, where $r_s = 0.485(\sigma/1'')(D/1\mathrm{Mpc})$ pc is the width of the component in physical units, and Υ is the mass-to-light ratio.

4.4.4 Potential coefficients file

Stores coefficients for BSE, Spline and Scale-free SH potentials. This file is automatically created when a potential is initialized from a point mass set, and later may be used to load the same potential without spending time on computing the coefs. The first few lines are the header:
`BSEcoefs/BSECcoefs/SHEcoefs/CylSpline/SFcoefs` – specifies potential type (BSE, BSECompact, Spline, CylSpline, or Scale-free SH);
`n_radial` – number of radial coefs or radial grid points, correspondingly;
`n_angular` – $l_{\max}$, order of angular spherical-harmonic expansion; 0 means just one coefficient for a spherically symmetric model, or an axisymmetric CylSpline;
`alpha` – shape parameter for BSE potential, or `Rmax` – radius of compact support for BSECompact, or `Gamma` – exponent for Scale-free SH, or `n_vertical` – number of coefficients in z direction for CylSpline, or unused for Spline;
`time` – presently unused;
`#commented out` line (except CylSpline; text header for the table below).
The rest of file is the coefficients table with the format depending on the potential type. For BSE/BSECompact/Spline/Scale-free potential all angular coefficients for a given radial index (BSE) or radius (Spline) are written in one line. First number is radial coefficient index or radius, second is the $l = 0$ coef (spherical part), and so on. If the number of fields in a line is less than $1+(n_{\mathrm{angular}}+1)^2$, the rest is filled with zeroes; if it is greater then the rest is ignored (so one may adjust the numbers in header without changing the table). For Scale-free potential there could be only one radial coefficient.
For CylSpline the format is somewhat different: the file consists of several blocks that list two-dimensional coefficients for a given azimuthal harmonic m; there could be at most $2l_{\max}+1$ such blocks, but the blocks with all zeroes may be omitted. Each block is preceded by a line with the value of m; then a line with the list of radial grid points (starting from zero and preceded by `#z\R`); then $2n_{\mathrm{vertical}}-1$ lines for the z grid (it contains symmetrically positive and negative values of z, and the values of coefficients are usually symmetric w.r.t. change of sign of z). Each line starts with the value of z for the vertical grid node, then n_{radial} values of coefficients. For the $m = 0$ harmonic these are just the values of the potential in the meridional plane.

The coefficients file extension typically corresponds to the potential type as follows: `.coef_bse` (BSE), `.coef_bsec` (BSECompact), `.coef_spl` (Spline), `.coef_cyl` (Cyl-

Spline), `.coef_sf` (Scale-free SH); this ensures a correct determination of file type when using the simplified interface of SMILEPOT library.

4.4.5 Orbit file

This file type is used to export or import trajectory; each line contains the following data: `time x y z vx vy vz`

Upon import of such file, the orbit is assigned initial conditions from the first line, and all the relevant parameters (energy, dynamical time $T_{\rm orb}$ and unit of frequency) are calculated from the current potential parameters (which, however, are not stored along with the orbit).

May be useful to import an orbit recorded from N-body simulation, with potential expressed as frozen-N-body or BSE taken from density profile from the same simulation, and check how does this orbit look like if re-integrated in a fixed potential.

4.4.6 Schwarzschild grid statistics file

Used for export only, this file contains statistical information about Schwarzschild model grid (and hence can be created only when a model is created or loaded). For the density constraint block, the following values are written (one constraint per line): `index required actual diff norm {decomp}`,
`index` is just the index of the constraint;
`required` is the required value of this constraint in the density model (e.g. the mass in the given cell of the Classic *SM*);
`actual` is the value obtained by the orbit superposition;
`diff` is the difference between these two (best if zero);
`norm` is the normalization factor used to scale the contribution of constraint deviation to the penalty function;
`chaos` is the fraction of chaotic orbits that contribute to this constraint;
`numorbits` is the total number of orbits that could in principle contribute to this constraint (have a non-zero matrix element for this index);
`numorbitsnonzero` is the number of orbits that actually do so because they have a positive weight in the model;
`decomp` is the decomposition of index into readable numbers: for the Classic and Cylindrical models, it is the radius and x, y, z values of the cell center; for the SHGrid model, it is the radius and l, m indices of angular expansion; for the BSE model, it is n, l, m triplet of basis function indices.
For the kinematic constraint block, the following data is written for each radial shell:
`index radius M(r) sigma_r^2 sigma_t^2`,
where $M(r)$ is the enclosed mass in this shell, and σ's are the radial and tangential velocity dispersions from the orbit superposition in this shell.

4.4.7 Potential sampling file

Used for export only. Each line contains potential (Phi), density (Rho) and forces (F) sampled at a given point.

`x y z Phi Fx Fy Fz Rho`

Points are logarithmically spaced in radius from 0.001 to 1000 and lie along seven lines: principal axes, diagonals of principal planes ($z = 0, y = x$, etc.) and the diagonal $x = y = z$. NB: if an orbit library was non-empty, `ExportPotential("...")` also creates another file in the same format with additional extension `.points`, containing potential/force/density sampled at locations of points in orbit library. This data may be useful, for example, to test the accuracy of BSE/Spline approximation.

4.5 N-body snapshot files

An N-body snapshot used to initialize the BSE, Spline or Frozen-Nbody potential, or created as the representation of the Schwarzschild model, may be in one of the supported formats. Presently, there are the following options available: a text file (Sec. 4.4.1), a NEMO snapshot format, or a GADGET snapshot file. The latter two options are available if the program was compiled with the UNSIO library; however, export to NEMO format is possible without this library, using built-in routines. Reading a snapshot file during the potential initialization determines the file format automatically.

5 A short guide to practical Schwarzschild modelling

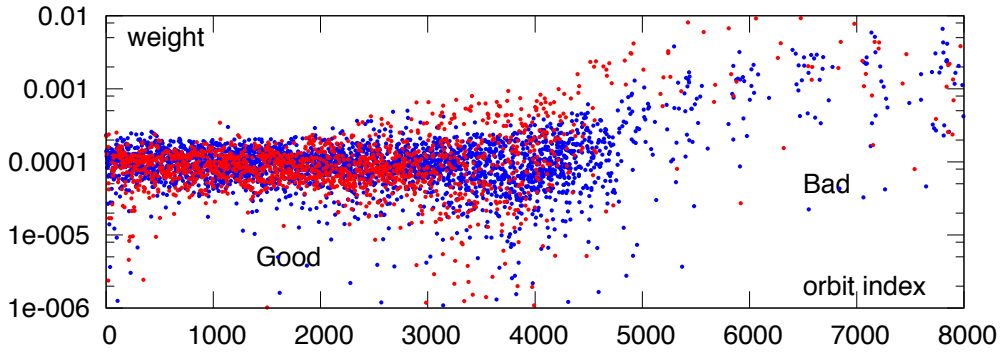
One of main applications of SMILE is to create equilibrium N-body models with a predefined triaxial density profile (and possibly a given velocity anisotropy profile). Here we outline the necessary steps and checks to be made.

- Choose a density profile out of existing variants, or implement one as an Ellipsoidal or MGE mass model described by a text file. One may also use an N-body snapshot as a Monte-Carlo realization of a density profile; however it is less preferable due to inherent discreteness noise which translates into fluctuations of high-order potential expansion coefficients.

- Use BSE, or, preferrably, Spline potential solver with appropriate number of coefficients. For strongly flattened systems such as disk galaxies, CylSpline provides the most accurate representation. For the number of radial coefficients, 20 is a good choice in most situations; for the angular order, it depends on the degree of flattening. A moderately flattened model with $y/x, z/x \gtrsim 0.5$ is fine with $l_{\max} = 6-8$, more flattened systems require more. With CylSpline, the number of angular coefficients depends on the degree of triaxiality (0 is for axisymmetric model with arbitrary flattening in z direction). A good way to check whether the approximation works well is to export a file with potential, forces and density sampled at coordinate axes and/or given set of points (Sec. 4.4.7) and compare density with the expected profile. At the very least, it should not oscillate wildly and drop to zero at some finite radius (a warning will be given if this is violated). Another possibility is to use the `testaccuracy` utility (Sec. C.3).

- Construct a *SM* with some fiducial number of constraints and orbits. A good starting choice is 10^4 orbits for a few$\times 10^2$ constraints, e.g. 20 radial shells and

$2-3$ lines per segment (in case of Classic *SM*) or $l_{\max}=6-8$ (for SHGrid *SM*). Use Quadratic optimization solver to obtain orbit weights. The weight distribution should ideally be fairly flat and close to uniform.



In the example above, the innermost orbit weights are distributed quite well (many orbits with nonzero weights), while the outermost parts of model are overconstrained, which results in quite a few orbits with large weight. This situation may be remedied by using more orbits for the same number of constraints.
For some cases, a model with a given density and flattening profile may not be feasible at all, no matter how many orbits one has. In this case, allowing the flattening rate vary with radius, so that in the outer parts it is close to axisymmetry, may alleviate the problem.
Overall, the very first condition for a good model is its feasibility (that all constraints are satisfied), and the second is to be *under*constrained, so that many orbits are given a similar weight and not just a few outstanding orbits are protruding above the rest. The latter situation is not only dissatisfying aesthetically (one may clearly see these high-weight orbits in the N-body realization), but also unwelcome for the stability of the model. Therefore, it is better to have a model with fewer constraints and a "relaxed" weight distribution, than the one in which a multitude of constraints are barely satisfied with a few orbits.

- Export the model to N-body snapshot. Again, ideally one should not have too many sampling points per orbit, say, $\lesssim 100$ for all but a few orbits. This means that, for instance, to get a 10^6 particle model, one will need few$\times 10^4$ orbits with a reasonably flat weight distribution.

This can be summarized by the following script for the console version (assuming that all relevant parameters are specified in the `.ini` file):

```
# read input parameters
LoadIni("model.ini")
# create orbit library and Schwarzschild model data, this takes most of the time
BuildSchw
# save results, orbit weights are not assigned yet
ExportModel("model")
# do the optimization
```

```
SchwQuadratic
# save results once again, now with orbit weights
ExportModel("model")
# export model to a NEMO N-body snapshot
ExportNbody(1e5, "model_nb", "Nemo")
```

6 Known bugs, subtleties and limitations

- All analytic density models assume triplanar symmetry w.r.t. change of sign of any coordinate. For the general-purpose expansions (BSE/Spline) and frozen-N-body potential it is not necessary: one may choose the desired level of symmetry for the expansion. If initializing it from a discrete set of points, it is crucial that the density center is at origin (except for N-body potential, however even for it this is necessary in order for orbit analysis to work properly), and principal axes of figure should be directed along x, y, z as longest to shortest: while the spherical-harmonic expansion should be invariant to rotation (at least when symmetry type is downgraded to "Reflection") the correct ordering of axes is important for orbit analysis. General-purpose expansions and the entire *SM* module are agnostic to mass normalization of model, but the length scale (typical half-mass radius) should be of order unity (i.e. not too much off, say by a factor of 10), because a number of design choices break the invariance. Therefore, it is recommended to work in dimensionless, N-body units, rather than in parsecs, for example.

- BSE/Spline expansions assume smooth density profiles and a power-law density behaviour at $r \to 0$ with the index $0 \le \gamma < 2$ (they still work pretty well for $\gamma = 2$, but not for steeper profiles which have divergent potential at origin). Inner and outer density asymptotics are assumed to be power-laws, which may introduce some systematic errors for other type of profiles (such as Sérsic), but they are negligible for a sufficient number of terms. Density is assumed to fall monotonically with radius. This also means that any user-defined density profile should be devoid of sharp jumps or abrupt drops to zero, unless you are using the BSECompact or CylSpline expansions which have a finite radial extent.

- *Chaotic* attribute in the orbit description should not be relied upon (it is added when there are lines in spectrum that cannot be fitted as a linear combination of no more than `N_dim` fundamental frequencies, but sometimes the accuracy demanded might be too stringent or too weak). A better indicator is the Frequency Diffusion parameter or Lyapunov exponent. Moreover, in the case of two very nearby spectral lines the method often gets confused and assigns "chaotic" attribute (and even a rather high frequency diffusion rate) to a regular orbit. This typically may be overcome by setting a longer integration time, to better resolve these nearby lines.

- Scale-free potential and its BSE approximation are implemented only for $0 \le \gamma < 2$. In addition, variation equation option for computing Lyapunov exponent is not implemented for scale-free potential, only for its BSE variant. This is not a severe

restriction, as the approximation works fairly good and much faster, so it is the preferred option.

- Computing Lyapunov exponent by integration of nearby trajectory is possible for all potentials (except N-body, of course), and this is the recommended option (`calcLyapunovMethod=1` in the INI file), since it is usually faster than variation equation approach. Only for orbits close to the black hole may it give incorrect results: a regular orbit seems to have nonzero Lyapunov exponent, which is probably due to roundoff errors in keeping these nearby orbits really near. So to study these orbits, one should use "true" variation equation approach (Update: even in this case "chaotic" attribute may be triggered on erroneously for some tightly-bound orbits, requires further study).

