Galactic dynamics with AGAMA

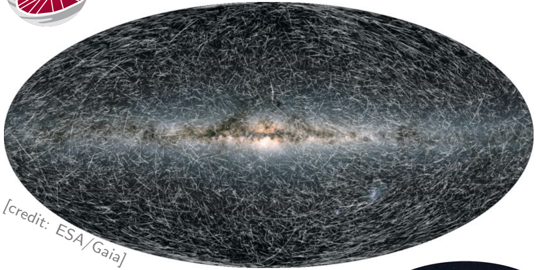
Eugene Vasiliev

Challenges and Innovations in Computational Astrophysics V

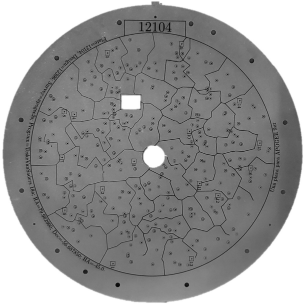
9 November 2023

Observational context: Galactic and extragalactic

gaia



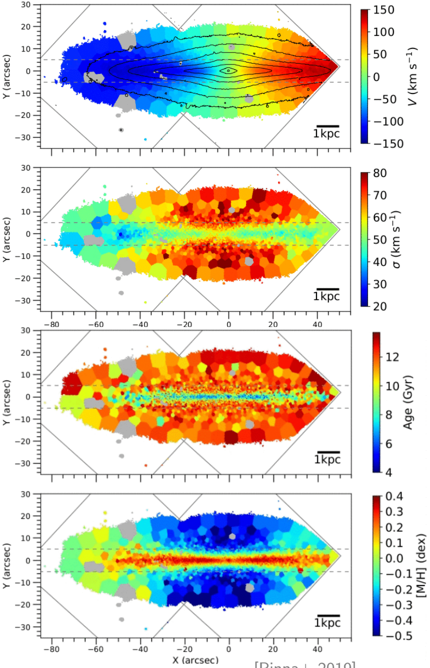
[credit: ESA/Gaia]



APOGEE

GALAH

muse
multi unit spectroscopic explorer



[Pinna+ 2019]

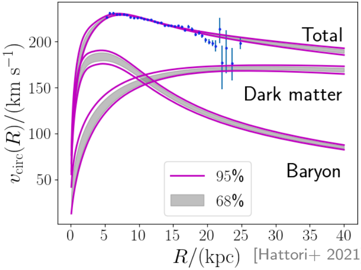
Astrophysical context

- Orbits of individual stars, clusters, etc.
- Classification and properties of stellar populations.
- Mass distribution in galaxies (DM haloes, SMBH. . .)

Key software requirements

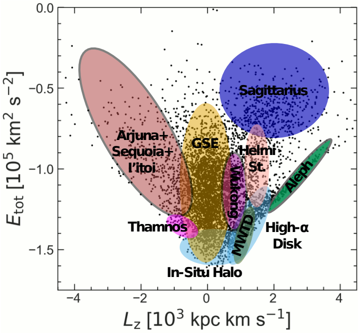
- Represent density and gravitational potential models.
- Compute and characterize orbits of stars.
- Handle stellar distribution functions.
- Fit galactic models to observational data.
- Help to set up and analyze N-body simulations.
- Interact with other stellar-dynamical software.
- Be flexible and extensible.

Milky Way rotation curve



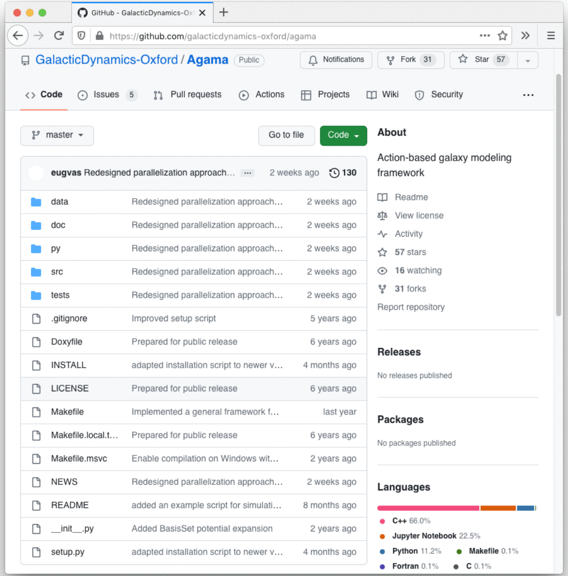
[Hattori+ 2021]

