Stellar-dynamical modelling with AGAMA

Eugene Vasiliev

Institute of Astronomy, Cambridge

29 November 2022

Basic elements of stellar dynamics

Software packages for stellar dynamics	`Galpy` [Bovy 2015]	`Gala` [Price-Whelan 2017]	`Agama` [Vasiliev 2019]
density and potential profiles:			
collection of analytic models	**+**	+	+
solution of the Poisson equation for an arbitrary $\rho(\mathbf{r})$ or an N-body snapshot	+	+	**+**
numerical integration of orbits	+	+	+
conversion between position/velocity and action/angle variables	+	+	**+**
distribution functions and their moments	+	-	**+**
construction of equilibrium models	-	-	**+**
modelling of tidal streams	-	+	-
integration with `astropy`	+	+	-
language	Python, C	Cython	C++, Python

Gravitational potentials

Commonly used analytic potential–density pairs: `Plummer`, `NFW`, `MiyamotoNagai`, `Dehnen`, `Ferrers` ...

If one needs more flexibility, there are three general-purpose Poisson solvers:

0. Direct integration:

$$\cancel{\Phi(\mathbf{x}) = -\iiint d^3x'\, \rho(\mathbf{x}') \times \frac{G}{|\mathbf{x}-\mathbf{x}'|}}. \qquad \text{(impractical)}$$

1. Azimuthal-harmonic expansion (`CylSpline`):

$$\Phi(R,z,\phi) = \sum_{m=-\infty}^{\infty} \Phi_m(R,z)\, \mathrm{e}^{im\phi}.$$

interpolated functions

2. Spherical-harmonic expansion (`Multipole`):

$$\Phi(r,\theta,\phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \Phi_{lm}(r)\, Y_l^m(\theta,\phi).$$

3. `BasisSet` expansion (a.k.a. self-consistent field method of Hernquist&Ostriker 1992):

$$\Phi(r,\theta,\phi) = \sum_{n=0}^{\infty} \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \Phi_{nlm}\, A_{nl}(r)\, Y_l^m(\theta,\phi).$$

Gravitational potentials

Both `BasisSet` and `Multipole` use spherical-harmonic expansion to represent the angular dependence of the potential, but the radial part is either expanded into a sum of basis function or represented by quintic splines.

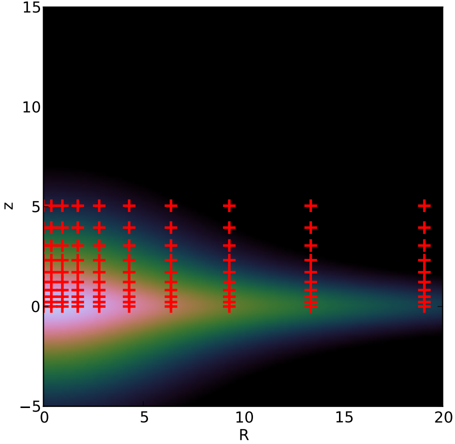
In practice, `Multipole` is more computationally efficient and more accurate (at least for analytic density profiles); `BasisSet` is kept mainly for compatibility with other packages.

Spherical harmonics are poorly suited for highly flattened systems; a better alternative is the `CylSpline` potential, which uses Fourier expansion for the azimuthal ϕ angle, but represents each Fourier term directly on a grid in the meridional plane R, z. It is more expensive to construct than `Multipole`, but is similarly efficient to evaluate.

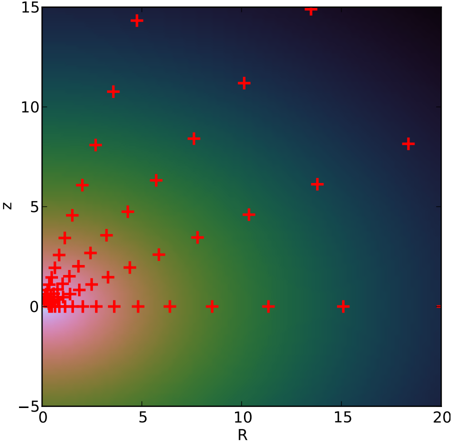
These general-purpose Poisson solvers can be initialized from arbitrary user-defined density functions or N-body snapshots.