- Since floating-point values are stored in text format (in Orbit Library file), sometimes it may happen that re-integration of the same orbit gives a different result. (This definitely may happen if Lyapunov exponent is used, since the deviation vector is initalized randomly). Although some measures were taken to diminish the possible damage of this effect, one should still keep it in mind. In addition, orbits with the same initial conditions may be different on different machines.

- Inner and outer extrapolation in Spline potential is not twice continuously differentiable at the first/last grid nodes; this may trigger false positive Lyapunov exponent for an orbit which extends beyond the last grid node. (Perhaps a lower-order integrator could be used for this potential).

- On 32-bit systems, there seems to be an issue with inefficient memory management which leads to fragmentation of the application heap during Schwarzschild model construction; as a result, the program may run out of memory even if the orbit library size is not very large and could well fit into the 2Gb limit.

- Energy conservation is far from perfect for orbit integration in the N-body potential with adaptive softening length; it is somewhat improved by using timestep symmetrization.

7 Version history and future plans

- 1.0 (2010), 1.1 (2011) – for internal use only.

- 2.0 (July 2013) – first public release.

- 2.5 (January 2015) – major update:

  - New potential and density models, in particular, Cylindrical spline expansion;
  - New methods for generating initial conditions; refinement in energy and angular momentum;
  - New orbit integrators (IAS15, Hermite, odeint family);

– Multi-component potential and Schwarzschild models;
– Figure rotation;

Ideas waiting to be implemented:

- Observationally-driven Schwarzschild modelling;
- Option for using modified Newtonian gravity;
- Implementation of orbit integrator using GPU acceleration;
- Refactor the computation core and parallelization strategy to migrate from Qt threads to OpenMP plus optionally MPI;

A Program structure and compilation

The main program is written in C++ using the Qt framework (for the GUI and for other features like inter-object and inter-thread communication), so it should compile wherever Qt is supported (at least Linux, Windows, and MacOS). Some mathematical parts (in particular, orbit integration and analysis, potential solvers, spherical models) do not use Qt and may be used independently in other programs.

In addition, a number of other libraries and software is used in SMILE (optional components are marked with *): GSL (GNU scientific library) is mandatory for the entire program; (*) interp2d (GSL-like interpolation library) is required to use cylindrical spline and other disk-related features (cylindrical Schwarzschild grid, axisymmetric Jeans equation, etc.). Schwarzschild modelling requires at least one of the following libraries: (*)GLPK (GNU linear programming kit, if this option was selected as the optimization routine), (*)CVXOPT (quadratic optimization solver, requires Python) and/or (*)BPMPD solver (as a standalone application); the latter option is the fastest one but is not publicly available, while the first two are comparable in speed and are free software). Data input/output optionally uses (*)NEMO, (*)UNSIO and (*)HDF5 libraries. GUI version needs Qwt (version 5.x, not 6) and (*)QwtPlot3d[2]libraries for plotting, and (*)`qdelaunay` program from QHull package (to render an orbit as a solid body). Console version benefits from the use of (*)readline library (installed on most UNIX systems).

Build is typical for Qt applications – check the project include file `smile.pri` (common for GUI and console versions) for correct paths to libraries (`INCLUDEPATH`, `LIBS`), run `qmake` (from qt4!), then `make`. There is a global Makefile, which compiles both versions of SMILE, the SMILEPOT library, and additional programs described in Sec. C; you may try it.
On Mac, one might need to run `qmake -spec macx-g++` to use GNU compiler.
`qdelaunay` (compiled separately) and `bpmpd.exe` should reside in the main application folder (the latter is windows-only binary, so it is run using `wine` in Linux/MacOS).

The architecture of SMILE is rather modular and flexible, with common interfaces between various parts allowing for replacement of internal implementation or augmenting

[2]On some systems, it may be called qwtplot3d-qt4

the functionality. More information on the internal structure of the software can be found on the documentation webpage: `http://td.lpi.ru/~eugvas/smile/doc/`.

This program includes code from Hairer et al. DOP853 and Rein&Spiegel's IAS15 ODE integrators, substantially reworked tree-force potential solver HACKCODE by J.Barnes from NEMO toolbox, and simplified NEMO snapshot writer by S.Rodionov. Everything else is written from scratch. You may use any part of the program in any your project.

A.1 smilepot library

To facilitate the use of the extensive and flexible framework for representing potential and forces in other programs, a subset of the SMILE codebase is included in a separate library LIBSMILEPOT. It comprises the code for dealing with potentials and spherical mass models, reading of potential coefficients files (Sec. 4.4.4) and N-body snapshots (to compute these coefficients from a set of points), and parsing the potential parameters from the `[Potential]` section of an INI file (Sec. 4.1). The library has simplified C and PYTHON interfaces that allow to initialize the potential from the parameters from an INI file, or by specifying its parameters directly, and provide functions for computing potential, density, forces and force derivatives.

The C/C++ interface is accessed by including the file `smilepot.h` and offers several way of constructing the potential: `smilepot_create("params.ini")`, taking the parameters from an ini file, or `smilepot_create("file.coef_spl")`, loading potential coefficients (in this case for a Spline potential) from a text file; `smilepot_create_params`, using parameters provided as a list of "key=value" arguments; or `smilepot_create_from_particles`, constructing a potential expansion from an N-body snapshot. Of course, C++ programs may additionally include some other header files and access the entire SMILE machinery related to potential creation and manipulation.

The PYTHON interface is accessed by importing `py_smilepot.py` and creating an instance of `py_smilepot.smilePot` class, which has methods like `potential`, `density`, `force`, using a list of `keyword=value` parameters for the constructor. One may either take the parameters from an INI file with `file="file.ini"`, or specify the potential type and all relevant parameters directly, like `type="Spline", density="Dehnen", Gamma=1.5, mass=42.0, scalerad=0.1234, q=0.9, p=0.75`

See `readme_smilepot.pdf` for a more detailed usage description.

B Technical details on the algorithms and formulae used

B.1 Frequency analysis and orbit classification

Orbit classification is based on detection of most prominent spectral lines in Fourier spectrum of trajectory in each coordinate. Here we summarize the method used to extract spectral lines.

We start from computing complex Fourier transform $c_i, i = 0..[N/2]$ of input time series $x_k, k = 0..N-1$. At each iteration, we locate the most prominent line in the

spectrum and then subtract it from c_i, until the maximum number of lines has been reached or the amplitude of lines drops below a certain fraction of the amplitude of the first line. First we locate the integer index m so that $|c_m|$ has the maximal value over the remaining spectrum, and then find a fractional correction s so that the exact location of line is $m+s$. If possible, the correction is determined by a more precise method of Hunter(2002) [12] with Hanning window filtering; if not a more approximate method of Carpintero&Aguilar(1998) [13] is used. [TODO: explain more details].

After all prominent lines have been determined for all d coordinates, we search for at most N_d fundamental frequencies Ω_k so that all line frequencies are expressed as linear combinations of these (within a certain tolerance): $\omega_{d,j} = \sum_{k=1}^{N_d} a_{djk}\Omega_k$ with integer a_{djk}. If no such decomposition is possible then an orbit must be chaotic (or the frequencies were not properly determined, which may happen if two lines are too close to become aliased) and labelled as such (this attribute does not rely on frequency diffusion rate or Lyapunov exponent and is in general a less reliable chaos detector). Furthermore, if the most prominent lines in each coordinate happen to be in resonance ($\sum_{d=1}^{N_d} r_d\omega_{d,0} = 0$ with integer r_d), the orbit is called a thin (r_1, r_2, r_3) orbit. If one of these integers is zero then the orbit is a commonly defined "resonant" orbit (e.g. an orbit with $r_1 = 2, r_2 = -1, r_3 = 0$ is a 1:2 banana in $x-y$ plane which may have some non-resonant motion in z direction).

Orbits that conserve the sign of any component of angular momentum are called tubes; they usually correpond to 1:1 resonance in the plane perpendicular to that of the conserved component, although the converse is not generally true (a weakly chaotic orbit may look like a resonance but flip the sign of angular momentum). Note that under certain conditions, more than one component of angular momentum may conserve sign, although it usually indicates that the orbit has not been integrated long enough (may happen in the vicinity of the central black hole, where the precession period is much longer than the period of radial motion).

In the near-Newtonian potential box orbits usually are replaced by pyramids, and some short-axis tubes are non-symmetric about the $x-y$ plane and are called saucers. Long-axis tubes are further divided into inner and outer subfamilies, based on the detection of "waist" near $y-z$ plane (not always robust). All in all, there are many classes of orbits, most common being `SAT`, `LAT (inner)`, `LAT (outer)`, `box`, various kinds of `thin orbit` and `resonance`, and, near the central black hole, `pyramid` and `saucer` orbits. SAT orbits are further analyzed for the resonances between radial and angular frequencies, and may be denoted as inner or outer Lindblad resonances (ILR/OLR), if the radial frequency is twice the angular frequency (depending on the sign of the latter), or corotation resonance (CR) if the angular frequency is zero (possible only in the case of rotating frame). In the rotating frame, all SAT orbits are also divided into prograde and retrograde classes, based on the sign of z component of angular momentum.

B.2 Spherical mass models

A spherical model is specified by an array of N pairs: r_i, M_i, $i = 1..N$, where $M_i \equiv M(< r_i)$ is enclosed mass, and $M_0 \equiv M(r = 0)$ is the central point mass (possibly zero). Alternatively, for Spline potential model we have pairs of r_i, Φ_i giving potential as a function of radius (excluding the central point mass). We assume that density is a power-

law function of radius inside r_1 and outside r_N, with slopes $\gamma_{\rm in}$ and $\gamma_{\rm out}$, correspondingly. Spherical models are used throughout SMILE in several flavours: (a) as a base for Spline spherical-harmonic potential approximation (in this case the input data is $r, \Phi(r)$ and up to second derivative of Φ must be continuous), (b) directly as a Spherical potential, (c) to calculate the distribution function via Eddington inversion for a model given by $r, M(r)$ – this is used to generate initial conditions for Schwarzschild model, to compute the velocity perturbations mimicking the effect of two-body relaxation, and in the standalone tool *mkspherical* (Sec. C.1).

The total mass M_∞ can be estimated by the following argument. Integrating the density from r to ∞ we get

$$M(r) = M_\infty - K\, r^{3-\gamma_{\rm out}} \quad , \quad \Phi(r) = -\frac{M_\infty}{r} + \frac{K}{2-\gamma_{\rm out}} r^{3-\gamma_{\rm out}} \tag{2}$$

with constants $M_\infty, K, \gamma_{\rm out}$ to be determined. Writing this for the last three points we obtain the relation

$$\ln\frac{M_\infty - M_{N-2}}{M_\infty - M_{N-1}}\,\ln\frac{r_{N-1}}{r_N} = \ln\frac{M_\infty - M_{N-1}}{M_\infty - M_N}\,\ln\frac{r_{N-2}}{r_{N-1}} \tag{3}$$

This becomes especially simple if $r_{N-1}^2 = r_{N-2} r_N$, in which case

$$M_\infty = \frac{M_N M_{N-2} - M_{N-1}^2}{M_N + M_{N-2} - 2M_{N-1}} \tag{4}$$

This relation is used to estimate total mass for a given density model which provides a smooth function $M(r)$, by constructing successive approximations to M_∞ at $r, 2r, 4r, \ldots$

In the more general case, it is easier to find $\gamma_{\rm out}$ first, numerically solving

$$\frac{M_N - M_{N-1}}{M_N - M_{N-2}} = \frac{r_{N-1}^{3-\gamma_{\rm out}} - r_N^{3-\gamma_{\rm out}}}{r_{N-2}^{3-\gamma_{\rm out}} - r_N^{3-\gamma_{\rm out}}} \tag{5}$$

If we have Φ_i instead of M_i, then in the above formula we replace M_i by $-\Phi_i r_i$. Then M_∞ is given by

$$M_\infty = \frac{M_N r_N^{\gamma_{\rm out}-3} - M_{N-1} r_{N-1}^{\gamma_{\rm out}-3}}{r_N^{\gamma_{\rm out}-3} - r_{N-1}^{\gamma_{\rm out}-3}} \tag{6}$$

And finally, K is obtained from (2).