Classification of accreted stars



[Naidu+ 2020]

Agama software package



eugvas Redesigned parallelization approach… ··· 2 weeks ago 130

data	Redesigned parallelization approach…	2 weeks ago
doc	Redesigned parallelization approach…	2 weeks ago
py	Redesigned parallelization approach…	2 weeks ago
src	Redesigned parallelization approach…	2 weeks ago
tests	Redesigned parallelization approach…	2 weeks ago
.gitignore	Improved setup script	5 years ago
Doxyfile	Prepared for public release	6 years ago
INSTALL	adapted installation script to newer v…	4 months ago
LICENSE	Prepared for public release	6 years ago
Makefile	Implemented a general framework f…	last year
Makefile.local.t…	Prepared for public release	6 years ago
Makefile.msvc	Enable compilation on Windows wit…	2 years ago
NEWS	Redesigned parallelization approach…	2 weeks ago
README	added an example script for simulati…	8 months ago
__init__.py	Added BasisSet potential expansion	2 years ago
setup.py	adapted installation script to newer v…	4 months ago

About

Action-based galaxy modeling framework

Readme
View license
Activity
57 stars
16 watching
31 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

● **C++** 66.0%
● **Jupyter Notebook** 22.5%
● **Python** 11.2%
● **Makefile** 0.1%
● **Fortran** 0.1%
● **C** 0.1%

Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY
MNRAS **482,** 1525–1544 (2019)
Advance Access publication 2018 October 3

AGAMA: action-based galaxy modelling architecture

Eugene Vasiliev [1,2,3★]

development started in 2015;
code paper published in 2018;
used in ∼250 publications.

Main features:

- core library written in C++;
- OpenMP parallelization;
- hand-made Python interface;
- extensible with user-defined functions;
- several dozen tests and example programs in C++ and Python;
- detailed documentation (∼130 pages);
- ∼80 000 lines of code.

Gravitational potential

Task: given the density profile $\rho(\mathbf{x})$, determine the potential $\Phi(\mathbf{x})$ from the Poisson equation: $\nabla^2\Phi = 4\pi\, G\, \rho$.

Example 1: spherical Plummer model

$$\rho(r) = \frac{3\,M}{4\pi\, a^3\, \left(1 + r^2/a^2\right)^{5/2}} \quad\Longrightarrow\quad \Phi(r) = -\frac{G\,M}{\sqrt{r^2+a^2}}.$$

Example 2: triaxial Hernquist model

$$\rho(x,y,z) = \frac{M}{2\pi\, abc\; s\,(1+s)^3},\quad s \equiv \sqrt{\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}} \quad\Longrightarrow$$

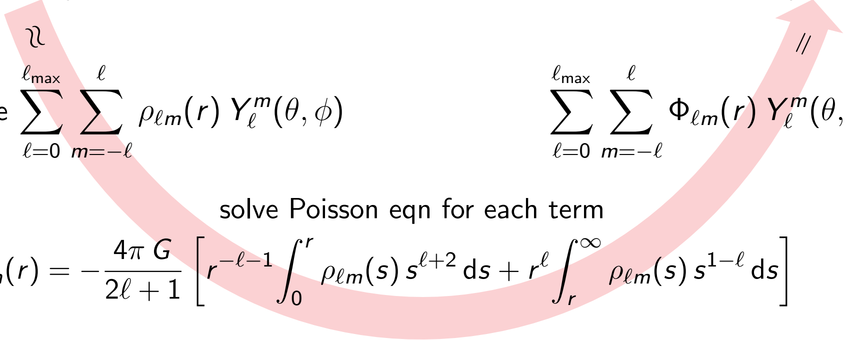
$$\Phi(x,y,z) = -G\,M \int_0^\infty \mathrm{d}\tau\, \frac{\left(1+\sqrt{\frac{x^2}{a^2+\tau}+\frac{y^2}{b^2+\tau}+\frac{z^2}{c^2+\tau}}\right)^{-2}}{2\sqrt{(a^2+\tau)(b^2+\tau)(c^2+\tau)}}.$$

It gets very complicated very quickly!

