SMILE reference

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

Version 2.0β_1
July 25, 2013

Contents

1 Introduction

SMILE[1] is a software for orbital analysis and Schwarzschild modelling. The scientific reference paper is [1]; 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, 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

---

[1] Schwarzschild Modelling Interactive expLoratory Environment

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.

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 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 types are implemented, having different subsets of parameters.

- Logarithmic: $\Phi(\tilde r) = \ln(R_c^2 + \tilde r^2)$;
- Anisotropic harmonic oscillator: $\Phi(\tilde r) = \tilde r^2$;
- Dehnen: $\rho(\tilde r) = \frac{3-\gamma}{4\pi pq} \tilde r^{-\gamma}(1+\tilde r)^{-(4-\gamma)}$;
- Scale-free: $\rho(\tilde r) = \tilde r^{-\gamma}$;
- Basis-set expansion (BSE, see below);
- Spline expansion (see below)
- Frozen-N-body (see below).

Here $\tilde r = (x^2 + y^2/q^2 + z^2/p^2)^{1/2}$ is the elliptical 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_{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_c (core radius) – in log.potential denotes region of constant-density core, may be zero.
- N_{radial} and N_{angular} – order of basis-set and spline expansions.
- 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, [2]) and Spline are two general-purpose potential expansions which may be used to approximate almost any potential model. 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:

- Dehnen (same as above);
- Plummer: $\rho(\tilde r) = \frac{3}{4\pi pq}(1+\tilde r^2)^{-5/2}$;
- Perfect ellipsoid: $\rho(\tilde r) = \frac{1}{\pi^2 pq}(1+\tilde r^2)^{-2}$;
- Isochrone: $\rho(\tilde r) = \frac{3}{4\pi pq}\frac{3(1+a)a^2-(1+3a)\tilde r^2}{a^3(1+a)^3}$, $a \equiv \sqrt{1+\tilde r^2}$;
- 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. $\rho(\tilde r) = C\, r^{-1}\,(1+r)^{-2}\,(1+r/r_{\text{cut}})^{-1}$ with r_{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). The total mass is then normalized to unity. r_{cut} is somewhat (but not much) smaller than c because the cutoff is smoother, which is crucial for an efficient potential expansion.
- 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 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, the parameter α is controlling the shape of radial basis functions in the Zhao(1996) basis set [3]; 0 means auto-detect, values between 1 and 2 are reasonable in most cases. 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_spl` 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 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 BSE/Spline and N-body potentials, 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.

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 (switch radiobuttons at left). The latter case is primarily used in construction of frequency map, while the former is for studying individual orbits. T_{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 Dehnen and BSE/Spline/Nbody $E < 0$, for log it may be arbitrary.

Calc 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. Slows down computation approximately twice. Not applicable for frozen-N-body potential (would give positive values anyway).

Integration time: given in units of $T_{\rm orb}$; **steps per orbit** affect basically only orbit rendering, but if set too low, it may hinder to find higher-frequency spectral lines, so keep it at least ≥ 10.

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. 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), fractional conservation of energy (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**);
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);

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 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.

Each orbit integration adds a series of points with a new color to the plot (a point of x, v_x coordinates corresponds to the passage of y axis with $v_y > 0$). 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 (where $v_y = 0$).

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 (x and v_x are taken from the plot, $y = 0$, and v_y is calculated from the given energy). (To integrate the orbit, press Enter thereafter).

Changing the energy clears the plot (as does the eponimous button). One may also export its content to a text file.

2.2.3 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 precise Hunter method, black – by 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 nearby lines). Lines may be turned off by checkbox. Left mousebutton zooms, middle button moves, right click zooms out.

2.2.4 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).

2.2.5 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).
Or, 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.

2.2.6 Schwarzschild model

Orbit library tab specifies how many orbits are in the model (with randomly assigned initial conditions over the entire range of energies). Number of **sampling points** specifies how many points drawn randomly from each orbit trajectory will be stored in `.sam` 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 **use existing start-space** is unchecked) and begins orbit integration in parallel threads.

Create *N*-body initial conditions 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.