Gravitational potentials

- for spheroidal density profiles – spherical-harmonic expansion (`Multipole`).
- for disk-like density profiles – azimuthal-harmonic expansion (`CylSpline`).



`CylSpline`



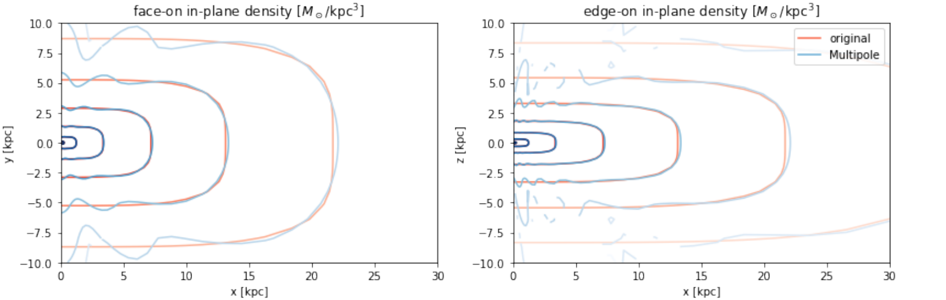
`Multipole`

Gravitational potentials: example 1

User-defined density model: a boxy bar $\rho(x, y, z) = \rho_0 \exp\left(-d^{1/n}\right)$,
where $d \equiv \left[(x/a)^k + (y/b)^k + (z/c)^k\right]^{1/k}$ is the generalized ellipsoidal radius (an ordinary ellipsoid has $k = 2$), and n is the Einasto index.

```
def dens_bar(xyz):
    x,y,z = abs(xyz).T
    r = ((x/a)**k + (y/b)**k + (z/c)**k)**(1./k)
    return rho0 * numpy.exp(-r**(1./n))

pot_bar = agama.Potential(type='Multipole', density=dens_bar,
    lmax=20, mmax=10)
```



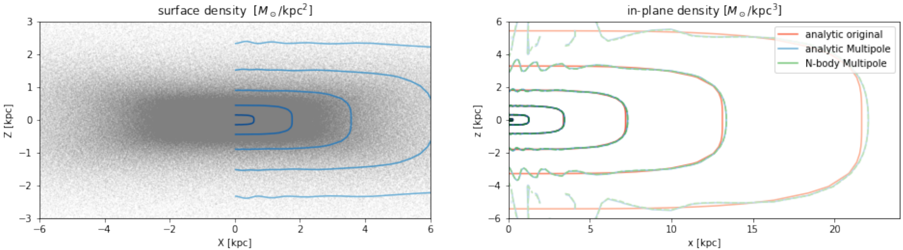
Gravitational potentials: example 1

Create an N-body snapshot from the density profile
(only positions and masses, no velocities):

```
pos, mass = agama.Density(dens_bar).sample(1000000)
```

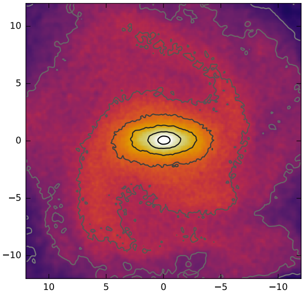
And then feeding this snapshot as input to the Multipole potential:

```
pot_bar_nbody = agama.Potential(type='Multipole',
    particles=(pos, mass), symmetry='triaxial', lmax=20, mmax=10)
```

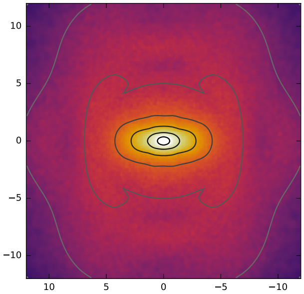


Gravitational potentials: example 2

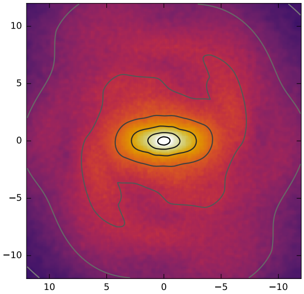
One may construct a smooth potential approximation with a desired level of symmetry from an N-body simulation (or even a time-dependent potential from a series of snapshots) and use it to integrate and analyze orbits, e.g., in a barred galaxy.