The inner density profile may require more elaborate treatment. Without loss of generality we set $M_0 = 0$ (a central point mass may be added trivially). Basically we need to find the density slope $\gamma_{\rm in}$, assuming that density behaves as $\rho \propto A\, r^{-\gamma_{\rm in}}(1 - B\, r)$. Then

$$M(r) = \tilde{A}\, r^{3-\gamma_{\rm in}}(1 - \tilde{B}\, r) \quad , \quad \Phi(r) - \Phi(0) = \hat{A}\, r^{2-\gamma_{\rm in}}(1 - \hat{B}\, r) \tag{7}$$

Here tilde/hat quantities are trivially related to A, B (we do not write it down because we need only $\gamma_{\rm in}$).

$$\gamma_{\rm in} = 3 - \left[\ln\frac{M_2}{M_1} - \ln\frac{1-\tilde{B}r_2}{1-\tilde{B}r_1}\right] \Big/ \ln\frac{r_2}{r_1} \;=\; 2 - \left[\ln\frac{\Phi_2 - \Phi_0}{\Phi_1 - \Phi_0} - \ln\frac{1-\hat{B}r_2}{1-\hat{B}r_1}\right] \Big/ \ln\frac{r_2}{r_1} \tag{8}$$

If we are happy with taking into account only the leading term (setting $B=0$), then the slope is trivially obtained from the above equation. However, the slope will usually be underestimated if the radii are not too small. To get a more accurate estimate, first we compute B by solving

$$\left(\ln\frac{M_2}{M_1}-\ln\frac{1-\tilde{B}r_2}{1-\tilde{B}r_1}\right)\ln\frac{r_3}{r_1}=\left(\ln\frac{M_3}{M_1}-\ln\frac{1-\tilde{B}r_3}{1-\tilde{B}r_1}\right)\ln\frac{r_2}{r_1} \tag{9}$$

with a similar modification for Φ. Then substituting B into (8) we obtain $\gamma_{\rm in}$. This way we use 3 instead of 2 innermost points ($r_0=0$ doesn't count), but obtain generally a more accurate estimate for the slope (factor of $\gtrsim 2$ closer to the true one). This is [presently?] only used for Spline potential approximation, not for $r-M(r)$ models.

The rest of this section is devoted to the spherical model given by $r, M(r)$ pairs, which must be smooth enough to give a reasonable $f(E)$ via Eddington inversion formula. The model is initialized by fitting a cubic spline to the scaled quantities $\tilde{r}\equiv\ln r$, $\tilde{M}\equiv\ln[M(r)/(M_\infty-M(r))]$, where M_∞ has been found from (6). The endpoint derivatives are set by hand (i.e. the spline is not "natural" but "clamped") from the power-law extrapolation of density profile at small and large radii with slopes $\gamma_{\rm in}$ and $\gamma_{\rm out}$, estimated from (5) and (8) with $B=0$.

For the case $\gamma_{\rm in}=0$ an additional step is needed to accurately represent the behaviour of distribution function at origin (since it depends on $d^2\rho/d^2\Phi$ and the potential is close to parabolic, with its second derivative close to a constant). This matters only for the construction of spherical models via Eddington inversion formula, and is irrelevant for the potential approximation. If the estimated $\gamma_{\rm in}<0.1$, we assume that it is zero and instead take $\rho(r)=\rho_0(1-Pr^\alpha)$ and find the three parameters from three innermost grid points:

$$M(r)=\frac{4\pi}{3}\rho_0 r^3\left(1-\tfrac{3}{\alpha+3}Pr^\alpha\right)\ ,\quad Q_{k1}\equiv\frac{r_1^3\,M_k}{r_k^3\,M_1}=\frac{1-\frac{3}{\alpha+3}Pr_k^\alpha}{1-\frac{3}{\alpha+3}Pr_1^\alpha},\quad k=1,2,3$$

$$\alpha \text{ is found from } (1-Q_{21})(r_3^\alpha-Q_{31}r_1^\alpha)=(1-Q_{31})(r_2^\alpha-Q_{21}r_1^\alpha) \tag{10}$$

$$\text{and then } P=\tfrac{\alpha+3}{3}\,\frac{1-Q_{21}}{r_2^\alpha-Q_{21}r_1^\alpha} \tag{11}$$

In the case $\gamma_{\rm in}>0$, $\rho(r)$ is extrapolated to $r<r_1$ using simple power-law: $\rho(r)=(3-\gamma_{\rm in})M(r)/(4\pi r^3)$. The extrapolation to $r>r_N$ is equally simple: $\rho(r)=(\gamma_{\rm out}-3)Kr^{-\gamma_{\rm out}}/(4\pi)$.

The potential is evaluated at the grid points r_i by integrating $\int_0^r M(x)/x^2$, including the contribution of central point mass if present. Let $\Phi_0\equiv\Phi(0)$ be the potential at origin, if $M_0>0$ then $\Phi_0=-\infty$. The scaled potential is defined as $\tilde{\Phi}\equiv-\ln[1/\Phi_0-1/\Phi(r)]$ and it is represented as a spline function in $\tilde{r}\equiv\ln r$. Again the endpoint derivatives are evaluated from the asymptotic expressions and supplied to the clamped spline initialization.

To compute the distribution function, one needs to integrate $d^2\rho/d\Phi^2$. We represent $\rho(\Phi)$ as a spline in scaled variables $\tilde{\rho}\equiv\ln\rho$ and $\tilde{\Phi}$, computed at grid nodes $\Phi_i\equiv\Phi(r_i)$. The accurate extrapolation beyond grid is crucial. For $\Phi\to 0$ we simply substitute

$\Phi \approx -M_\infty/r$ to $\rho(r) \propto r^{-\gamma_{\rm out}}$, and for $\Phi \to \Phi_0$ a more elaborate expression is needed. In the case of a constant-density core,

$$\frac{d\rho}{d\Phi} = -\frac{3\alpha P}{4\pi}\left(\frac{3}{2\pi\rho_0}\right)^{\alpha/2-1}(\Phi-\Phi_0)^{\alpha/2-1}. \tag{12}$$

Otherwise, we find $r(\Phi)$ numerically and then substitute power-law asymptotes for $\rho(r)$ and $\Phi(r)$ as $r \to 0$.

The derivatives of $\rho(\Phi)$ are computed from the log-scaled spline as follows:

$$\frac{d\rho}{d\Phi} = \frac{d\tilde\rho}{d\tilde\Phi}\frac{\rho}{\Phi(1-\Phi/\Phi_0)}, \tag{13}$$

$$\frac{d^2\rho}{d\Phi^2} = \frac{\rho}{\Phi^2(1-\Phi/\Phi_0)^2}\left[\frac{d\tilde\rho}{d\tilde\Phi}\left(2\frac{\Phi}{\Phi_0}-1\right)+\left(\frac{d\tilde\rho}{d\tilde\Phi}\right)^2+\frac{d^2\tilde\rho}{d\tilde\Phi^2}\right] \tag{14}$$

The distribution function is computed by the Eddington inversion formula:

$$f(E) = \frac{1}{\sqrt{8}\,\pi^2}\int_E^0 \frac{d^2\rho}{d\Phi^2}\frac{d\Phi}{\sqrt{\Phi-E}} \tag{15}$$

It is evaluated at grid points Φ_i and then its logarithm is approximated by a spline in $\tilde\Phi$. This quantity may turn out to be negative at some points, in this case the point is excluded from the spline initialization (i.e. the approximated quantity will always be positive), but an error indication is given.

B.3 Spherical-harmonic and other potential expansions

BSE and Spline potentials share the representation of angular dependence of potential and density via spherical-harmonic expansion. We use the real-valued trigonometric functions instead of $\mathbf{e}^{im\phi}$ and introduce the convention that $m<0$ terms correspond to sine and $m\ge 0$ – to cosine terms. Moreover we introduce another factor of $\sqrt{2}$ in $m\neq 0$ coefficients, to make the sum of squared coefficients at a given l invariant under rotations of coordinate system. Define

$$\rho(r,\theta,\phi) = \sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l} A_{lm}(r)\,\sqrt{4\pi}\tilde P_l^m(\cos\theta)\ \mathrm{trig}\, m\phi \tag{16}$$

$$\Phi(r,\theta,\phi) = \sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l} C_{lm}(r)\,\sqrt{4\pi}\tilde P_l^m(\cos\theta)\ \mathrm{trig}\, m\phi$$

$$\mathrm{trig}\, m\phi \equiv \begin{cases} 1 & , \quad m=0 \\ \sqrt{2}\cos m\phi & , \quad m>0 \\ \sqrt{2}\sin|m|\phi & , \quad m<0 \end{cases}$$

Here $\tilde P_l^m(x) \equiv \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}\ P_l^m(x)$ are normalized associated Legendre polynomials.

For the force calculation we need the first derivatives of potential:

$$\begin{aligned}
\frac{\partial\Phi}{\partial r} &= \sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l}\frac{\partial C_{lm}(r)}{\partial r}\sqrt{4\pi}\tilde{P}_l^m(\cos\theta)\ \mathrm{trig}\, m\phi \\
\frac{\partial\Phi}{\partial\theta} &= \sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l}C_{lm}\sqrt{4\pi}\tilde{P}_l^{m\,\prime}(\cos\theta)\,(-\sin\theta)\ \mathrm{trig}\, m\phi \qquad (17)\\
\frac{\partial\Phi}{\partial\phi} &= \sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l}C_{lm}\sqrt{4\pi}\tilde{P}_l^m(\cos\theta)\ \mathrm{trig}'\, m\phi\ , \quad \mathrm{trig}'\, m\phi \equiv \begin{cases} -\sqrt{2}\, m\, \sin m\phi & , \quad m \geq 0 \\ \sqrt{2}\, m\, \cos m\phi & , \quad m < 0 \end{cases}
\end{aligned}$$

For the variation equation we need second derivatives of potential:

$$\begin{aligned}
\frac{\partial^2\Phi}{\partial\theta^2} &= \sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l}C_{lm}\sqrt{4\pi}\left(\cos\theta\,\tilde{P}_l^{m\,\prime}(\cos\theta) - \left[l(l+1) - \frac{m^2}{\sin^2\theta}\right]\tilde{P}_l^m(\cos\theta)\right)\ \mathrm{trig}\, m\phi \\
\frac{\partial^2\Phi}{\partial\theta\partial\phi} &= \sum_{l=0}^{l_{\max}}\sum_{m=-l}^{l}C_{lm}\sqrt{4\pi}\tilde{P}_l^{m\,\prime}(\cos\theta)\,(-\sin\theta)\ \mathrm{trig}'\, m\phi \qquad (18)\\
\frac{\partial^2\Phi}{\partial\phi^2} &= -m^2\,\Phi
\end{aligned}$$

(differentiation w.r.t. r is trivial substitution of $C_{lm} \to \partial C/\partial r$ in Eq. 17).

B.3.1 Basis-set potential expansion for models with infinite extent

In the BSE potential we represent the coefficients A_{lm}, C_{lm} as a weighted sum over basis functions defined in Zhao(1996) [4]:

$$A_{lm}(r) = \sum_{n=0}^{n_{\max}} A_{nlm}\,\rho_{nl}(r)\ ,\ C_{lm}(r) = \sum_{n=0}^{n_{\max}} A_{nlm}\,\Phi_{nl}(r)\ , \qquad (19)$$

$$\begin{aligned}
\Phi_{nl}(r) &= -\frac{r^l}{(1+r^{1/\alpha})^{(2l+1)\alpha}}\,G_n^w(\xi) \qquad (20)\\
\rho_{nl}(r) &= \frac{K_{nl}}{2\pi}\frac{r^{l-2+1/\alpha}}{(1+r^{1/\alpha})^{(2l+1)\alpha+2}}\,G_n^w(\xi)
\end{aligned}$$

$$K_{nl} \equiv \frac{4(n+w)^2-1}{8\alpha^2},\ w \equiv (2l+1)\alpha + 1/2\ ,\ \xi \equiv \frac{r^{1/\alpha}-1}{r^{1/\alpha}+1},$$

where $G_n^w(\xi)$ are Gegenbauer (ultraspherical) polynomials.