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 shells** in radial direction, number of **lines** splitting each of three segments of each shell (so the number of cells in a shell is $3n_{\text{lines}}^2$, and total number of constraints is $3\,n_{\text{lines}}^2\,n_{\text{shells}}$).
- **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** ($= l_{\max}$, so that the number of terms in each shell is $(l_{\max}/2 + 1)(l_{\max}/2 + 2)/2$ since only terms with even l are used).
- **BSE**: evaluating coefficients of BSE expansion of every orbit. Parameters: number of angular coefs (same as above), radial coefs (equivalent to n_{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 variant chosen, 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. For the first two methods, this radial grid coincides with the partitioning of configuration space for the model itself; for BSE it is not related to the number of radial BSE coefficients. Using anisotropy profile adds $n_{\rm shells}$ constraints to the optimization problem.

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.

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 of zero; thus some orbits are deliberately assigned to lie (mostly) beyond the grid, thus ensuring that the total model mass could be more than $M_{\rm outer}$.

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

Constraint penalty is the contribution to the objective function penalizing the constraint violation. If it is zero, the constraints must be satisfied exactly (not recommended). If it is positive, this is the factor α in the penalty function (47). If it is negative, its absolute value gives the relative tolerance range for the constraints β (48). *Note* that in the second case 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 (49), which drives the orbits towards a more uniform weight distribution.

Chaotic penalty is the factor κ in (50) 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 β_{in} in the first shell to β_{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 *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, 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 two 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 automatically 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.

`BuildFreqMap` creates an orbit library with the number of points specified in the `Frequency_map` section of INI file, for the energy 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 `ExportOrbitsSchw` command.

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

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` (and their default values).

`[Potential]` – potential properties and integration accuracy.

`Type` – variants: Logarithmic (default), Harmonic, Dehnen, Scale-free, Scale-free SH, BSE, Spline, Nbody.

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

`DensityModel` – for BSE/Spline potentials, this is the underlying density model used to compute coefficients. Variants: Dehnen, Plummer, Perfect Ellipsoid, Isochrone, NFW, 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` – for treecode N-body potential this is the name of file with set of point masses, for BSE/Spline potentials it may be any of the above described three kinds of text files.

`N_dim (3)` – 2 or 3.

`q_YtoX, p_ZtoX` – y/x and z/x axis ratio, must be $p \le q \le 1$.

`Mbh (0)` – central point mass (supermassive black hole).

`Rc (0)` – core radius of logarithmic potential, *or* concentration parameter (effectively cutoff radius) for NFW potential.

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

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

`Ncoefs_radial, Ncoefs_angular (10, 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 $l_{\rm max}$ is even; 0 means spherical model only).

`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.

`accuracyRelative, accuracyAbsolute (1e-10)` – accuracy parameters for the 8th order Runge-Kutta integrator (all potentials except N-body).

`accuracyTreeCode (0.25)` – η, factor in timestep selection for the leap-frog 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) [6] algorithm for time-symmetrizing adaptive timestep, to improve energy conservation at the expense of almost twice as slower integration.

`splineSmoothFactor (1)` – value of ΔAIC (40) determining the amount of smoothing of spline coefficients initialized from an N-body snapshot.

`splineRadiusMin, splineRadiusMax (0)` – determines the extent of the radial grid in spline potential; 0 means compute by default.

`[Orbit]` – single orbit integration properties.

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

`E` – IC energy.

`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).

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

`stepsPerPeriod (50)` – output timesteps per T_{orb}: determines the maximum orbit frequency which can be detected, and the smoothness of rendered orbit. Has nothing to do with integration accuracy.

`calcLyapunov (false)` – whether to compute Lyapunov exponent.

`calcLyapunovMethod (1)` – the choice of method for computing Lyapunov exponent: 1 – integration of a nearby orbit, 2 – variational equation. The latter option is typically slower but potentially more reliable.

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

`intTimeMax (0)` – maximum integration time (in T_{orb}) if Pfenniger's adaptive method [9] is used (only in the context of Schwarzschild modelling; 0 to disable). [currently unused].

`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, BSE, SHGrid.