original snapshot

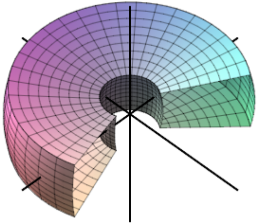


triaxial



bisymmetric

Action–angle variables



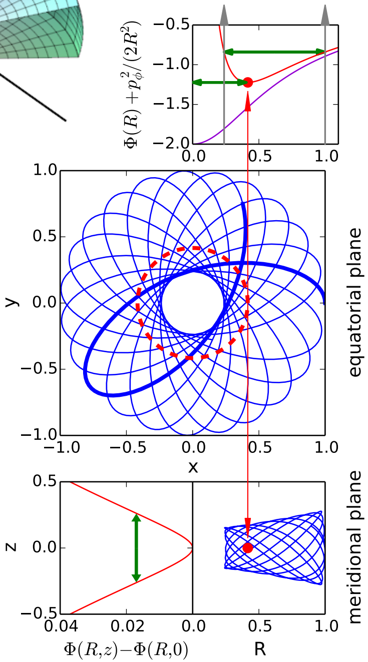
Most orbits in axisymmetric potentials look like "rectangular tori" with three parameters defining the shape:
$J_\phi \equiv L_z = R_\mathrm{g}\, v_\mathrm{circ}(R_\mathrm{g})$ determines the overall size of the orbit ("guiding radius" R_g);
J_R determines the extent of radial oscillations;
J_z does the same for vertical oscillations.

For orbits with low eccentricity and inclination ($J_{R,z} \ll |J_\phi|$), the epicyclic approximation gives

$$J_R \approx \frac{1}{2\pi} \oint V_R\, \mathrm{d}R = \frac{1}{\pi} \int_{R_\mathrm{peri}}^{R_\mathrm{apo}} \sqrt{2\left[E - \Phi(R)\right] - (L_z/R)^2}\, \mathrm{d}R.$$

The Stäckel approximation [Binney 2012] is the state-of-the-art method for generic axisymmetric potentials, and delivers percent-level accuracy for most orbits.

Corresponding phase angles $\theta_{\phi,R,z}$ determine the location on the orbit.



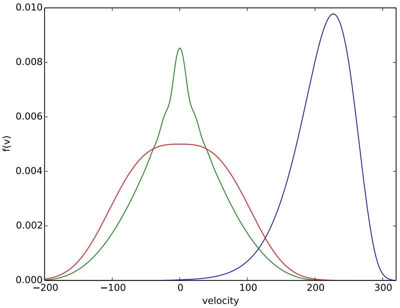
Distribution functions

DF $f(\mathbf{x},\mathbf{v})$ offers a complete description of the stellar population:

- density $\rho(\mathbf{x}) = \int f(\mathbf{x},\mathbf{v})\, d^3v$,
- mean velocity $\overline{\mathbf{v}}(\mathbf{x}) = \dfrac{1}{\rho(\mathbf{x})}\int \mathbf{v}\, f(\mathbf{x},\mathbf{v})\, d^3v$,
- second moment of velocity $\overline{v^2_{ij}}(\mathbf{x}) = \dfrac{1}{\rho(\mathbf{x})}\int v_i\, v_j\, f(\mathbf{x},\mathbf{v})\, d^3v$,

  velocity dispersion tensor $\sigma^2_{ij} \equiv \overline{v^2_{ij}} - \overline{v_i}\,\overline{v_j}$,
- more generally, velocity distribution at a given point $\mathfrak{f}(v_1;\ \mathbf{x}) = \dfrac{1}{\rho(\mathbf{x})}\int f(\mathbf{x},\mathbf{v})\, dv_2\, dv_3$

  (note that it may be rather non-Gaussian!).



Jeans' theorem: in a steady state, DF must be a function of integrals of motion $f\big(\mathcal{I}(\mathbf{x},\mathbf{v};\ \Phi)\big)$.

Distribution functions

Correspondence between DF and density profile:

$$\rho(\mathbf{x}) = \int f\left(\mathcal{I}(\mathbf{x}, \mathbf{v};\ \Phi)\right) \mathrm{d}^3 v.$$

In spherical systems: $\mathbf{x} \Rightarrow r,\ \mathcal{I} \Rightarrow \{E, L, L_z\}$.