The derivatives of basis functions of potential are the following:

$$\begin{aligned}
\frac{\partial \Phi_{nl}}{\partial r} &= \frac{r^l}{(1+r^{1/\alpha})^{(2l+1)\alpha}} \left(G_n^w(\xi) \frac{l-(l+1)r^{1/\alpha}}{r(1+r^{1/\alpha})} + \frac{dG}{dr} \right) \qquad (21)\\
\frac{\partial^2 \Phi_{nl}}{\partial r^2} &= \frac{r^l}{(1+r^{1/\alpha})^{(2l+1)\alpha}} \left\{ -\frac{2}{r}\frac{dG}{dr} + \frac{G_n^w(\xi)}{r^2(1+r^{1/\alpha})^2} \left[(l+1)(l+2)r^{2/\alpha} \; + \right.\right.\\
&+ \left.\left. \left\{ 1 - 2l(l+1) - (2n+1)(2l+1)/\alpha - n(n+1)/\alpha^2 \right\} r^{1/\alpha} + l(l-1) \right] \right\}\\
\frac{dG}{dr} &= \frac{1}{2\alpha r} \left[-n\xi\, G_n^w(\xi) + (n+2w-1) G_{n-1}^w(\xi) \right]
\end{aligned}$$

The basis ρ_{nlm}, Φ_{nlm} (where $[*]_{nlm} \equiv [*]_{nl}\sqrt{4\pi}\,\tilde{P}_l^m(\cos\theta)\,\mathrm{trig}\, m\phi$) is biorthogonal, which means that

$$\int d\boldsymbol{r}\, \rho_{nlm}(\boldsymbol{r})\, \Phi_{n'l'm'}(\boldsymbol{r}) = I_{nl}\, \delta_{nn'}\delta_{ll'}\delta_{mm'} = \int_0^\infty dr\, 4\pi r^2\, \rho_{nl}(r)\, \Phi_{n'l'}(r)\, \delta_{mm'} \qquad (22)$$

$$I_{nl} \equiv -K_{nl} \frac{4\pi\alpha}{2^{4w}} \frac{\Gamma(2w+n)}{n!\,(n+w)\,[\Gamma(w)]^2} \qquad (23)$$

Given a certain density distribution $\rho(r,\theta,\phi)$, one may find the expansion coefficients by multiplying (16, 19) by $\Phi_{nlm}(\boldsymbol{r})$ and integrating over the entire space:

$$\begin{aligned}
A_{nlm} &= \frac{1}{I_{nl}} \int d\boldsymbol{r}\, \rho(\boldsymbol{r})\, \Phi_{nlm}(\boldsymbol{r}) \;=\; \frac{1}{I_{nl}} \int_0^\infty dr\, 4\pi r^2\, \Phi_{nl}(r)\, \langle \rho_{lm} \rangle_{\theta,\phi}(r) \;=\\
&= \frac{1}{I_{nl}} \int_{-1}^{1} d\xi\, \frac{8\pi\alpha r^3}{1-\xi^2}\, \Phi_{nl}(r)\, \langle \rho_{lm} \rangle_{\theta,\phi}(r)\;, \quad r = \left(\tfrac{1+\xi}{1-\xi} \right)^\alpha \qquad (24)\\
\langle \rho_{lm} \rangle_{\theta,\phi}(r) &\equiv \frac{1}{\sqrt{4\pi}} \int_0^\pi d\theta\, \sin\theta\, \tilde{P}_l^m(\cos\theta) \int_0^{2\pi} d\phi\, \mathrm{trig}\, m\phi\, \rho(r,\theta,\phi) \qquad (25)
\end{aligned}$$

If the density is represented by a set of point masses M_i located at positions $\boldsymbol{r}_i$, $i = 1..N$, then the coefficients are evaluated as follows:

$$\begin{aligned}
A_{nlm} &= \frac{1}{I_{nl}} \sum_{i=1}^{N} \Phi_{nl}(r_i)\, \rho_{lm,i} \qquad (26)\\
\rho_{lm,i} &\equiv M_i\, \sqrt{4\pi}\, \tilde{P}_l^m(\cos\theta_i)\; \mathrm{trig}\, m\phi_i \qquad (27)
\end{aligned}$$

B.3.2 Basis-set potential expansion for compact models

For models that have finite radius $R_{\max}$, we use spherical Bessel functions as the basis set. In the notation of the previous section,

$$\begin{aligned}
\Phi_{nl}(r) &= -\sqrt{\frac{2\alpha_{nl}}{\pi}}\, j_l(\alpha_{nl} r) \qquad (28)\\
\rho_{nl}(r) &= \frac{\alpha_{nl}^{5/2}}{4\pi} \sqrt{\frac{2}{\pi}}\, j_l(\alpha_{nl} r)\;,
\end{aligned}$$

where $j_l(x)$ are the spherical Bessel functions (e.g. $j_0(x) \equiv \sin(x)/x, j_{-1}(x) \equiv \cos(x)/x$), $\alpha_{nl} \equiv \beta_{nl}/R_{\max}$, and β_{nl} are consecutive (in n) roots of the function $j_{l-1}(x)$. For $r > R_{\max}$, the basis functions are just standard multipoles: $\Phi_{nl} \propto r^{-l-1}$. The normalization factors for the orthogonal basis are

$$I_{nl} \equiv \int_0^{R_{\max}} dr\, 4\pi r^2\, \rho_{nl}(r)\, \Phi_{nl}(r) = -\frac{\beta_{nl}^3 j_l^2(\beta_{nl})}{\pi} \tag{29}$$

The derivatives of basis functions of potential (for $r \le R_{\max}$) are the following:

$$\begin{aligned} \frac{\partial \Phi_{nl}}{\partial r} &= -\sqrt{\frac{2\alpha_{nl}}{\pi}}\, [\alpha_{nl} r j_{l-1}(\alpha_{nl} r) - (l+1) j_l(\alpha_{nl} r)]\, r^{-1} \\ \frac{\partial^2 \Phi_{nl}}{\partial r^2} &= \sqrt{\frac{2\alpha_{nl}}{\pi}}\, \left[2\alpha_{nl} r j_{l-1}(\alpha_{nl} r) + (\alpha_{nl}^2 r^2 - (l+1)(l+2)) j_l(\alpha_{nl} r)\right]\, r^{-2} \end{aligned} \tag{30}$$

B.3.3 Spherical-harmonic expansion for the scale-free potential

In this case, the density and potential are represented as

$$A_{lm}(r) = A_{lm}\, r^{-\gamma}\ ,\ C_{lm}(r) = C_{lm}\, r^{2-\gamma}\ , \tag{31}$$

The density profile $\rho(\boldsymbol{r}) = (x^2 + y^2/q^2 + z^2/p^2)^{-\gamma/2}$ is used to compute $A_{lm} = r^\gamma \langle \rho_{lm} \rangle_{\theta,\phi}(r)$ using (25). The relation between potential and density coefficients is given by

$$C_{lm} = \frac{4\pi}{(l+3-\gamma)(2-l-\gamma)}\, A_{lm} \tag{32}$$

B.3.4 Spline spherical-harmonic potential expansion

In the Spline potential the radial dependence of expansion coefficients $A_{lm}(r), C_{lm}(r)$ in (16) is represented directly as a spline function in (scaled) radius. More specifically, we define the scaled functions $\tilde{C}_{lm}$ as

$$\begin{aligned} C_{00}(r) &= -[\exp(-\tilde{C}_{00}(\xi)) - 1/C_{00}(0)]^{-1}, \quad \xi \equiv \ln r \\ C_{lm}(r) &= C_{00}(r) \tilde{C}_{lm}(\zeta), \quad \zeta \equiv \log(1+r) \end{aligned} \tag{33}$$

Here $C_{00}(0)$ is the value at origin, and $\tilde{C}_{00} \equiv -\ln[1/C_{00}(0) - 1/C_{00}(r)]$. Tilded functions are approximated by cubic splines. The evaluation of derivatives w.r.t. r is straightforward:

$$\frac{dC_{00}}{dr} = -\frac{C_{00}^2 \exp(-\tilde{C}_{00})}{r} \frac{d\tilde{C}_{00}}{d\xi} \tag{34}$$

$$\frac{d^2 C_{00}}{dr^2} = \frac{C_{00}^2 \exp(-\tilde{C}_{00})}{r^2} \left[\frac{d\tilde{C}_{00}}{d\xi} - \frac{d^2\tilde{C}_{00}}{d\xi^2} + \left(\frac{d\tilde{C}_{00}}{d\xi}\right)^2 (2C_{00}\exp(-\tilde{C}_{00}) + 1) \right]$$

$$\frac{dC_{lm}}{dr} = \frac{C_{00}}{1+r} \frac{d\tilde{C}_{lm}}{d\zeta} + \tilde{C}_{lm} \frac{dC_{00}}{dr} \tag{35}$$

$$\frac{d^2 C_{lm}}{dr^2} = \frac{C_{00}}{(1+r)^2} \left[\frac{d^2\tilde{C}_{lm}}{d\zeta^2} - \frac{d\tilde{C}_{lm}}{d\zeta} \right] + \frac{2}{1+r} \frac{d\tilde{C}_{lm}}{d\zeta} \frac{dC_{00}}{dr} + \tilde{C}_{lm} \frac{d^2 C_{00}}{dr^2}$$

Density is evaluated via Poisson equation as the second derivative of potential, rather than represented separately. The spherical-harmonic expansion of density coefficients is defined by (25): $A_{lm}(r) = \langle \rho_{lm} \rangle_{\theta,\phi}(r)$. The relation between density and potential coefficients is given by

$$C_{lm}(r) = -\frac{4\pi}{2l+1}\left[r^{-l-1} \int_0^r A_{lm}(s)\, s^{l+2}\, ds + r^l \int_r^\infty A_{lm}(s)\, s^{1-l}\, ds \right] \tag{36}$$

If the density is represented by a set of discrete points, $C_{lm}(r)$ is computed using the definition (27) as

$$C_{lm}(r) = -\frac{1}{2l+1}\left[r^{-1-l} \sum_{r_i<r} \rho_{lm,i}\, r_i^l + r^l \sum_{r_i>r} \rho_{lm,i}\, r_i^{-l-1} \right] \tag{37}$$

B.3.5 Direct evaluation of potential

It is possible to compute potential directly from Poisson equation for a given smooth density model. Let the density be represented by a Fourier expansion

$$\rho(R,z,\phi) = \sum_{m=-m_{\max}}^{m_{\max}} \rho_m(R,z) \times \begin{cases} \cos(m\phi), & m \ge 0 \\ \sin(|m|\phi), & m < 0 \end{cases} \tag{38}$$

$$\rho_m(R,z) = \frac{1}{2\pi}\int_0^{2\pi} \rho(R,z,\phi) \times \begin{cases} 1, & m = 0 \\ 2\cos(m\phi), & m \ge 0 \\ 2\sin(|m|\phi), & m < 0 \end{cases} \tag{39}$$

The potential generated by this density is also written as a sum of Fourier components:

$$\Phi(R,z,\phi) = \sum_{m=-m_{\max}}^{m_{\max}} \Phi_m(R,z) \times \begin{cases} \cos(m\phi), & m \ge 0 \\ \sin(|m|\phi), & m < 0 \end{cases} \tag{40}$$

Each term in the expansion is given by

$$\Phi_m(R,z) = -\int_{-\infty}^{+\infty} dz' \int_0^\infty dR'\, 2\pi R'\, \rho_m(R',z')\, \Xi_m(R,R',z,z')\ , \tag{41}$$

$$\Xi_m \equiv \int_0^\infty dk\, J_m(kR)\, J_m(kR')\, \exp(-k|z-z'|)\ , \quad \text{which evaluates to} \tag{42}$$

$$\Xi_m = \frac{1}{\pi\sqrt{RR'}} Q_{m-1/2}\left(\frac{R^2+R'^2+(z-z')^2}{2RR'} \right) \quad \text{if } R>0, R'>0,$$

$$\Xi_m = \frac{1}{\sqrt{R^2+R'^2+(z-z')^2}} \quad \text{if } R=0 \text{ or } R'=0, \text{ and } m=0, \text{ otherwise } 0.$$

The Legendre function Q may be expressed in terms of Gauss' hypergeometric function ${}_2F_1$:

$$Q_m(x) = \frac{\sqrt{\pi}\,\Gamma(m+1)}{(2x)^{m+1}\,\Gamma(m+3/2)}\, {}_2F_1\left(1+\frac{m}{2}, \frac{1}{2}+\frac{m}{2}; \frac{3}{2}+m; x^{-2}\right). \tag{43}$$

To speed up evaluation of Q, we pre-compute the values of ${}_2F_1$ on a suitably defined grid and use cubic interpolation to obtain its values instead of direct (expensive) computation by series summation.

For a discrete point mass set, the potential harmonics are computed as

$$\Phi_m(R,z) = -\sum_{i=1}^{N} m_i\,\Xi(R,z,R_i,z_i) \times \begin{cases} 1, & m=0 \\ 2\cos(m\phi_i), & m\geq 0 \\ 2\sin(|m|\phi_i), & m<0 \end{cases} \tag{44}$$

If the density profile is axisymmetric, then $\rho(R,z,\phi)=\rho_0(R,z)$ is the only term in the expansion, otherwise we pre-compute the Fourier terms $\rho_m(R,z)$ on a rather fine grid in R,z before computing potential, and use 2d cubic spline interpolation for obtaining the values of ρ_m used in (41); this introduces a negligible interpolation error. If the requested values of R,z are outside grid, then the corresponding Fourier term (39) is computed on-the-fly.

This direct potential is not used in orbit integration, as it does not provide forces and is very computationally expensive; instead it is used for initializing the cylindrical spline potential expansion (see next section).

B.3.6 Cylindrical spline potential expansion

For systems far from spherical symmetry (but still reasonably represented by azimuthal Fourier expansion in ϕ) the spherical-harmonic expansion is rather inefficient. Instead we may write down the potential as a Fourier expansion (40), and use 2d cubic spline interpolation to obtain the values of potential, forces and density at arbitrary point R,z,ϕ.