`numOrbitsRandom (10000)` – number of orbits in the entire model.

`numShells (25)` – number of radial shells (used to store velocity dispersion information in all variants of *SM*, and to partition configuration space in Classic SM).

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

`numRadialCoefs (25)` – in BSE and SHGrid *SM*, number of radial coefs or radial grid points.

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

`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.

`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 $\Lambda = 0$).

`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 (0), betaOut (0.5)` – values of β for the inner and the outer radial shells (linearly interpolated in enclosed mass between these two values).

`constraintPenalty (1)` – α (positive) or β (negative) factors in the optimization problem (see Sec. 2.2.6 for the description).

`regularization (0.01)` – λ factor (49) which drives the distribution of orbit weights towards a more uniform one.

`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.

`numSamplingPoints (1000)` – number of sampling points from each orbit that are stored in `.sam` file. They are drawn randomly from the trajectory after orbit integration is finished, avoiding duplicates if possible (i.e. if total number of points in trajectory, `intTime*intTimeStep`, is larger than `numSamplingPoints`). The sampling points are used in creating N-body model from Schwarzschild model, so if this feature is not used, one may set this number to zero.

`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).

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

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

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

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 timeunit timestep inttime maxtime ...
     ... 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.

`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); `timestep` is the timestep of trajectory output.

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

`energy` is (initial) total orbit energy 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 square roots of diagonal components of inertia tensor, 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 (only in the GUI version), 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 an array of 20×6 floats with the name `BSE`, plus 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). 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_{points} is written (4-byte integer), then the array of $6 \times n_{\mathrm{points}}$ 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 shortened version of orbitlib file (Sec. 4.2).

4.4.2 Ellipsoidal model file

The Ellipsoidal mass model, which may be used as an input to BSE or, preferrably, 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.

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

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

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/SHEcoefs` – specifies potential type (BSE or Spline);
`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;
`alpha` – BSE parameter (unused in Spline);
`time` – unused;
`#commented out` line (text header for the table below).
The rest of file is the coefficients table, 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 $1 + (n_{\rm angular} + 1)^2$, 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).

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;
`decomp` is the decomposition of index into readable numbers: for the Classic model, 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, but 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.

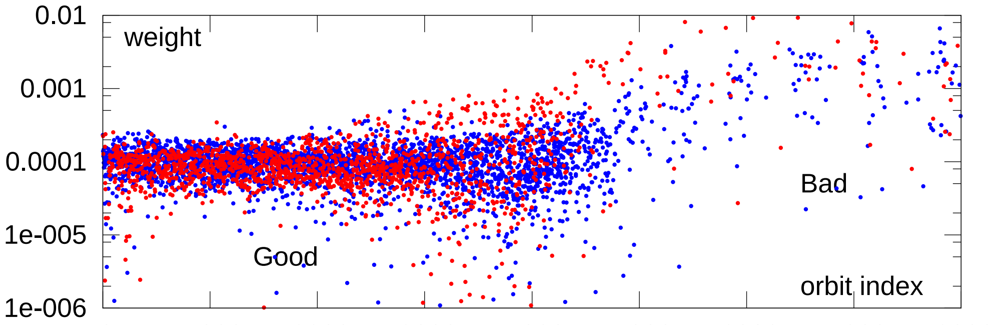
5 A short guide to practical Schwarzschild modelling

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 the number of radial coefficients, 20 is a good choice in most cases; 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. 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).

- 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. None of the potential expansion methods are expected to perform well for highly flattened models. 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 $\gamma \in [0, 2)$ (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.

- *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.

Ideas waiting to be implemented:

- Observationally-driven Schwarzschild modelling;
- Full support for figure rotation in potentials;
- 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), (*)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); at least one solver should be present to perform modelling (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 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).

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* and additional programs described in Sec. C; you may try it.

---

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

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 the functionality. More information on the internal structore 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` Runge-Kutta integrator, 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.

B Technical details on the algorithms and formulae used

B.1 Frequency analysis

Orbit classification requires 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) [7] with Hanning window filtering; if not a more approximate method of Carpintero&Aguilar(1998) [8] 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). 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. $r_1 = -r_2 = 1, r_3 = 0$ corresponds to 1:1 z-tube, although to classify an orbit as a genuine tube we furthermore require that the sign of angular momentum component about this axis is conserved).