Gravitational potential

General solution(s): separable potential expansions

spherical-harmonic (`Multipole`, `BasisSet`), azimuthal-harmonic (`CylSpline`).



original $\rho(r,\theta,\phi)$ $\Phi(r,\theta,\phi)$

$\wr$ $\parallel$

approximate $\sum_{\ell=0}^{\ell_\mathrm{max}} \sum_{m=-\ell}^{\ell} \rho_{\ell m}(r)\, Y_\ell^m(\theta,\phi)$ $\sum_{\ell=0}^{\ell_\mathrm{max}} \sum_{m=-\ell}^{\ell} \Phi_{\ell m}(r)\, Y_\ell^m(\theta,\phi)$

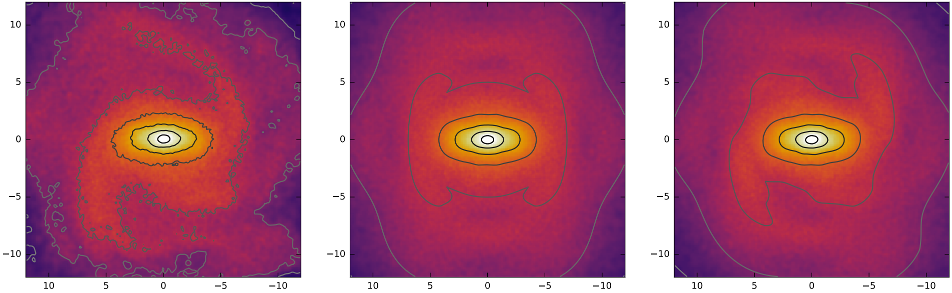
solve Poisson eqn for each term

$$\Phi_{\ell m}(r) = -\frac{4\pi\,G}{2\ell+1}\left[r^{-\ell-1}\int_0^r \rho_{\ell m}(s)\, s^{\ell+2}\,\mathrm{d}s + r^\ell \int_r^\infty \rho_{\ell m}(s)\, s^{1-\ell}\,\mathrm{d}s\right]$$

a similar procedure for `CylSpline` with $\rho \approx \rho_m(R,z)\,\mathrm{e}^{im\phi}$, etc.

Gravitational potential

One may construct these potential expansions either from an analytic density profile (including any user-defined Python function for ρ or Φ) or from an N-body snapshot, specifying the desired level of symmetry.