In axisymmetric systems: $\mathbf{x} \Rightarrow \{R, z\},\ \mathcal{I} \Rightarrow \{E, L_z, I_3\}$ or $\{J_R, J_\phi, J_z\}$.

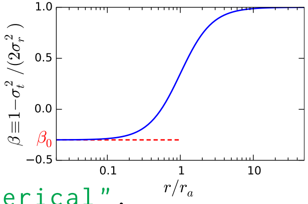
Whatever the geometry, there is more freedom in DF than in the density profile $\Longrightarrow$ DF is non-unique.

To derive f from ρ and Φ, need to assume a specific form of the DF.

E.g. for spherical systems: Cuddeford–Osipkov–Merritt inversion:

$f(E, L) = L^{-2\beta_0} f_Q(Q)$, where $Q \equiv E + L^2/(2r_a^2)$.

Velocity anisotropy β changes with radius from β_0 to 1.



Example:

```
df = agama.DistributionFunction(type="QuasiSpherical",
    density=dens, potential=pot, beta0=-0.3, r_a=1)
```

Distribution functions for axisymmetric systems

For non-spherical systems, one could use $f(E, L, L_z)$, but action-based DFs $f(\mathbf{J})$ offer some practical advantages (e.g., their definition does not depend on Φ, although the kinematics of course does).

Example of a halo-type DF [Posti+ 2015]:

$$f(\mathbf{J}) = A\left[1 + \left(J_0/h(\mathbf{J})\right)^{\eta}\right]^{\Gamma/\eta}\left[1 + \left(g(\mathbf{J})/J_0\right)^{\eta}\right]^{(\Gamma-\mathrm{B})/\eta},$$

where $h(\mathbf{J})$, $g(\mathbf{J})$ are some linear combinations of actions at small and large radii.

```
df = agama.DistributionFunction(type="DoublePowerLaw",
    J0=1.0, slopeIn=1.0, slopeOut=5.0, steepness=1.0, norm=1.0,
    coefJrIn=0.8, coefJzIn=1.5, coefJrOut=0.9, coefJzOut=1.6)
```

The density and kinematics produced by a DF depend on the potential, but generally these systems can be made flattened in z, radially or tangentially anisotropic, and rotating.

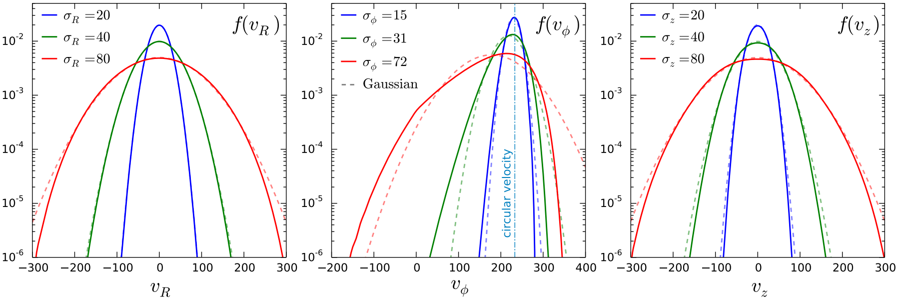
Quasi-isothermal DFs [Binney & McMillan 2011] produce disk-like systems with nearly-exponential surface density $\Sigma(R) \approx \Sigma_0 \exp(-R/R_{\mathrm{disk}})$, constant scaleheight h_{disk} and radial velocity dispersion $\sigma_R(r) \approx \sigma_{R,0} \exp(-R/R_{\sigma,R})$:

```
df = agama.DistributionFunction(type="QuasiIsothermal",
    Sigma0=1.0, Rdisk=1.0, hdisk=0.1, sigmaR0=0.5, RsigmaR=2.5)
```

Using distribution functions

A combination of a DF and a potential is used to compute DF moments ($\rho, \overline{v}_i, \overline{v_i v_j}$), marginalized values over some missing dimensions (e.g., projected DF $\hat{f}(x, y, v_z) = \iiint f\big(J(\mathbf{x}, \mathbf{v};\ \Phi)\big)\, \mathrm{d}v_x\, \mathrm{d}v_y\, \mathrm{d}z$), velocity distributions $f(\mathbf{x}, v_i)$, or drawing samples from the DF:

```
galmod = agama.GalaxyModel(pot, df)
rho, meanv, sigma = galmod.moments(xyz, dens=True, vel=True, vel2=
dfproj = galmod.projectedDF(xyvz)
fvR,fvphi,fvz = galmod.vdf(xyz)
xv, m = galmod.sample(1000000)
```