The two-dimensional grid in meridional plane covers a box $0\leq R\leq R_{\max}$, $-z_{\max}\leq z\leq z_{\max}$. The extrapolation beyond the box uses the quadrupole approximation of the mass distribution, with the coefficients computed from a fit to the values of potential at the outer boundary of the box. The grid is using logarithmic scaling: $\tilde{R}\equiv\ln(1+R/R_0)$, $\tilde{z}\equiv\mathrm{sgn}(z)\ln(1+|z|/R_0)$. The values of potential are also scaled using the Plummer weight function:

$$\Phi_m(R,z) = S(R,z)\,\tilde{\Phi}(\tilde{R},\tilde{z})\ , \quad S(R,z)\equiv S(r=\sqrt{R^2+z^2})\equiv 1/\sqrt{R_0^2+r^2}. \tag{45}$$

The scaling coefficient R_0 is computed from the condition that the value of potential at origin and the total mass are related by $\Phi_0(0,0)=-GM_{\rm total}/R_0$.

The values of potential at grid nodes are assigned directly from the potential of the input model (if available), using the same Fourier transform as in (39), or – if only an expression for density is given – the direct evaluation of potential (§B.3.5) is used as the intermediate stage. The latter can also be initialized from a discrete point mass set (Eq.44).

The evaluation of forces and their derivatives is a straightforward application of (45)

with the spline derivatives computed by standard expressions. We have

$$\frac{\partial\Phi_m}{dR} = \frac{R}{r}\,S'\,\tilde\Phi_m + S\,\tilde R'\,\frac{\partial\tilde\Phi_m}{\partial\tilde R}\,,\quad \tilde R' \equiv \frac{d\tilde R}{dR}\,,\quad S' \equiv \frac{dS}{dr}\,,\quad \text{same for } z, \tag{46}$$

$$\frac{\partial^2\Phi_m}{\partial R^2} = \left\{\frac{R^2}{r^2}\,S'' + \frac{z^2}{r^3}\,S'\right\}\tilde\Phi_m + \frac{2S'}{r}\,R\,\tilde R'\,\frac{\partial\tilde\Phi_m}{\partial\tilde R} + S\left\{\frac{\partial^2\tilde\Phi_m}{\partial\tilde R^2}\tilde R'^2 + \frac{\partial\tilde\Phi_m}{\partial\tilde R}\tilde R''\right\}\,, \tag{47}$$

$$\frac{\partial^2\Phi_m}{\partial R\partial z} = (rS''-S')\,\frac{Rz}{r^3}\,\tilde\Phi_m + \frac{S'}{r}\left\{z\tilde R'\,\frac{\partial\tilde\Phi_m}{\partial\tilde R} + R\tilde z'\,\frac{\partial\tilde\Phi_m}{\partial\tilde z}\right\} + \frac{\partial^2\tilde\Phi_m}{\partial\tilde R\partial\tilde z}\,S\,\tilde R'\tilde z'\,, \tag{48}$$

$$\begin{aligned}\nabla^2\Phi_m &= S\left\{\frac{\partial^2\tilde\Phi_m}{\partial\tilde R^2}\tilde R'^2 + \frac{\partial^2\tilde\Phi_m}{\partial\tilde z^2}\tilde z'^2 + \frac{\partial\tilde\Phi_m}{\partial\tilde R}\left(\tilde R'' + \tilde R'/R\right) + \frac{\partial\tilde\Phi_m}{\partial\tilde z}\tilde z'' - \frac{m^2}{R^2}\tilde\Phi_m\right\} + \\ &+ \frac{2\,S'}{r}\left\{\tilde\Phi_m + R\tilde R'\frac{\partial\tilde\Phi_m}{\partial\tilde R} + z\tilde z'\frac{\partial\tilde\Phi_m}{\partial\tilde z}\right\} + S''\,\tilde\Phi_m\,.\end{aligned} \tag{49}$$

B.3.7 Ferrers potential

The density profile of this model is given by

$$\rho(x,y,z) = \frac{105}{32\pi}\frac{M}{abc}\begin{cases}(1-m^2)^2\,,\ m\le 1\\ 0\,,\ m>1\end{cases}\quad,\quad m^2 \equiv \frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2},\ \ a>b>c. \tag{50}$$

The expressions for potential and its derivatives for the Ferrers potential (with index $m=2$) are given in the appendix of [15], and they use a set of 20 coefficients[3]. Inside the region of non-zero density, these coefficients are pre-computed at the time of initialization, but outside they need to be computed afresh at each force evaluation, which makes it rather costly; furthermore, for large radii (and also for axis ratios close to unity) the errors in calculation of elliptic functions become unacceptable. Therefore, for large r we approximate the potential by its first three spherical-harmonic coefficients:

$$\Phi(x,y,z) = -\frac{M}{r}\left(1 + \frac{(2c^2-a^2-b^2)}{36\,r^2}\frac{(2z^2-x^2-y^2)}{r^2} + \frac{(b^2-a^2)}{12\,r^2}\frac{(y^2-x^2)}{r^2}\right). \tag{51}$$

For $r > 4a$ this approximation is accurate to better than 10^{-4} for potential and forces.

B.4 Penalized spline approximation

Initialization of Spline potential from a set of point masses requires representing the radial dependence of spherical-harmonic expansion coefficients, which are computed at each particle's radius, by a small number of terms. In other words, we seek to find best-fit spline approximation to the "true" radial dependence of $C_{lm}(r)$. In addition, the coefficients calculated from a point mass set are subject to discreteness fluctuations, in other words, they represent the actual smooth density model, which is sampled by this set of particles, with some random noise which needs to be smoothed out.

[3]There is a sign error in coefficients $W_{210}, W_{021}, W_{102}$ in that paper, but not in the accompanying code.

In this section we describe the penalized linear least-square fitting method used for this purpose. Suppose we have the original data points $x_i, y_i, i = 0..N_d - 1$, and we need to find the smooth function $\hat{y}(x)$ which minimizes the following functional:

$$\sum_{i=0}^{N_d-1} \{y_i - \hat{y}(x_i)\}^2 + \lambda \int \{\hat{y}''(x)\}^2 \, dx \tag{52}$$

Here $\lambda \geq 0$ is the smoothing parameter. Standard mathematical arguments [16] show that the solution to the above equation is given by a cubic spline with knots at each data point x_i, but for the present purposes it is impractical (N_d may be as large as 10^6). Instead, we require $\hat{y}(x)$ to be a cubic spline with knots at $X_k, k = 0..N_k - 1$, number of knots being $\mathcal{O}(10)$. Denote Y_k the values of spline at its knots; a cubic spline is uniquely specified by X_k, Y_k and two additional parameters, for example the derivatives at endpoints (the standard case of natural cubic spline is when the second derivatives at endpoints are zero). Another way to represent $\hat{y}(x)$ is via b-splines, that is, $\hat{y}(x) = \sum_{p=0}^{N_b-1} w_p B_p(x)$, where $B_p(x)$ are basis functions, each of them is non-zero at most on four consecutive intervals $X_k..X_{k+1}$. The number of basis functions is $N_b = N_k + 2$.

Equation (52) is solved by the following linear system:

$$(\mathsf{A} + \lambda \mathsf{R})\mathbf{w} = \mathbf{z} \;\; , \quad \mathsf{A} \equiv \mathsf{C}^T \mathsf{C} \;\; , \;\; \mathbf{z} \equiv \mathsf{C}^T \mathbf{y} \tag{53}$$

$$\mathsf{C} \equiv C_{ip} \equiv B_p(x_i) \;\; , \;\; \mathsf{R} \equiv R_{pq} \equiv \int B_p''(x) B_q''(x) \, dx \;\; , \;\; \mathbf{w} \equiv w_p \;\; , \;\; \mathbf{y} \equiv y_i \tag{54}$$

In other words, given the x-coordinates of original data points x_i and of spline knots X_k, we construct the b-spline basis functions $B_p(x)$, calculate the matrices C_{ip}, A_{pq} and R_{pq}, and solve the equation (53) for any given set of data points y_i and smoothing factor λ.Note that the size of linear system is only $N_b \ll N_d$. An efficient way of solving the minimization problem for multiple values of $\mathbf{y}, \lambda$ is described below [17].

1. Obtain Cholesky decomposition of $\mathsf{A} = \mathsf{L}\,\mathsf{L}^T$, where L is a lower triangular matrix.

2. Obtain singular value decomposition of $\mathsf{Q} \equiv \mathsf{L}^{-1}\,\mathsf{R}\,\mathsf{L}^{-T} = \mathsf{U}\,\mathrm{diag}(S)\,\mathsf{V}^T$, where U and V are square orthogonal matrices (so that $\mathsf{U}^{-1} = \mathsf{U}^T$), and the diagonal matrix in between them holds the vector of singular values S. Since the matrix Q is symmetric positive definite, U and V are the same matrix (so in effect SVD is just eigenvalue decomposition). Next, obtain another auxiliary matrix $\mathsf{M} \equiv \mathsf{L}^{-T}\,\mathsf{U}$.

3. Now for any vector of data points $\mathbf{y}$ and value of smoothing parameter λ, the solution of (53) for weight coefficients $\mathbf{w}$ is given by first computing $\mathbf{z} = \mathsf{C}^T\mathbf{y}$ and then setting

$$\mathbf{w} = \mathsf{M}\,(\mathsf{I} + \lambda\,\mathrm{diag}(S))^{-1}\,\mathsf{M}^T\,\mathbf{z} \tag{55}$$

   In the case when $\lambda = 0$, step 2 is not necessary, as the solution is given by

$$\mathbf{w} = \mathsf{L}^{-T}\mathsf{L}^{-1}\mathbf{z} \tag{56}$$

The quality of fit is usually assessed by residual sum of squares (RSS), which is the first term in (52). In case of penalized smoothing, several modified criteria are used, for example, generalized cross-validation score (GCV), or Akaike information criterion (AIC):

$$\mathrm{GCV} \equiv \frac{\mathrm{RSS}/N_d}{(1-\mathrm{EDF}/N_d)^2} \quad , \quad \mathrm{AIC} \equiv \ln(\mathrm{RSS}) + \frac{2\,\mathrm{EDF}}{N_d - \mathrm{EDF} - 1} \quad , \quad \text{where} \tag{57}$$

$$\mathrm{RSS} = |\mathbf{y}|^2 - 2\mathbf{w}^T\mathbf{z} + |\mathsf{L}^T\mathbf{w}|^2 \quad , \quad \text{and} \;\; \mathrm{EDF} = \mathrm{tr}(\mathsf{I} + \lambda\,\mathrm{diag}(S))^{-1} = \sum_{p=0}^{N_b-1} \frac{1}{1+\lambda\,S_p}$$

is the equivalent number of degrees of freedom (varies from N_b for $\lambda = 0$ to 2 for $\lambda \to \infty$, in which case the smoothing spline is just a two-parameter linear regression).

Standard practice is to choose λ which minimizes GCV or AIC. In practice, for $N_d \gg N_b$ this results in very little smoothing, as RSS invariably grows with increasing λ and EDF/N_d changes only from a small number N_b/N_d to an even smaller one $2/N_d$. So the conventional criterion may only suppress small-scale noise (variations of y on scales much smaller than distance between knots). In the present application, we expect most of the larger-scale fluctuations in radial dependence of SH coefficients to be dominated by discreteness noise as well (in the simplest density models, all coefs typically have only one outstanding maximum in the entire range of radii). So we may need to smooth more than "optimal", which is achieved by finding a value of λ which yields the value of AIC higher than $\mathrm{AIC}(\lambda = 0)$ by a pre-defined parameter, $\Delta\mathrm{AIC}$. The justification of this approach is that if the non-smoothed regression describes the true curve reasonably well (i.e. with small RMS), then increasing AIC by a moderate constant, say 1-2, will increase RMS by a factor of few while still keeping it small. On the other hand, for the case when the data is noisy and wildly fluctuating, RMS is quite large even for non-smoothed regression, so that increasing it by a factor of few will produce a much smoother fit, probably even a linear fit (in which case the data is believed to be consistent with pure noise and discarded altogether).

B.5 Statistics of pericenter passages

In some applications, it is important to distinguish between centrophilic and centrophobic orbits. The first may approach arbitrarily close to the origin of coordinates (examples include non-resonant box orbits, pyramids and a substantial fraction of chaotic orbits), the second do not come closer than a certain distance from the center (like usual loop orbits). As part of orbit analysis, the statistics of pericenter passages is examined to determine if the orbit is centrophilic or not, or, more generally, what is the distribution of squared angular momentum values recorded at pericenter passages (defined as moments when $\dot{r} = 0$ and $\ddot{r} > 0$).