original snapshot

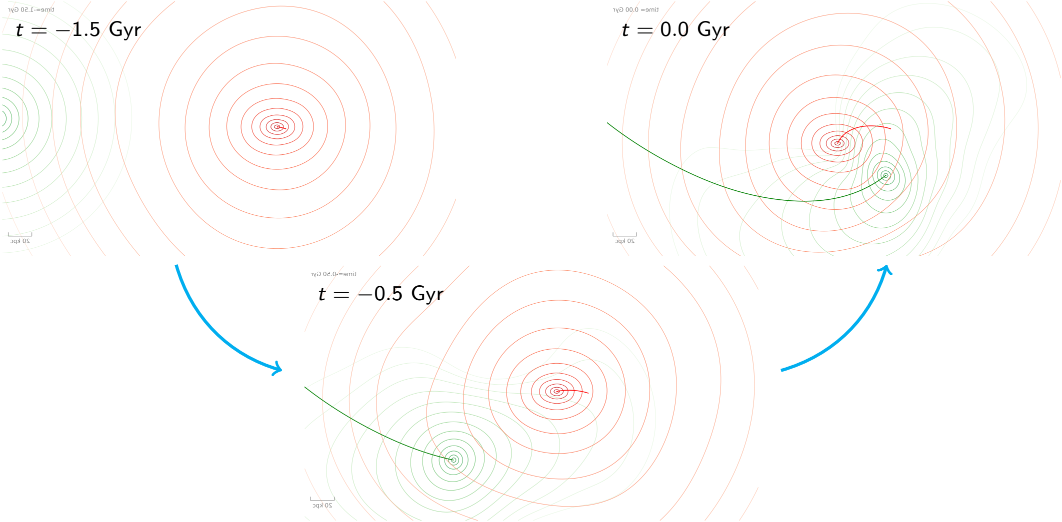
triaxial

bisymmetric

Composite and time-dependent potentials

Potentials can be added, scaled or interpolated with time, rotated (e.g., bar or spiral arms), shifted along a time-dependent trajectory, etc.

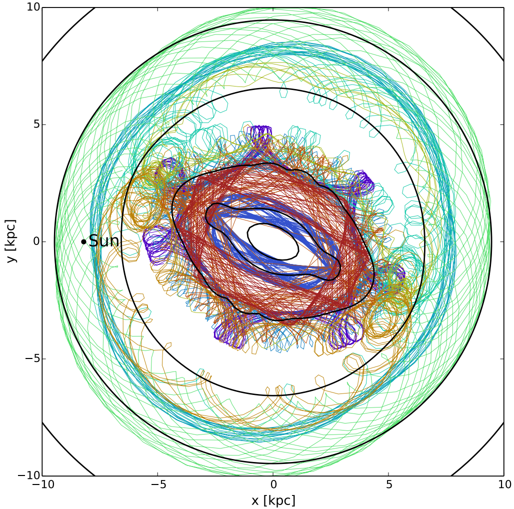
Example: potentials of Milky Way and LMC extracted from an N-body simulation.

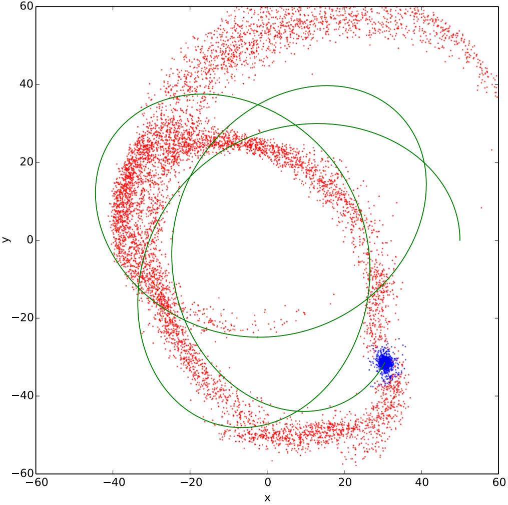


Numerical integration of orbits

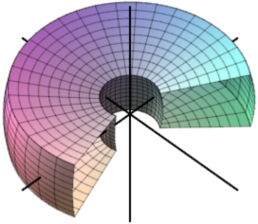
One of the most common tasks in galactic dynamics.

Examples: orbits in the MW bar; test-particle simulations of a tidal stream





Action–angle variables



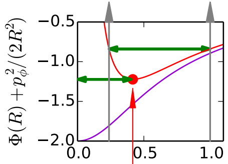
Most orbits in axisymmetric potentials look like "rectangular tori" with three parameters defining the shape:

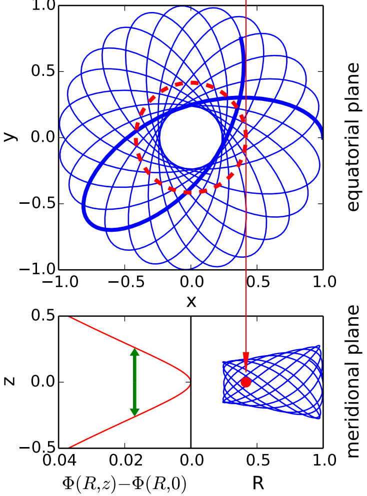
$J_\phi \equiv L_z = R_\mathrm{g}\, v_\mathrm{circ}(R_\mathrm{g})$ determines the overall size of the orbit ("guiding radius" R_g);

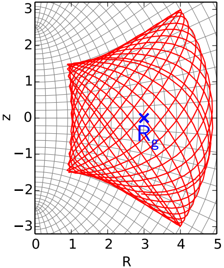
J_R determines the extent of radial oscillations;

J_z does the same for vertical oscillations.