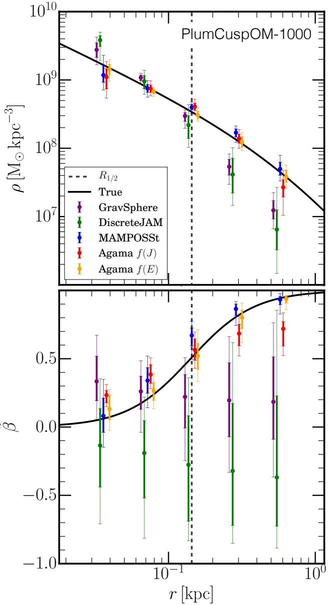
Example application

Inferring the gravitational potential of dwarf galaxies from the kinematics of stellar tracers (which have negligible mass).
Given $N_{\rm stars}$ measurements of x, y and $v_{\rm los}$, the likelihood of a model specified by a potential $\Phi(\mathbf{r};\ \beta_\Phi)$ and DF $f(\mathcal{I};\ \beta_f)$ with some parameters β_Φ, β_f is

$$\ln \mathcal{L} = \sum_{i=1}^{N_{\rm stars}} \ln f\big(\mathcal{I}(\mathbf{x}_i, \mathbf{v}_i;\ \Phi)\big).$$

A comprehensive test on mock data ("Gaia Challenge") [Read+ 2021] demonstrates good accuracy of DF models.

In this example I used both $f(\mathbf{J})$ and $f(E, L)$ constructed from Φ and ρ, but the potential was specified independently from the DF (i.e. not self-consistently).

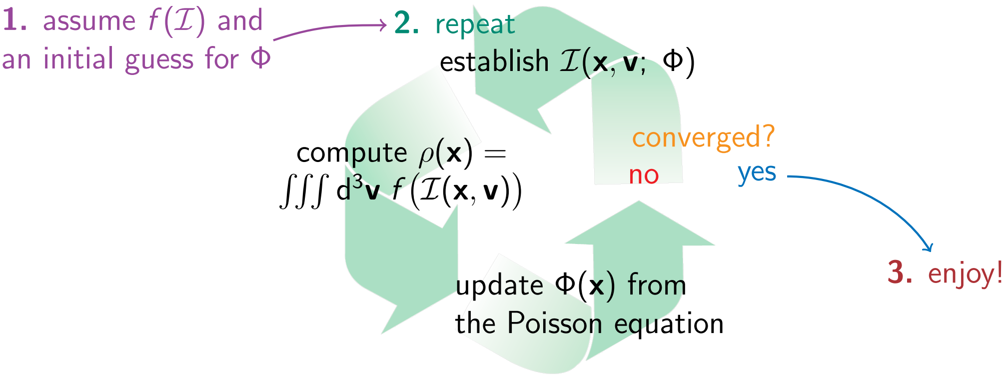


Distribution functions and self-consistent models

One may start with f and determine the corresponding ρ and Φ.

A general DF $f(\mathcal{I})$ is specified in terms of integrals of motion in the given potential $\mathcal{I}(\mathbf{x}, \mathbf{v};\ \Phi)$. To compute the density $\rho(\mathbf{x})$ generated by this DF, one needs to know $\Phi(\mathbf{x})$, but in the gravitationally self-consistent case, Φ is determined by ρ via the Poisson equation – thus we have a circular dependency.

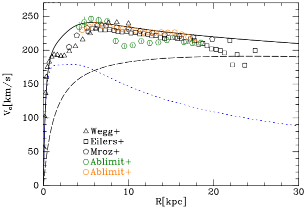
Such models are constructed by the iterative approach [Kuijken & Dubinski 1995; Widrow+ 2005], which works best for action-based DFs [Binney 2014; Piffl+ 2015; Binney & Vasiliev 2022]:

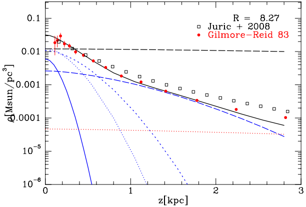


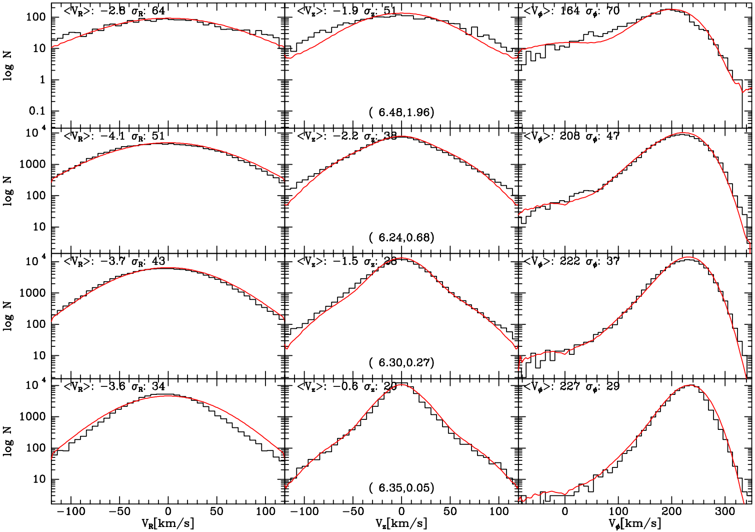
Example application

Global model of the Milky Way specified by several disk-like DFs and constrained by velocity distributions of Gaia DR2 stars with 6d phase-space coordinates

[Binney & Vasiliev 2022].







Schwarzschild's orbit-superposition method

Introduced by Schwarzschild (1979) as a practical approach
for constructing self-consistent triaxial models with prescribed $\rho(\mathbf{x}) \Leftrightarrow \Phi(\mathbf{x})$.

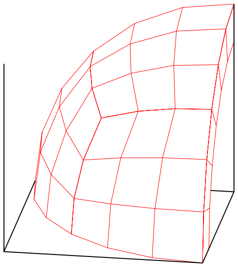
integrals of motion

To invert the equation $\rho(\mathbf{x}) = \iiint f\left(\mathcal{I}\left[\mathbf{x}, \mathbf{v} \mid \Phi\right]\right)\, d^3\mathbf{v}$,

discretize both the density profile and the distribution function:

$\rho(\mathbf{x}) \implies$ cells of a spatial grid;

mass of each cell is $M_c = \iiint\limits_{\mathbf{x}\in V_c} \rho(\mathbf{x})\, d^3x$;



$f(\mathcal{I}) \implies$ collection of orbits with unknown weights:

$$f(\mathcal{I}) = \sum_{k=1}^{N_\mathrm{orb}} w_k\, \delta(\mathcal{I} - \mathcal{I}_k)$$

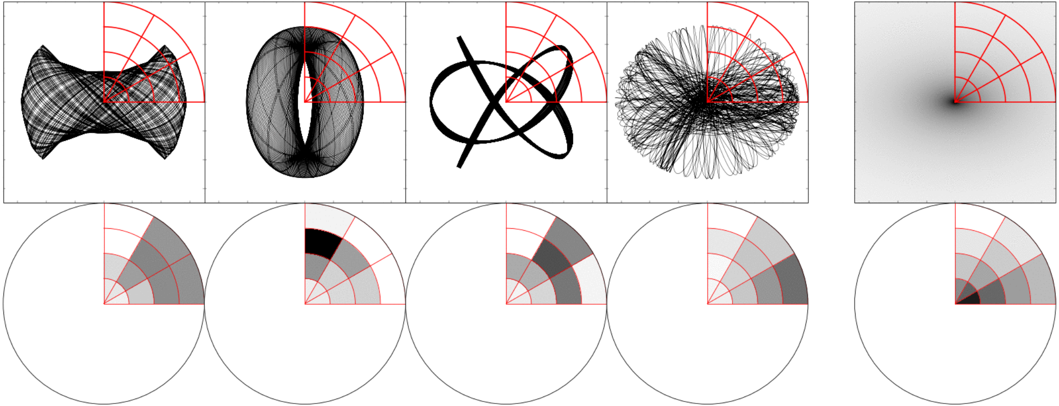
each orbit is a delta-function in the space of integrals of motion

adjustable weight of each orbit [to be determined]