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 two 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), or to compute distribution function via Eddington inversion for a model given by $r, M(r)$ – this is used to generate initial conditions for Schwarzschild model, 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 $r, 2r, 4r, \dots$

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:

$$\begin{aligned}
\frac{d\rho}{d\Phi} &= \frac{d\tilde\rho}{d\tilde\Phi} \frac{\rho}{\Phi(1-\Phi/\Phi_0)}, &(13)\\
\frac{d^2\rho}{d\Phi^2} &= \frac{\rho}{\Phi^2(1-\Phi/\Phi_0)^2} \left[\frac{d\tilde\rho}{d\tilde\Phi} \left(2\frac{\Phi}{\Phi_0} - 1 \right) + \left(\frac{d\tilde\rho}{d\tilde\Phi} \right)^2 + \frac{d^2\tilde\rho}{d\tilde\Phi^2} \right] &(14)
\end{aligned}$$

The distribution function is computed by the Eddington inversion formula:

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

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 expansion

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 $m \geq 0$ – 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

$$\begin{aligned}
\rho(r,\theta,\phi) &= \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} A_{lm}(r)\, \sqrt{4\pi} \tilde P_l^m(\cos\theta)\ \mathrm{trig}\, m\phi &(16)\\
\Phi(r,\theta,\phi) &= \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} C_{lm}(r)\, \sqrt{4\pi} \tilde P_l^m(\cos\theta)\ \mathrm{trig}\, m\phi \\
\mathrm{trig}\, m\phi &\equiv \begin{cases} 1 &,\ m=0 \\ \sqrt{2}\cos m\phi &,\ m>0 \\ \sqrt{2}\sin |m|\phi &,\ m<0 \end{cases}
\end{aligned}$$

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:

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

For the variation equation we need second derivatives of potential:

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

(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

In the BSE potential we represent the coefficients A_{lm}, C_{lm} as a weighted sum over basis functions defined in Zhao(1996) [3]:

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

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

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

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 Spherical-harmonic expansion for the scale-free potential

In this case, the density and potential are represented as

$$A_{lm}(r) = A_{lm}\, r^{-\gamma}\;,\; C_{lm}(r) = C_{lm}\, r^{2-\gamma}\;, \qquad (28)$$

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

$$C_{lm} = \frac{4\pi}{(l+3-\gamma)(2-l-\gamma)}\, A_{lm} \qquad (29)$$

B.3.3 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

$$\begin{aligned} C_{00}(r) &= -[\exp(-\tilde{C}_{00}(\xi)) - 1/C_{00}(0)]^{-1}, \quad \xi \equiv \ln r \qquad (30) \\ C_{lm}(r) &= C_{00}(r)\tilde{C}_{lm}(\zeta), \quad \zeta \equiv \log(1+r) \end{aligned}$$

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:

$$\begin{aligned}
\frac{dC_{00}}{dr} &= -\frac{C_{00}^2 \exp(-\tilde{C}_{00})}{r} \frac{d\tilde{C}_{00}}{d\xi} \qquad (31) \\
\frac{d^2C_{00}}{dr^2} &= \frac{C_{00}^2 \exp(-\tilde{C}_{00})}{r^2} \left[\frac{d\tilde{C}_{00}}{d\xi} - \frac{d^2\tilde{C}_{00}}{d\xi^2} + \left(\frac{d\tilde{C}_{00}}{d\xi} \right)^2 (2C_{00} \exp(-\tilde{C}_{00}) + 1) \right] \\
\frac{dC_{lm}}{dr} &= \frac{C_{00}}{1+r} \frac{d\tilde{C}_{lm}}{d\zeta} + \tilde{C}_{lm} \frac{dC_{00}}{dr} \qquad (32) \\
\frac{d^2C_{lm}}{dr^2} &= \frac{C_{00}}{(1+r)^2} \left[\frac{d^2\tilde{C}_{lm}}{d\zeta^2} - \frac{d\tilde{C}_{lm}}{d\zeta} \right] + \frac{2}{1+r} \frac{d\tilde{C}_{lm}}{d\zeta} \frac{dC_{00}}{dr} + \tilde{C}_{lm} \frac{d^2C_{00}}{dr^2}
\end{aligned}$$