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







Actions are defined as

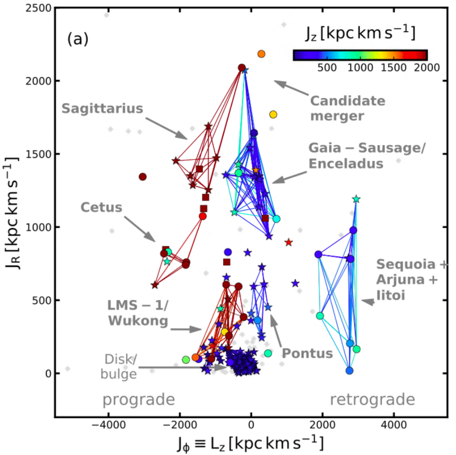
$$J_i \equiv \frac{1}{2\pi} \oint p_i\, \mathrm{d}x_i,$$

and are computed in the Stäckel approximation [Binney 2012], using spheroidal coordinates for x_i.

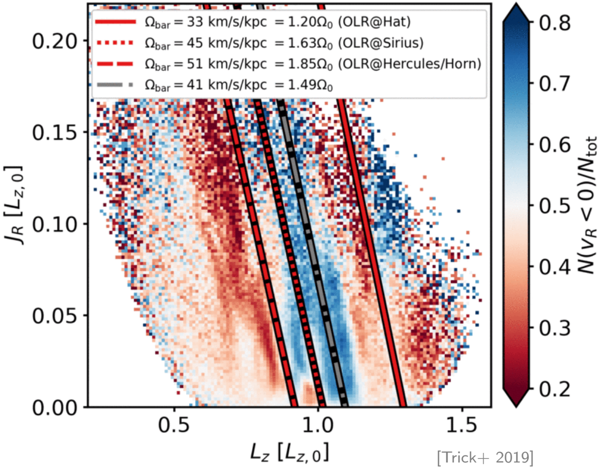
Example of use of action variables

Dynamical classification of stars and other objects into groups tracing a common origin

Measurement of Galactic bar pattern speed from resonances in action space



[Malhan+ 2022]

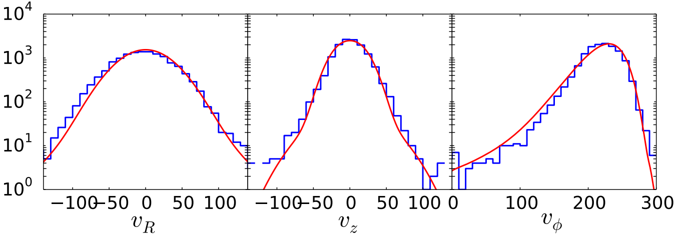


[Trick+ 2019]

Distribution functions

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

- density $\rho(\mathbf{x}) = \int f(\mathbf{x},\mathbf{v})\,\mathrm{d}^3v$,
- mean velocity $\overline{\mathbf{v}}(\mathbf{x}) = \dfrac{1}{\rho(\mathbf{x})}\int \mathbf{v}\, f(\mathbf{x},\mathbf{v})\,\mathrm{d}^3v$,
- second moment of velocity $\overline{v^2_{ij}}(\mathbf{x}) = \dfrac{1}{\rho(\mathbf{x})}\int v_i\, v_j\, f(\mathbf{x},\mathbf{v})\,\mathrm{d}^3v$,
- more generally, velocity distribution at a given point $\mathfrak{f}(v_1;\,\mathbf{x}) = \dfrac{1}{\rho(\mathbf{x})}\int f(\mathbf{x},\mathbf{v})\,\mathrm{d}v_2\,\mathrm{d}v_3$ (it can be strongly non-Gaussian!).