Define $L^2_{\mathrm{peri},k}, p = 1..N_{\mathrm{peri}}$ as the values of squared angular momentum recorded at pericenter passages, sorted in ascending order. It appears that for most orbits the distribution of these values is linear at small p, so we fit a linear regression

$$L^2_{\mathrm{peri},k} = L^2_{\mathrm{min}} + s \times k/N_{\mathrm{peri}} + \delta_k \;, \quad k = 1..N_{\mathrm{fit}}. \tag{58}$$

Here $N_{\rm fit}$ is typically $0.1\,N_{\rm peri}$ (but in addition we require that $10 \le N_{\rm fit} \le 50$), so we study only the low-L part of the distribution. The parameters $L^2_{\rm min}$ and s are to be estimated from the regression, and δ_k are the residuals. Additionally, we fit the same distribution to a one-parameter regression

$$L^2_{{\rm peri},k} = s' \times k/N_{\rm peri} + \delta'_k, \tag{59}$$

and compare the statistical significance of the fits, by standard χ^2 analysis. Since we do not have any intrinsic "measurement errors", we simply assign the intrinsic dispersion $\sigma^2_{\rm fit}$, same for each point, from the condition

$$\sigma^2_{\rm fit} = \frac{\sum_{k=1}^{N_{\rm fit}} \delta_k^2}{N_{\rm fit} - 2}\,, \tag{60}$$

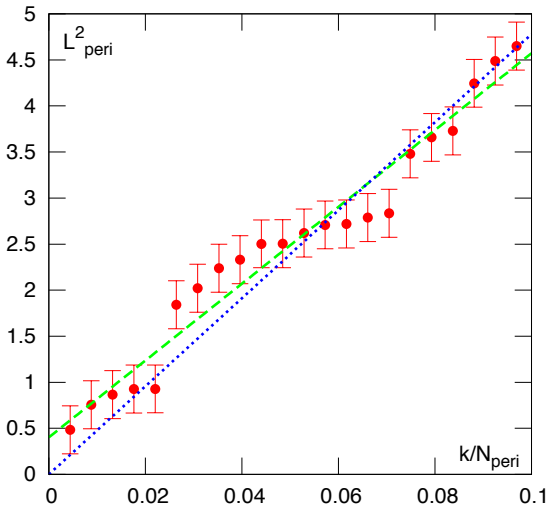
which is a standard practice giving exactly unity for χ^2 per number of degrees of freedom (d.o.f.) $N_{\rm fit} - 2$. Next, we do the same for the one-parameter regression and compute

$$\Delta\chi^2 \equiv \chi^2_{\rm one-param} - \chi^2_{\rm two-param} = \frac{1}{\sigma^2_{\rm fit}} \sum_{k=1}^{N_{\rm fit}} \delta_k'^2 - (N_{\rm fit} - 2) \tag{61}$$

Now the quantity $\Delta\chi^2$ should have a 2-d.o.f. χ^2 distribution, and we require it to be less than a certain threshold $\Delta\chi^2_{\rm thr}$ to accept the hypothesis that the one-parameter fit is good enough. In other words, we put $L^2_{\rm min} = 0$ if it happens to be < 0 in (58) or if $\Delta\chi^2 < \Delta\chi^2_{\rm thr} = 11.8$ in (61), which is a 3-σ deviation for 2-d.o.f. χ^2 distribution. Otherwise we take the best-fit value from the two-parametric regression and assign the `fitsignificance` parameter as the deviation of $L^2_{\rm min}$ from zero, measured in σ's. The *L2slope* value is assigned to either s or s' (depending on which regression is adopted). Another parameter measuring the reliability of the fit is

$$\texttt{fitscatter} \equiv \sigma_{\rm fit}/\sigma_{\rm typical}\,, \quad \text{where } \sigma_{\rm typical} \equiv \sqrt{N_{\rm fit}}\,s/N_{\rm peri} \tag{62}$$

is the "natural" scale for the magnitude of residuals. A value larger than ~ 0.5 for `fitscatter` usually indicates that the distribution of angular momenta is not well described by a linear regression.



Example of fitting the distribution of L^2 at pericenter passages by a linear regression with (dashed green, eq. 58) and without (dotted blue, eq. 59) constant term. Values of $L^2_{\rm peri}$ are sorted in ascending order, only lowest 10% of points are used in the fit; error bars are assigned from (60), with the `fitscatter` parameter (62) being $\simeq 0.3$. The fit with zero intercept ($L^2_{\rm min} = 0$) is just about 3 standard deviations worse than the two-parameter fit ($\Delta\chi^2 = 12.3$ in eq. 61). This orbit is quite likely to be centrophilic but will not be labelled as such by the 3σ criterion; a typical centrophobic orbit has a $10^2 - 10^4\,\sigma$ deviation of $L^2_{\rm min}$ from zero.

B.6 Solving the optimization problem

In general, the Schwarzschild modelling is formulated as the problem of finding weights of orbits $w_o \geq 0, o = 1..N_o$ such that N_c constraints are satisfied via $m_c = \sum_{o=1}^{N_o} t_{oc} w_o$, where t_{oc} is the contribution of o-th orbit to c-th constraint. For instance, in the Classic model, t_{oc} is the time spent by each orbit in each cell of the spatial grid. The linear system may contain the constraints from several `CSchwData` objects: density model (Classic, Cylindrical, BSE or SHGrid), kinematics (velocity anisotropy or angular momentum distribution), or, in principle, any other kind of constraints (e.g. from observations of surface brightness or line-of-sight velocity distribution), all combined in one set by the `CSchwModel` object.

In practice, not all constraints may be satisfied exactly, so there are two options to allow for the deviation from exact solution, given by the parameter `constraintPenaltyLin`. The first case, when `constraintPenaltyLin`>0, adds the penalty for constraint violation to the objective function which is minimized by the solver. This is done by introducing additional $2N_c$ non-negative variables μ_c, ν_c and rewriting the linear system as

$$m_c = \sum_{o=1}^{N_o} t_{oc} w_o + \mu_c - \nu_c \ , \ c = 1..N_c \tag{63}$$

The objective function is

$$\mathcal{F} = \alpha_1 \sum_{c=1}^{N_c} (\mu_c + \nu_c) + \alpha_2 \sum_{c=1}^{N_c} (\mu_c^2 + \nu_c^2) + \mathcal{F}_{\text{additional}} \ , \tag{64}$$

Here $\alpha_1 \equiv$ `constraintPenaltyLin`, and $\alpha_2 \equiv$ `constraintPenaltyQuad`.

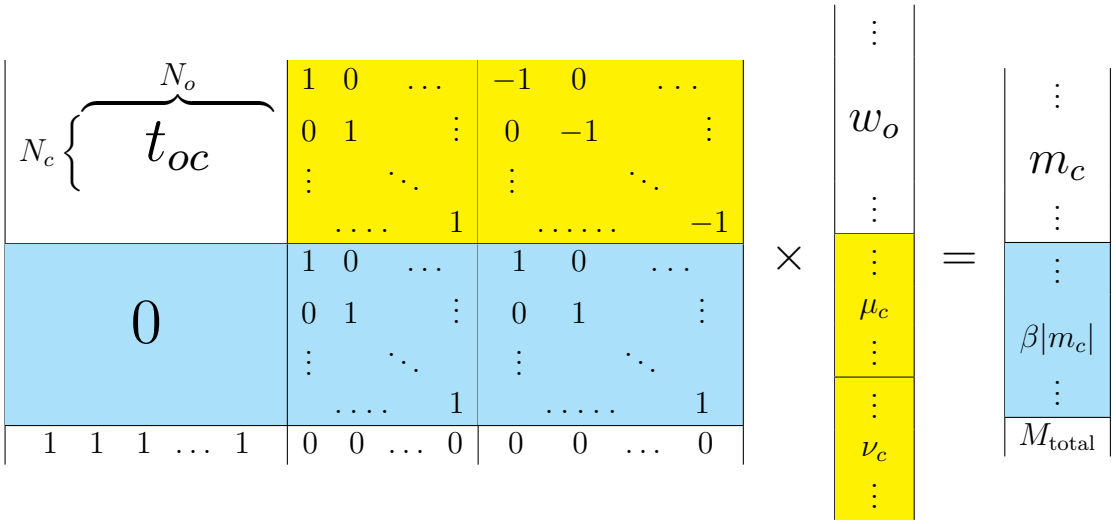
In the second case we allow the deviation in the constraint value not to exceed $\alpha_0 |m_c|$. This is achieved by adding another N_c equations to (63):

$$\mu_c + \nu_c = \alpha_0 |m_c| \ , \ \alpha_0 \equiv -\texttt{constraintPenaltyLin} \tag{65}$$

No additional terms are introduced in the objective function in this second case. It is parametrized by the fractional tolerance α_0, given by the same parameter `constraintPenaltyLin` if it is negative (its absolute value it taken).

The case when `constraintPenaltyLin=0` but `constraintPenaltyQuad`>0 is equivalent to the non-negative least-squares (NNLS) fitting method; if the former is not zero, the latter still may be used to discourage strong violation of just a few constraints in favor of moderate violation of a larger number of them.

The last equation in the linear system requires that the sum of all orbit weights is equal to the total mass of the model, and it must be satisfied exactly. All these possibilities are shown in the diagram below, which depicts the linear system to be solved. White blocks represent the original set of $N_c + 1$ equations for N_o variables (one additional for the total mass); if `constraintPenaltyLin`=0 and `constraintPenaltyQuad`=0, these are the only ones to be solved. Yellow block shows the augmented system having $2N_c$ additional variables. Cyan block shows the case with the tolerance range (it also has the additional variables and N_c more equations).



The objective function may have additional terms $\mathcal{F}_{\text{additional}}$: the quadratic regularization term is given by

$$\mathcal{F}_{\text{quadratic}} = \frac{\lambda}{N_o} \sum_{o=1}^{N_o} \left(\frac{w_o}{\tilde{w}_o} \right)^2 , \ \lambda \equiv \texttt{regularization} \tag{66}$$

Here $\tilde{w}_o$ is the prior on orbit weight; in the default case without refinement of initial conditions (regulated by `weightSkewFactorE, weightSkewFactorL` parameters) this is just the average mass per orbit M_{total}/N_o.

If one wishes to increase or reduce contribution from chaotic orbits (or, in principle, any subset of orbits based on their properties, evaluated via an instance of `COrbitFilteringFunction`), then an additional term is introduced in the objective function, given by

$$\mathcal{F}_{\text{bias}} = \frac{\mu}{M_{\text{total}}} \sum_{o=1}^{N_o} w_o \times E(\{\text{orbit}\}) \ , \ \mu \equiv \texttt{chaoticPenalty}, \tag{67}$$

$$E = \begin{cases} 1 \text{ if Lyapunov exponent } \Lambda > \Lambda_{\text{threshold}} \\ \Xi(\log_{10}(\text{FDR}/\text{FDR}_{\text{threshold}})) \text{ otherwise} \end{cases} , \ \Xi(x) \equiv \begin{cases} 0, & x < -0.5 \\ 0.5 + x, & -0.5 \le x \le 0.5 \\ 1. & x > 0.5 \end{cases}$$

The latter equation explains how the filtering function for chaotic orbits works: if the Lyapunov exponent is greater than the threshold `chaoticMinLambda`, then the evaluation function equals 1, otherwise its value is 0 for "strongly regular", 1 for "strongly chaotic" orbits and in between for the intermediate case – based on the value of frequency diffusion rate compared to the threshold `chaoticMinFreqDiff`.

The normalization factors in (64, 66, 67) are arranged so as to keep the magnitude of objective function independent of N_o and M_{total}; in addition, the values of m_c and t_{oc} entering the linear system are normalized by factors $\tilde{m}_c$ which are specific to the given variant of Schwarzschild model. This is introduced to reduce the strong variation in

magnitude of coefficients in BSE and, to a lesser extent, Spline models. For instance, in the Classic model the normalization coefficient of a cell in a given shell is the expected shell mass for a spherically-symmetric density profile, divided by the number of cells in the shell.

The linear system and the objective function are passed to the instance of linear or quadratic optimization solver, which at present may be chosen from one of three variants: BPMPD (an external program), CVXOPT (a Python-based solver) and GLPK (C library for linear programming). The interface between the Schwarzschild model object and the solver is defined in the abstract way, allowing to isolate the details of Schwarzschild modelling from the implementation of the solver (it may not even need to be a linear/quadratic one).

B.7 Multi-component models

SMILE supports multi-component potential and Schwarzschild models in the following way. The potential used for orbit integration may consist of an arbitrary number of components, each one given by its own parameters. For instance, one may have an exponential disk represented by a CylSpline potential initialized from an ExpDisk density model, a triaxial Ferrers bar, a slightly non-spherical halo represented by a Spline potential with the density coming from an Ellipsoidal model with variable axis ratio, and a central massive black hole. These three extended components reside in individual sections `[Potential], [Potential1]` and `[Potential2]` of the INI file, and the black hole mass is attached to the first of them. The Schwarzschild model could consist of two species, one for the disk plus bar, the other for the halo; the first could use the Cylindrical spatial grid, the second – SHGrid; we use JeansAxi initial conditions generator for the first one and Eddington for the second one, and additionally enforce velocity isotropy for the halo by setting `constrainBeta=true`. To specify which of the potential components constitute the target density profile for *SM*, one uses the INI parameter `[Schwarzschild_model]/densityComponents` – it should contain `0,1` for the first species and `2` for the second one. Each of the two models is constructed separately, and exported to an *N*-body file (the first one will also contain a particle corresponding to the massive black hole). Then one may stack together these two files and run an *N*-body simulation. Thus the only common requirement for the two species is that they use the same total potential; there is no way of using a density model for *SM* that is not included as part of this composite potential.