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

$$C_{lm}(r) = -\frac{4\pi}{2l+1} \left[r^{-l-1} \int_0^r A_{lm}(s)\, s^{l+2}\, ds + r^l \int_r^\infty A_{lm}(s)\, s^{1-l}\, ds \right] \qquad (33)$$

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

$$C_{lm}(r) = -\frac{1}{2l+1} \left[r^{-1-l} \sum_{r_i<r} \rho_{lm,i}\, r_i^l + r^l \sum_{r_i>r} \rho_{lm,i}\, r_i^{-l-1} \right] \qquad (34)$$

For convenience, values of $C_{lm}(r)$ in the coefficients file (Sec. 4.4.4) are stored with inverted sign, so that $C_{00}(r)$ is positive.

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.

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:

$$\sum_{i=0}^{N_d-1} \{y_i - \hat{y}(x_i)\}^2 + \lambda \int \{\hat{y}''(x)\}^2\, dx \qquad (35)$$

Here $\lambda \geq 0$ is the smoothing parameter. Standard mathematical arguments [10] 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 (35) is solved by the following linear system:

$$(\mathsf{A} + \lambda \mathsf{R})\mathbf{w} = \mathbf{z} \quad , \quad \mathsf{A} \equiv \mathsf{C}^T\,\mathsf{C} \quad , \quad \mathbf{z} \equiv \mathsf{C}^T\mathbf{y} \tag{36}$$

$$\mathsf{C} \equiv C_{ip} \equiv B_p(x_i) \quad , \quad \mathsf{R} \equiv R_{pq} \equiv \int B''_p(x) B''_q(x)\, dx \quad , \quad \mathbf{w} \equiv w_p \quad , \quad \mathbf{y} \equiv y_i \tag{37}$$

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 (36) 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 [11].

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 (36) for weight coefficients $\mathbf{w}$ is given by first computing $\mathbf{z} = \mathsf{C}^T\mathbf{y}$ and then setting
$$\mathbf{w} = \mathsf{M}\,(\mathsf{I} + \lambda\,\mathrm{diag}(S))^{-1}\,\mathsf{M}^T\,\mathbf{z} \tag{38}$$
In the case when $\lambda = 0$, step 2 is not necessary, as the solution is given by
$$\mathbf{w} = \mathsf{L}^{-T}\mathsf{L}^{-1}\mathbf{z} \tag{39}$$

The quality of fit is usually assessed by residual sum of squares (RSS), which is the first term in (35). In case of penalized smoothing, several modified criteria are used, for example, generalized cross-validation score (GCV), or Akaike information criterion (AIC):

$$\mathrm{GCV} \equiv \frac{\mathrm{RSS}/N_d}{(1 - \mathrm{EDF}/N_d)^2} \quad , \quad \mathrm{AIC} \equiv \ln(\mathrm{RSS}) + \frac{2\,\mathrm{EDF}}{N_d - \mathrm{EDF} - 1} \quad , \quad \text{where} \tag{40}$$

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

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 AIC$(\lambda = 0)$ by a pre-defined parameter, ΔAIC. 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

$$L^2_{\mathrm{peri},k} = L^2_{\min} + s \times k/N_{\mathrm{peri}} + \delta_k \ , \quad k = 1..N_{\mathrm{fit}}. \tag{41}$$

Here N_{fit} is typically $0.1\,N_{\mathrm{peri}}$ (but in addition we require that $10 \le N_{\mathrm{fit}} \le 50$), so we study only the low-L part of the distribution. The parameters $L^2_{\min}$ 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

$$L^2_{\mathrm{peri},k} = s' \times k/N_{\mathrm{peri}} + \delta'_k, \tag{42}$$

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 σ^2_{fit}, same for each point, from the condition

$$\sigma^2_{\mathrm{fit}} = \frac{\sum_{k=1}^{N_{\mathrm{fit}}} \delta_k^2}{N_{\mathrm{fit}} - 2} \ , \tag{43}$$