Schwarzschild's orbit-superposition method: self-consistency

orbits in the model

target density



discretized orbit density

(fraction of time t_{kc} that k-th orbit spends in c-th cell)

discretized density

(mass M_c in grid cells)

For each c-th cell we require $\sum_k w_k \, t_{kc} = M_c$, where $w_k \geq 0$ is orbit weight

Schwarzschild's orbit-superposition method: fitting procedure

- Assume some potential $\Phi(\mathbf{x})$
  (e.g., from the deprojected luminosity profile plus parametric DM halo or SMBH)
- Construct the orbit library in this potential:
  for each k-th orbit, store its contribution to the discretized density profile t_{kc}, $c = 1..N_\mathrm{cell}$ and to the kinematic observables u_{kn}, $n = 1..N_\mathrm{obs}$
- Solve the constrained optimization problem to find orbit weights w_k:

$$\text{minimize } \chi^2 + \mathcal{S} \equiv \sum_{n=1}^{N_\mathrm{obs}} \left(\frac{\sum_{k=1}^{N_\mathrm{orb}} w_k\, u_{kn} - U_n}{\delta U_n} \right)^2 + \mathcal{S}(\{w_k\})$$

regularization term

observational constraints

their uncertainties

$$\text{subject to } w_k \ge 0, \quad k = 1..N_\mathrm{orb},$$

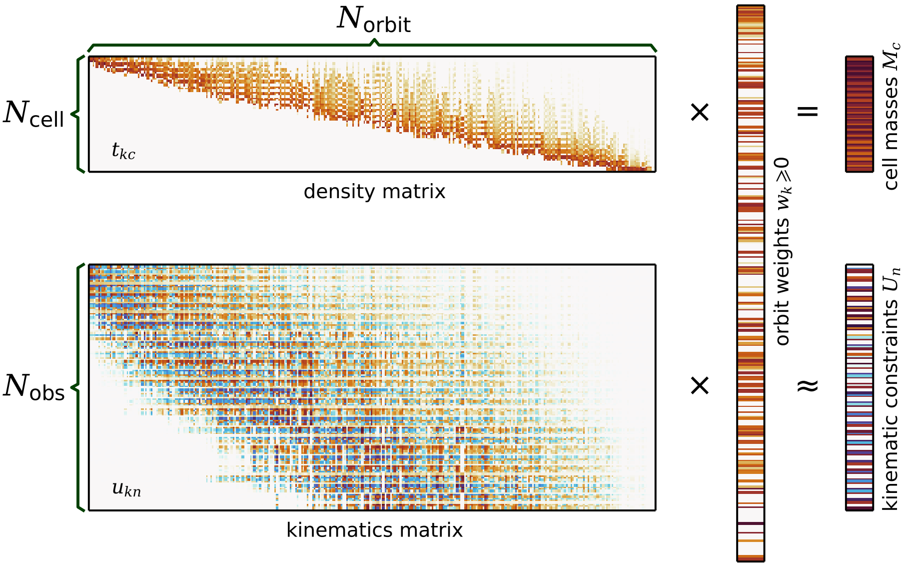
$$\sum_{k=1}^{N_\mathrm{orb}} w_k\, t_{kc} = M_c, \quad c = 1..N_\mathrm{cell}$$

density constraints (cell masses)