B.8 Generation of initial conditions for orbit library

Even though the Schwarzschild method is "self-tuning" in the sense that it automatically picks up the orbits that are suitable for the self-consistency of the model, its performance still depends on the method of generating the initial conditions (IC) for the orbit library. For instance, if the user wishes to create a spherical model with purely circular orbits, the method will be unable to do so unless ICs were arranged to contain only such orbits.

ICs for frequency map are chosen from one or more "start spaces" at the same value of energy. For the Schwarzschild model, we do not use the (commonly employed) scheme

of having a composition of a small number (few dozens) of discrete energy levels represented by these start spaces. Instead, a true three-dimensional density of the model is sampled as faithfully as possible, using a Monte Carlo sampler that assigns positions in proportional to the local density. Then the initial velocities are assigned using one of the available methods, described below. The choice of method is governed by the INI parameter `[Schwarzschild_model]/ICgenerator`.

B.8.1 Eddington sampler

It draws velocities from the Eddington isotropic distribution function $f(E)$ constructed for a spherical mass model (Sec. B.2) that approximates the true 3d density profile of the given mass component.

B.8.2 Spherical Jeans equation

In this method we use a spherical Jeans equation constructed for the same approximating mass model as in the previous method, but drawing velocities from a Gaussian velocity distribution function instead of the self-consistend energy distribution function. The adjustable parameter is the velocity anisotropy coefficient $\beta \equiv 1 - \sigma_t^2/(2\sigma_r^2)$, where σ_t and σ_r are the velocity dispersions in (two) tangential and (one) radial directions; $\beta = 0$ corresponds to isotropy, $\beta = 1$ – to purely radial and $\beta = -\infty$ – to purely circular orbits.

B.8.3 Axisymmetric Jeans equation

Let $\nu(R,z)$ be the axisymmetrized density of the given mass component, and $\Phi(R,z)$ be the axisymmetrized total gravitational potential. They are computed as averages over the azimuthal angle ϕ of the actual density and potential models, if the latter are not axisymmetric themselves. Following [18], we introduce the "meridional anisotropy" parameter $\beta_m = 1 - \overline{v_z^2}/\overline{v_R^2}$. Then the Jeans equations are

$$\begin{aligned} \overline{v_z^2}(R,z) &= \frac{1}{\nu(R,z)} \int_z^\infty \nu(R,z') \frac{\partial \Phi(R,z')}{\partial z'} dz', \\ \overline{v_\phi^2}(R,z) &= \left[\overline{v_z^2} + \frac{R}{\nu} \frac{\partial(\nu \overline{v_z^2})}{\partial R} \right] / (1-\beta_m) + R \frac{\partial \Phi}{\partial R}. \end{aligned} \tag{68}$$

The mean velocities in R and z directions are assumed to be zero, but $\overline{v}_\phi$ is generally non-zero and not provided by the equations. We may parametrize it by k such that $\overline{v_\phi} = k\sqrt{\overline{v_\phi^2} - \overline{v_R^2}}$. $k = 1$ corresponds to a semi-isotropic rotator (equal velocity dispersions in R and ϕ directions); $k = 0$ yields no net rotation.

B.9 Rotating frame

If the model has figure rotation with the angular velocity Ω (about z axis), the equations of motion in the $x-y$ plane are modified as follows:

$$\begin{aligned} \dot{x} &= v_x + \Omega y, \quad \dot{v}_x = -\tfrac{\partial \Phi}{\partial x} + \Omega v_y, \\ \dot{y} &= v_y - \Omega x, \quad \dot{v}_y = -\tfrac{\partial \Phi}{\partial y} - \Omega v_x. \end{aligned} \tag{69}$$

In other words, we use the coordinates in the frame that co-rotates with the figure of potential, so that the latter is a time-independent function of these coordinates, but consider the velocities in the inertial frame which coincides with the instantaneous orientation of the rotating frame. The velocities are canonically conjugate momenta to the coordinates. Instead of total energy, the Jacobi constant is the integral of motion:

$$\begin{aligned} E_J &\equiv \Phi(x,y,z) + \frac{v_x^2+v_y^2+v_z^2}{2} - \Omega(x v_y - y v_x) = \\ &= \Phi_{\rm eff}(x,y,z) + \frac{\dot{x}^2+\dot{y}^2+\dot{z}^2}{2}\,, \quad \Phi_{\rm eff}(x,y,z) \equiv \Phi(x,y,z) - \frac{\Omega^2\,(x^2+y^2)}{2} \end{aligned} \tag{70}$$

The choice between using $\boldsymbol{v}$ or $\dot{\boldsymbol{x}}$ depends on the particular task: for example, to depict the isocontours of effective potential, and locate periodic orbits in the Poincaré surface of section, we use the second expression with $\dot{\boldsymbol{x}}=0$, while the initial conditions for orbits are stored in the inertial frame, using $\boldsymbol{x}, \boldsymbol{v}$, and the reference orbital period is computed without taking into account the rotation at all (even though no orbit will have a trajectory along x axis if $\Omega \neq 0$).

The introduction of rotating frame has different meaning for different ODE integrators: IAS15 and Hermite integrators use the formulation of equations of motion in terms of $\{\boldsymbol{x}, \dot{\boldsymbol{x}}\}$, while other integrators use canonical momenta. Symplectic Runge-Kutta does not work at all, and Hermite needs two corrector steps to achieve required 4th order of approximation (at present, the forces get recomputed twice, even though the second correction is only required for the non-inertial forces).

For any finite-mass model and $\Omega \neq 0$ we locate the Lagrangian points along x and y axes, in which the first derivatives of the effective potential are zero. The value $E_\times$ of the effective potential at the saddle point along x axis marks the boundary of region inside corotation: orbits with $E_J > E_\times$ may in principle escape to infinity. As the volume of phase space is infinite for $E_J > E_\times$, it cannot be sampled by random initial conditions at given E_J.

C Additional programs

C.1 mkspherical

This program uses spherical mass models for two different purposes and created from several possible sources: (a) a supplied table with $r, M(r)$ values specifying enclosed mass as a function of radius, or (b) an N-body snapshot, in which case it first fits a

penalized spline to compute $M(r)$ from the snapshot, or (c) an arbitrary SMILE potential with parameters provided as command-line arguments, or (d) a potential expansion file (Sec. 4.4.4), or (e) an INI file with all potential parameters (Sec. 4.1). The spherical model, in turn, may be used to compute a number of parameters as spline-interpolated functions of radius (potential, radial/circular period, distribution function via Eddington inversion formula, diffusion coefficients, etc.), or to generate an N-body snapshot, in which particles are distributed according to the given density profile and isotropically in velocities. In short, this is a generalization of tools such as *halogen* or *spherICs* for creating a spherical isotropic model with a given arbitrary density profile, and at the same time a useful tool to study dynamical properties of a given N-body system (or, rather, its spherically-symmetric isotropic counterpart).

Parameters [default values]:

- `file=[]` input file, which determines the operation mode depending on its extension: (a) `.mass`, (d) `.coef_***`, (e) `.ini`, (b) anything else; mode (a) may also be forced by providing another argument `density=Coefs`. In the mode (a) this file should contain at least two columns: $r, M_{\rm enclosed}(r)$. Radii and masses must be in ascending order, nonzero value of $M(0)$ indicates the presence of central point mass (e.g. a supermassive black hole). In the mode (b), the input file is interpreted as an N-body snapshot in one of the known file formats (Sec. 4.5). In this case, the spherical model is constructed by fitting a penalized smoothing spline to the log-scaled values of $r, M(r)$. More specifically, this is done as follows:
  - particles are sorted in radius;
  - an "extrapolated total mass" is found such that the dependence of $\log[M(r)/(M_{\rm tot} - M(r))]$ vs. $\log(r)$ is best described by a linear function for the outermost few percent points. This extrapolated mass has nothing to do with the subsequent total mass in the spherical model, but this step is required in order to ensure that the log-scaled mass defined above does not exhibit sharp variations at outer radii, so that spline fitting results in least bias. For a density profile which declines as a power-law in radius, this extrapolated mass will coincide with the true extrapolated total mass, even if the profile is sharply truncated at some finite radius; otherwise this is just an internal scaling parameter.
  - a smoothing spline is fitted to the scaled quantities defined above. This will, as a side result, fit a power-law to $M(r)$ at small radii. The degree of spline smoothing is adjusted by `smoothing` parameter. The spline uses a small number of nodes ($\sim 15-50$), which is chosed automatically, but may also be controlled by the command-line parameter `Ncoefs_radial`; similarly, its radial extent may be adjusted by `splineRmin` and `splineRmax` parameters.
- `density=[]` turns on the mode (c), in which the spherical model is constructed from a given (possibly non-spherical) density profile from the list of SMILE density models. In this case, one may specify additional parameters such as `mass=...`, `scalerad=...`, etc. (see the list of INI file parameters in the `[Potential]` section).

- `mbh=[0]` is the additional point mass (representing a supermassive black hole) at the center; may be used in any mode except (e).

- `smoothing=[1]` adjusts the spline smoothness; less smoothing means somewhat better representation of wiggles in density, at the expense of distribution function wildly oscillating and possibly becoming negative. If the computed quantities seem to be weird, this is the first parameter to play with.

- `outtab=[]` name of output text file containing the following columns:
  `r` $M(r)$ Φ ρ $f(E)$ $M(E)$ $T_{\rm epi}$ $T_{\rm rad}$ $r_{\rm circ}$ $L_{\rm circ}$ $\sigma_{\rm 1d}$ $\sigma_{\rm los}$ Σ D_{EE} D_E D_{LL} $D_{\mathcal{RR}}/\mathcal{R}$
  The dynamical quantities are given as functions of radius, projected radius, or energy ($E=\Phi(r)$). $f(E)$ is the distribution function, $M(E)\equiv\int_{-\infty}^{E} f(E')g(E')dE'$ is the total mass of particles having energies lower than E, where $g(E)\equiv 4\pi^2 T_{\rm rad} L_{\rm circ}^2$ is the density of states [9, Eq.4.55]. $T_{\rm epi}$ and $T_{\rm rad}(E)$ are the epicyclic and radial periods, $r_{\rm circ}$ is the radius of circular orbit and $L_{\rm circ}$ is the angular momentum of that orbit. $\sigma_{\rm 1d}(r)$ is the conventional one-dimensional velocity dispersion, $\sigma_{\rm los}(R)$ is the line-of-sight velocity dispersion at given projected radius R, and $\Sigma(R)$ is the surface density at projected radius R. Finally, $D_{EE}(E)$ and $D_{LL}(E)$ are the diffusion coefficients in energy and angular momentum, D_E is the drift coefficient in energy, and $D_{RR}/\mathcal{R}|_{\mathcal{R}=0}$ is the limiting value of diffusion coefficient in dimensionless squared angular momentum $\mathcal{R}\equiv L^2/L_{\rm circ}^2$ at small $\mathcal{R}$, as given, for instance, in Chapter 5 of [10]. These coefficients depend not only on the mass model itself, but on the number of particles in the model. To obtain the actual values for a given single-mass N-body snapshot with particles of mass $m_\star$, multiply the values given in the table by $m_\star \ln\Lambda$, where the Coulomb logarithm $\ln\Lambda$ is usually taken to be $\sim \ln N$. The coefficients represent orbit-averaged values, and the relaxation time in energy or angular momentum may be estimated as the inverse of (D_{EE}/E^2) and $D_{\mathcal{RR}}$, correspondingly. The more familiar local relaxation time [9] is given by $T_{\rm rel}=\frac{0.34\sigma_{\rm 1d}^3(r)}{m_\star\rho(r)\ln\Lambda}$.

- `rmin=[]`, `rmax=[]` inner and outer radii for the output table; as usual, a log-scaled grid is created between these radii and the quantities are output at the nodes of this grid. Default values are chosen to enclose all interesting features in the model.

- `npoints=[100]` number of output grid points. The output grid doesn't need to be related to the input $r, M(r)$ values from the text file, in particular, it is instructive to see how the model is extrapolated to smaller and larger radii (sometimes, however, is does weird things). If `npoints=-1` then the output grid is the same as input set of radial points (or the nodes of interpolation spline if the input is an N-body snapshot).

- `outsnap=[]`, `nbody=[]` – if given, an N-body representation of the spherical model is created and written as a snapshot file.

- `outformat=[Text]` – format of the output snapshot file (Text/Nemo/Gadget, may be abbreviated to one letter).

- `quiet=[0]` – specifies how random are particle positions in the output snapshot: 0 – totally random, 1 – random but each one within its own interval of equally-spaced $r(M)$, 2 – fixed at equal intervals of $r(M)$.