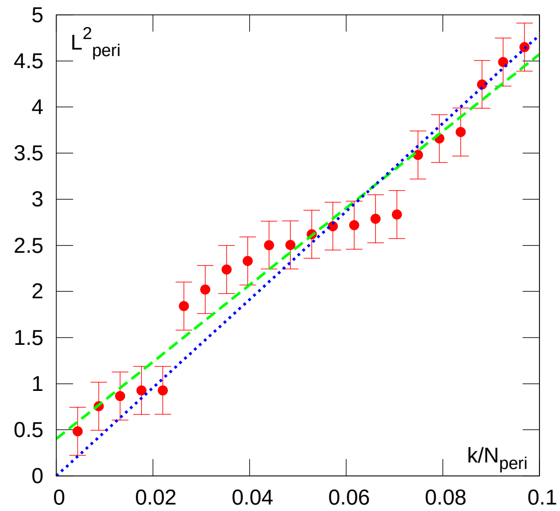
which is a standard practice giving exactly unity for χ^2 per number of degrees of freedom (d.o.f.) $N_{\rm fit}-2$. Next, we do the same for the one-parameter regression and compute

$$\Delta\chi^2 \equiv \chi^2_{\rm one-param} - \chi^2_{\rm two-param} = \frac{1}{\sigma^2_{\rm fit}} \sum_{k=1}^{N_{\rm fit}} \delta'^2_k - (N_{\rm fit}-2) \qquad (44)$$

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 (41) or if $\Delta\chi^2 < \Delta\chi^2_{\rm thr} = 11.8$ in (44), 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

$$\texttt{fitscatter} \equiv \sigma_{\rm fit}/\sigma_{\rm typical}\ , \qquad \text{where } \sigma_{\rm typical} \equiv \sqrt{N_{\rm fit}}\, s/N_{\rm peri} \qquad (45)$$

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. 41) and without (dotted blue, eq. 42) 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 (43), with the `fitscatter` parameter (45) 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. 44). 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 $w_o \ge 0, o=1..N_o$ 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, BSE or SHGrid), kinematics, 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 `constraintPenalty`. The first case, when `constraintPenalty`>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

$$m_c = \sum_{o=1}^{N_o} t_{oc} w_o + \mu_c - \nu_c \ , \ c = 1..N_c \tag{46}$$

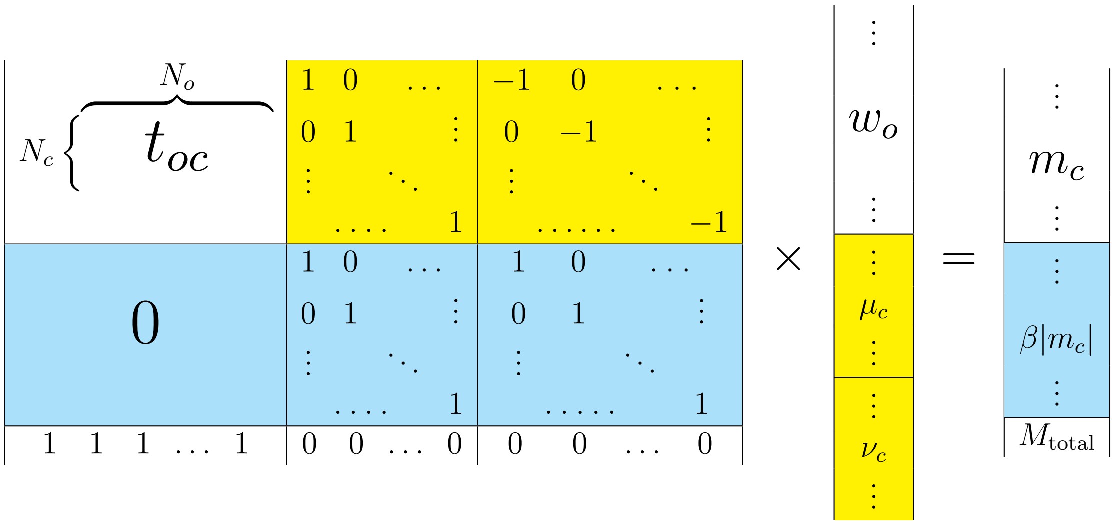
The objective function is

$$\mathcal{F} = \alpha \sum_{c=1}^{N_c} (\mu_c + \nu_c) + \mathcal{F}_{\rm additional} \ , \ \alpha \equiv \texttt{constraintPenalty} \tag{47}$$