- Repeat for different choices of potential and find the one that has lowest χ^2

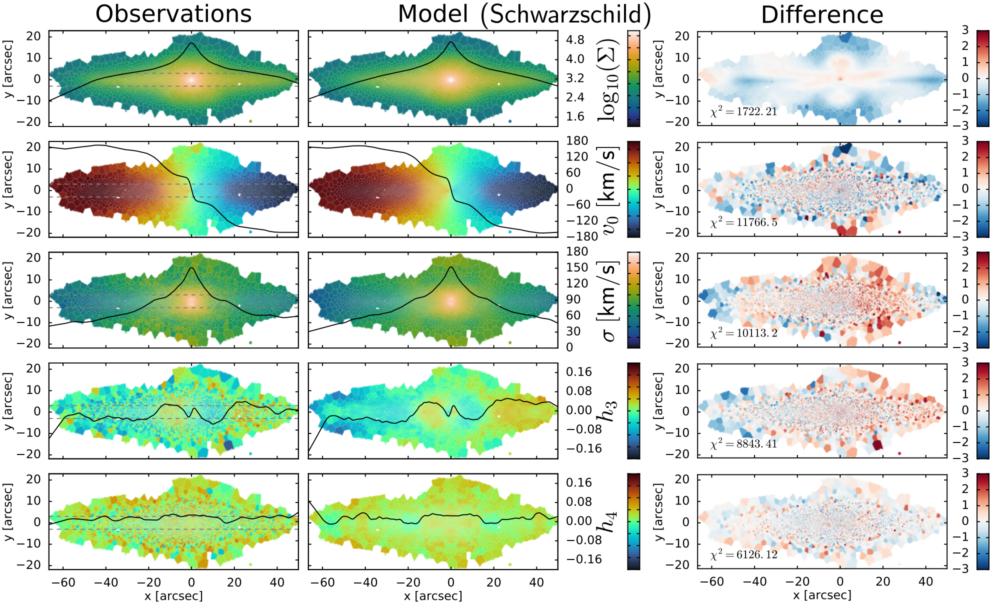
Schwarzschild's orbit-superposition method: fitting procedure

Solve the linear system with non-negativity constraints on the solution vector $w_k \geq 0$ (linear or non-linear optimization problem)



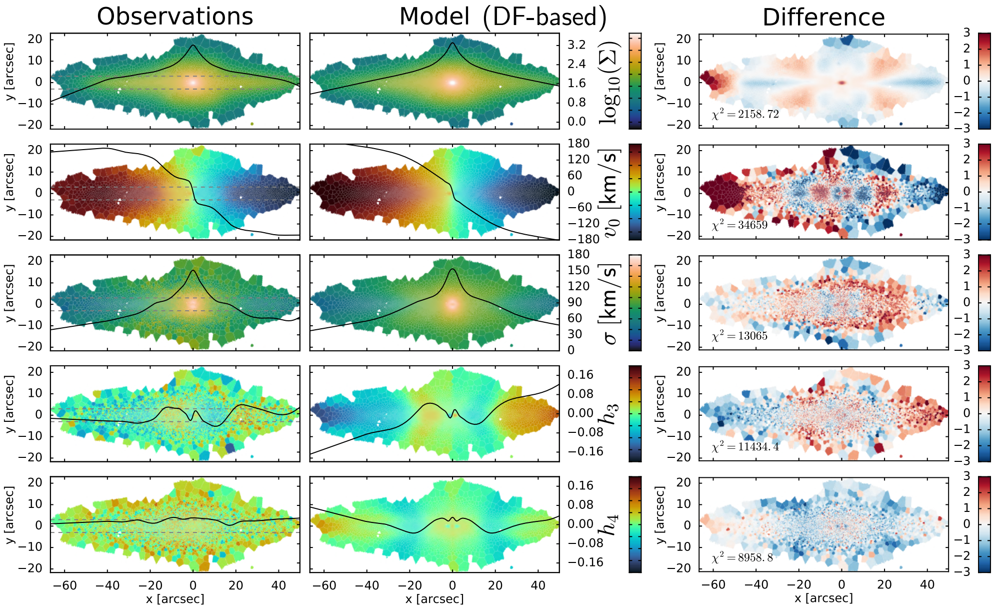
Example application

Model of an edge-on S0 galaxy FCC 170 constrained by MUSE IFU kinematics



Example application

Model of an edge-on S0 galaxy FCC 170 constrained by MUSE IFU kinematics



Summary

Agama is a versatile toolbox for stellar dynamics catering to many needs:

- Extensive collection of gravitational potential models
  (analytic profiles, azimuthal- and spherical-harmonic expansions)
  constructed from smooth density profiles or N-body snapshots;
- Conversion to/from action/angle variables;
- Self-consistent multicomponent models with action-based DFs;
- Schwarzschild orbit-superposition models;
- Generation of initial conditions for N-body simulations;
- Various math tools: 1d,2d,3d spline interpolation, penalized spline fitting and density estimation, multidimensional sampling;
- Efficient and carefully designed C++ implementation, examples, Python and Fortran interfaces, plugins for Galpy, Gala, NEMO, AMUSE.
  `https://github.com/GalacticDynamics-Oxford/Agama`