C.2 renderdensity

Create a visual representation of an arbitrary SMILE density or potential model, by populating the space with point masses in proportion to local density. No velocities are assigned to the particles. The input density model may be specified in one of the following modes: (a) as the analytic density profile, (b) as a potential expansion constructed from an analytic density profile, (c) as a potential coefficients file, or (d) as parameters provided in an INI file.

Parameters:

- `density=[]` – name of the density model in the modes (a) and (b); additional parameters are the same as for `mkspherical` (see Sec. 4.1).
- `type=[]` – type of the potential expansion to be constructed from the density model in mode (b), optional: if not given, the density model is used directly as mode (a). It may be instructive to render the same density profile twice – as the original density model, and as the approximation from the potential expansion, to see the possible artifacts from the approximation.
- `file=[]` – input potential coefs file in mode (c), see Sec. 4.4.4, or an INI file with all required parameters in mode (d); the choice depends on the file extension.
- `out=[required]` – output snapshot file.
- `nbody=[required]` – number of particles to represent the density profile.
- `outformat=[Text]` – format of the snapshot file (Text/Nemo/Gadget).

C.3 testaccuracy

This program checks the accuracy of potential approximation constructed from an N-body snapshot or from an analytic density profile, using one of the potential expansions available in SMILE (BSE, Spline, CylSpline, etc.). It compares the density from the original profile and from the approximation, by constructing a spatial grid (same as in Classic or Cylindrical Schwarzschild models) and computing mass of grid cells from both density models:

- If the original model was an N-body snapshot, then the masses of particles residing in each grid cell are summed and compared against the integrated density from the potential expansion. Due to finite-N fluctuations, the two numbers are not expected to agree exactly, but the relative difference should be of order $N^{-1/2}_{\text{points in cell}}$ if the approximation is faithful. In this case, the χ^2-like quantity is computed and reported as the measure of quality of the potential approximation, and it should be around unity for it to be good.

- If the original model is given by a smooth density profile, then the cell masses obtained by integrating the original and approximating models are compared; the smaller the difference, the better. Reported is the mean integrated square error (MISE) $M_{\rm total}^{-1} \int d^3r\, \rho_{\rm exact}(\boldsymbol{r})\, [1 - \rho_{\rm approx}(\boldsymbol{r})/\rho_{\rm exact}(\boldsymbol{r})]^2$.

Parameters:

- `type=[required]` – type of potential expansion (BSE, BSECompact, Spline, CylSpline).
- `file=[]` – name of input N-body snapshot file in one of recognized formats (Sec. 4.5), or
- `density=[]` – name of the smooth density profile, along with other optional parameters (same as in the `[Potential]` section of INI file, Sec. 4.1)
- `out=[]` – optional name of output files: one file stores the expansion coefficients and has the appropriate extension (`.coef_***`) depending on the potential type, another file with the extension `.grid` contains the coordinates of the center of each cell and their masses computed using two methods. If the input file is given, its name serves as the base of the output files.
- `cylGrid=[false]` – use Classic (false) or Cylindrical (true) grid.
- `numRadialCoefs=[20], numVerticalCoefs=[15], numAngularCoefs=[]` – grid dimensions; default value for the latter parameter depends on the order of angular expansion.

C.4 snaporbits

Perform orbit analysis for a N-body simulation from the input file contains multiple time snapshots, output OrbitLibrary file (Sec. 4.2) with orbit properties (i.e. trajectories of individual particles from the snapshot are analyzed). Useful to compute orbit population and/or proportion of centrophilic orbits; chaotic properties are not very meaningful. To analyze what kind of orbits are possible in a smoothed potential of an N-body model, it is better to use the entire SMILE machinery.

Parameters:

- `in=[required]` – input file in NEMO format, containing multiple (no less than several hundred) time snapshots of an N-body system in evolution, equally spaced in time. The test particles are traced from one snapshot to the other assuming that their indices are unchanged, so each particle produces an orbit.
- `out=[required]` – output OrbitLibrary file.
- `file=[]` – file with potential coefficients, or INI file with potential parameters. If no file is given then the first snapshot from the input file is taken to create a potential model. The potential is only used to compute $T_{\rm orb}$, so it needs not be a very accurate representation.

- `ncoefs_radial=[20], ncoefs_angular=[6], symmetry=[t], type=[Spline]` – parameters for potential approximation if it is created from the first N-body snapshot and not read from a coefs of INI file: number of radial and angular terms, symmetry type ([n]one, [r]eflection, [t]riaxial, [a]xisymmetric, [s]pherical), and the potential approximation type (BSE/BSECompact/Spline/CylSpline).

- `mbh=[0]` – additional point mass (black hole) at the origin (for all variants of potential specification this additional mass needs to be provided separately, since coefs file does not contain information about $M_{\rm bh}$, and input snapshot for creating a potential approximation should not contain it either).

C.5 measureshape

Computes axis ratios and orientation of principal axes of an N-body snapshot, as functions of radius. Use either equidensity ellipsoid axis ratios (if density is decreasing function of radius), or moment of inertia tensor. (Doesn't use SMILE).
The snapshot file may contain multiple time moments; presently only NEMO snapshots are processed.

Parameters:

- `in=[required]` – input snapshot file;

- `center=[t]` – whether to center the input snapshot on the median value of each coordinate;

- `out=[]` – if provided, output centered and rotated snapshot to this file;

- `nbins=[20]` – number of bins in radius for which to compute the axes;

- `cutoff=[0.999]` – fraction of total mass contained in the outermost bin;

- `minbin=[0]` – fraction of mass in the innermost bin; 0 by default means 1/nbins, i.e. all bins contain equal mass, otherwise bin masses are spaced quasi-exponentially in enclosed mass;

- `cumul=[t]` – each bin contains all mass interior to given radius [t] or only the mass in the (ellipsoidal) shell between the given and the previous radius [f]; the latter option should be used with care as it may not always converge (e.g. if the ellipsoid axes are varying rapidly with radius);

- `dens=[f]` – method of computing the axis ratio:

  - Equidensity ellipsoid [t], provided that density is a decreasing function of radius and that density information is present in the snapshot (used in [19]).
  - Inertia tensor [f] of particles with iteratively determined axes (see [20] for an extended discussion on variations of this method). The axes are determined from the inertia tensor of particles within ellipsoidal volume, with the ellipsoid itself using the axes determined on the previous iteration, until it converges.

- `compact=[f]` – print all data for given time in one line [t] or in nbins lines [f];
- `angles=[f]` – print the angles between x and z axes and the major and minor axes of the ellipsoid).

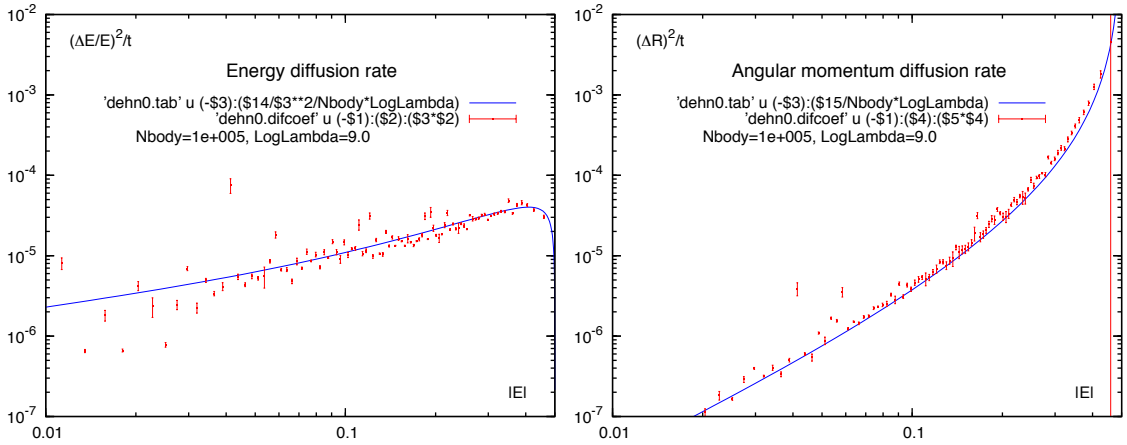
C.6 energydiff

Computes the rate of energy and angular momentum diffusion in an N-body simulation of a near-equilibrium system. Useful to check that the system under study is indeed in equilibrium, and that all changes in particles' energies are due to two-body relaxation. All particles are assigned to `nbins` bins, according to their initial energy. The program tracks the squared change of total energy and angular momentum for each particle in a simulation containing multiple time snapshots, and averages them over particles belonging to the same energy bin. For a diffusion process, the square change in the integral of motion should grow linearly with time, with the slope being the diffusion coefficient. Accordingly, the best-fit slopes for energy and angular momentum are reported as functions of energy.

Parameters:

- `in=[required]` – input NEMO snapshot file.
- `nbins=[100]` – number of bins in energy.
- `minbin=[0]` – fraction of mass in the innermost bin (0 by default means 1/nbins).
- `rel=[f]` – if false, report the diffusion coefficients for $\langle \Delta E^2/t \rangle$ and $\langle \Delta L^2/t \rangle$, if true, report the diffusion coefficients for relative (dimensionless) quantities $\langle (\Delta E/E)^2/t \rangle$ and $\langle \Delta \mathcal{R}^2/t \rangle$, where $\mathcal{R} \equiv L^2/L^2_{\rm circ}$ is the squared angular momentum normalized to that of a circular orbit with the same energy. To compute the latter quantity, the program uses a spherical mass model (Sec. B.2) approximating the N-body system. It could be constructed from either the first snapshot of the simulation (default), or specified by the $r, M(r)$ file (same as input file for `mkspherical` program, Sec. C.1).
- `tab=[]` may provide this mass model file for computing $L^2_{\rm circ}(E)$. If not given, the first snapshot is used. There is no need in smoothing the input snapshot as done in `mkspherical`.
- `mbh=[0]` – additional point mass at the center (in general, if a simulation contains a black hole, it should be filtered out, e.g. by the NEMO tool `snapmask`, before computing the diffusion rate, otherwise the data for inner bins will be distorted. This parameter is only relevant for computing $L_{\rm circ}$ in the case `rel=t`).
- `outdelta=[$in.delta]` – output file containing $\langle \Delta E^2 \rangle$, $\langle \Delta L^2 \rangle$ (or their dimensionless counterparts in the case `rel=t`) for each bin and each time. It contains `2*nbins+1` columns, the first is the snapshot time, and then two numbers for each energy bin. Each row corresponds to one time snapshot. The first line lists the average and maximal value of energy for each bin. This file may be used to check that the squared changes in E and L indeed grow linearly with time.

- `outcoef=[$in.difcoef]` – output file containing summary information, i.e. fitted diffusion coefficients for each energy bin and their uncertainties. Each line has 5 values: average (initial) energy of particles in each bin, $\langle \Delta E^2/t \rangle$ and its relative uncertainty, and the same for L (or analogous diffusion coefficients for dimensionless values). These coefficients may be compared to the predictions of two-body relaxation theory, by using a tab file from `mkspherical` (Sec. C.1) for the same mass model. The figure below shows a comparison for a $N = 10^5$ particle spherical $\gamma = 0$ Dehnen model, plotted in `gnuplot`.



References

[1] Vasiliev E., 2013, MNRAS, 434, 3174

[2] Vasiliev E., Athanassoula E., 2015, submitted

[3] Hernquist L., Ostriker J., 1992, ApJ, 386, 375

[4] Zhao H.-S., 1996, MNRAS, 278, 488

[5] Allen A., Palmer P., Papaloizou J., 1990, MNRAS, 242, 576

[6] Hairer E., Nørsett S., Wanner G., 1993, *Solving ordinary differential equations*, Springer-Verlag

[7] Rein, H., Spiegel, D., 2015, MNRAS, 446, 1424

[8] Ahnert, K., Mulansky, M. 2011, AIP Conf. Proc. 1389, 1586

[9] Binney J., Tremaine S., *Galactic Dynamics*, 2008, Princeton Univ. press

[10] Merritt D., *Dynamics and evolution of galactic nuclei*, 2013, Princeton univ. press

[11] Hut P., Makino J., McMillan S., 1995, ApJL, 443, L93

[12] Hunter C., 2002, SSRv, 102, 83

[13] Carpintero D., Aguilar L., 1998, MNRAS, 298, 1

[14] Pfenniger D., 1984a, A&A, 141, 171

[15] Pfenniger D., 1984b, A&A, 134, 373

[16] Green P., Silverman B., 1994, *Nonparametric regression and generalized linear models*, Chapman&Hall, London

[17] Krivobokova T., 2006, *Theoretical and practical aspects of penalized spline smoothing*, PhD thesis, Univ. Bielefeld

[18] Cappellari M., 2008, MNRAS, 390, 71

[19] Athanassoula E., Misiriotis A., 2002, MNRAS, 330, 35

[20] Zemp M., Gnedin O., Gnedin N., Kravtsov A., 2011, ApJS, 197, 30