In the second case we allow the deviation in the constraint value not to exceed $\beta|m_c|$. This is achieved by adding another N_c equations to (46):

$$\mu_c + \nu_c = \beta|m_c| \ , \ \beta \equiv -\texttt{constraintPenalty} \tag{48}$$

No additional terms are introduced in the objective function in this second case. It is parametrized by the fractional tolerance β, given by the same parameter `constraintPenalty` if it is negative. 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 `constraintPenalty=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 $\mathcal{F}_{\rm additional}$: the quadratic regularization term is given by

$$\mathcal{F}_{\rm quadratic} = \frac{\lambda}{N_o} \sum_{o=1}^{N_o} \left(\frac{w_o}{M_{\rm total}/N_o} \right)^2 \ , \ \lambda \equiv \texttt{regularization} \tag{49}$$

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

$$\mathcal{F}_{\rm bias} = \frac{\kappa}{M_{\rm total}} \sum_{o=1}^{N_o} w_o \times E(\{\mbox{orbit}\}) \;,\; \kappa \equiv \texttt{chaoticPenalty}, \qquad (50)$$

$$E = \left\{ \begin{array}{l} 1 \mbox{ if Lyapunov exponent } \Lambda > \Lambda_{\rm threshold} \\ \Xi(\log_{10}(\mbox{FDR}/\mbox{FDR}_{\rm threshold})) \mbox{ otherwise} \end{array} \right. ,\; \Xi(x) \equiv \left\{ \begin{array}{ll} 0, & x < -0.5 \\ 0.5 + x, & -0.5 \le x \le 0.5 \\ 1. & x > 0.5 \end{array} \right.$$

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 (47, 49, 50) are arranged so as to keep the magnitude of objective function independent of N_o and $M_{\rm total}$; 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).

C Additional programs

C.1 mkspherical

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

- `intab=[???]` text file with two columns: $r, M_{\mathrm{enclosed}}(r)$. Radii and masses must be in ascending order, nonzero value of $M(0)$ indicates the presence of central point mass (e.g. supermassive black hole). There is a simple perl script to generate this input text file for a user-specified analytical mass model.
- `insnap=[???]` N-body snapshot used as input data for mass model. The 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 $\log[M(r)/(M_{\mathrm{tot}} - M(r))]$ 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. Then the smoothed mass model is used as input in the same way as a text file in the `intab=???` mode.

  Either `insnap` or `intab` must be given, not both.
- `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 $f(E)$ $g(E)$ $M(E)$ T_{rad} r_{circ} L_{circ} σ_{1d} σ_{los} Σ D_{EE} $D_{\mathcal{RR}}$
  The dynamical quantities are given as functions of radius, projected radius, or energy ($E = \Phi(r)$). $f(E)$ is the distribution function, $g(E)$ is the density of states (according to the notation of [4]), and $M(E) \equiv \int_{-\infty}^{E} f(E')g(E')dE'$ is the total mass of particles having energies lower than E. $T_{\mathrm{rad}}(E)$ is the radial period, r_{circ} is the radius of circular orbit and L_{circ} is the angular momentum of that orbit. $\sigma_{\mathrm{1d}}(r)$ is the conventional one-dimensional velocity dispersion, $\sigma_{\mathrm{los}}(R)$ 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 $D_{\mathcal{RR}}(E)$ are the diffusion coefficients in energy and dimensionless squared angular momentum $\mathcal{R} \equiv L^2/L^2_{\mathrm{circ}}$, as given, for instance, in Chapter 5 of [5]. 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 [4] is given by $T_\mathrm{rel} = \frac{0.34\sigma^3_\mathrm{1d}(r)}{m_\star \rho(r) \ln\Lambda}$.

- `mbh=[0]` is the additional point mass (representing a supermassive black hole) at the center.
- `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=[Nemo]` – format of the output snapshot file (Text/Nemo/Gadget).
- `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 a BSE/Spline potential given by a coefs file (Sec. 4.4.4), by populating space with point masses according to local density (with zero velocities). It generates a sufficient number of points distributed according to the given spherically-averaged mass profile and isotropic in angles, computes the density at each point and draws a necessary number of sampling points from this set, proportionally to the density (which depends also on angles).

Parameters:

- `in=[???]` – input potential coefs file (Sec. 4.4.4).
- `out=[???]` – output snapshot file.
- `nbody=[???]` – number of particles to represent potential.
- `outformat=[Nemo]` – format of the snapshot file (Text/Nemo/Gadget).

C.3 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=[???]` – 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=[???]` – output OrbitLibrary file.
- `pot=[]` – file with potential coefficients, or a snapshot file containing field particles in which the test particles are assumed to move. 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_{orb}, so it needs not be a very accurate representation. If the potential is initialized from a N-body snapshot, its coefficients are stored in the coefs file with the same name as `pot` or `in` and extension `coef_bse/coef_spl`.
- `nrad=[20], nang=[6], sym=[t], pottype=[s]` – parameters for potential approximation if it is created from an N-body snapshot and not read from a coefs file: number of radial and angular terms, symmetry type ([n]one, [r]eflection, [t]riaxial, [a]xisymmetric, [s]pherical), and the potential type ([s]pline or [b]se).
- `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_{bh}, and input snapshot for creating a potential approximation should not contain it either.

C.4 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=[???]` – input snapshot 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 [12]).
  - Inertia tensor [f] of particles with iteratively determined axes (see [13] 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.5 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:

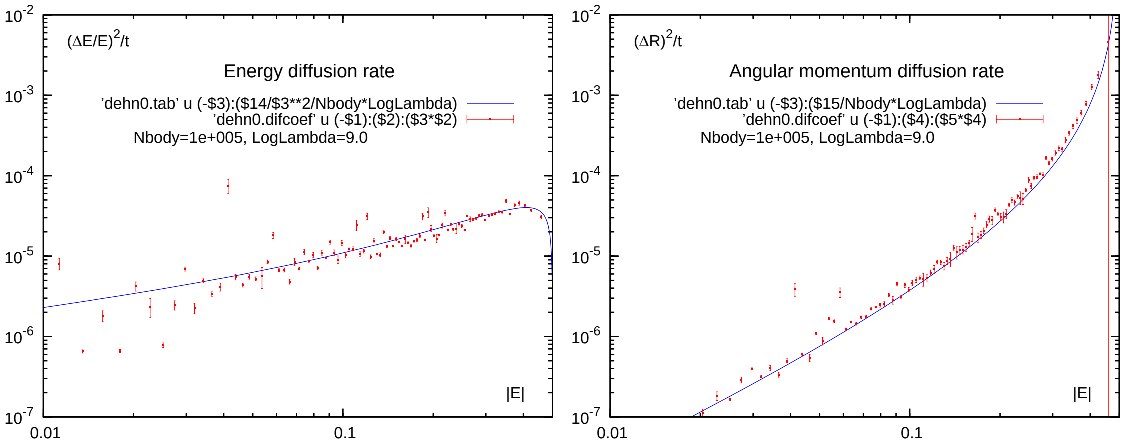
- `in=[???]` – input NEMO snapshot file.
- `nbins=[100]` – number of bins in energy.
- `minbin=[0]` – fraction of mass in the innermost bin (0 by default means 1/nbins).
- `rel=[f]` – if false, report the diffusion coefficients for $\langle \Delta E^2/t \rangle$ and $\langle \Delta L^2/t \rangle$, if 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 $\mathcal{R} \equiv L^2/L^2_{\mathrm{circ}}$ 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, in press

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

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

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

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

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

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

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

[9] Pfenniger D., 1984, A&A, 141, 171

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

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

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

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