Distribution functions

Fundamental principle of stellar dynamics (Jeans's theorem):

in a steady state, DF must be a function of integrals of motion $f\big(\mathcal{I}(\mathbf{x},\mathbf{v};\ \Phi)\big)$, and it is often convenient to use actions $\mathbf{J}$ as integrals $\mathcal{I}$.

Example: spherical *isotropic* Plummer model

$$\Longrightarrow\ f(E) = \frac{24\sqrt{2}\,a^2}{7\pi^3\,G^5\,M^4}\,|E|^{-7/2}.$$

In practice, realistic DFs are quite a bit more complex –

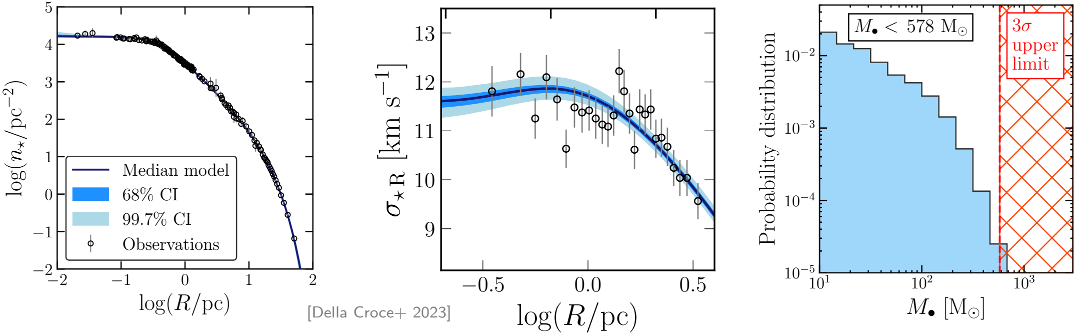
various DF families available in AGAMA are suitable for disky or spheroidal stellar systems, and one can always write own DF as a Python function.

Applications of distribution functions

DF is a *probability distribution* for finding a star with a given position and velocity, and it also depends on the potential Φ via the integrals of motion.

By maximising the likelihood of the observed dataset, one can determine the best-fit parameters of the stellar system, including its mass distribution.

Example: dynamical modelling of the globular cluster NGC 104 with the goal of constraining the mass of the central IMBH.

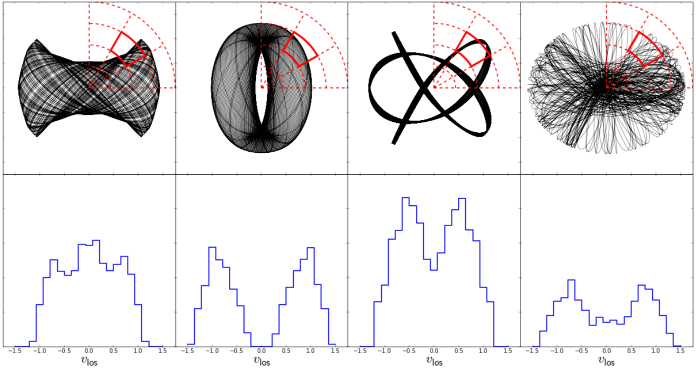


[Della Croce+ 2023]

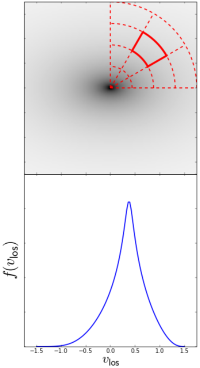
Orbit-superposition dynamical models [Schwarzschild 1979]

- assume some potential Φ;
- construct a library of $\mathcal{O}(10^4 - 10^5)$ orbits, record their kinematic footprint;
- determine orbit weights that maximize the resemblance of their sum to obs.data;
- repeat for many choices of potential to find the overall best-fit parameters.

FORSTAND is an efficient and flexible implementation of this method [Vasiliev & Valluri 2020].



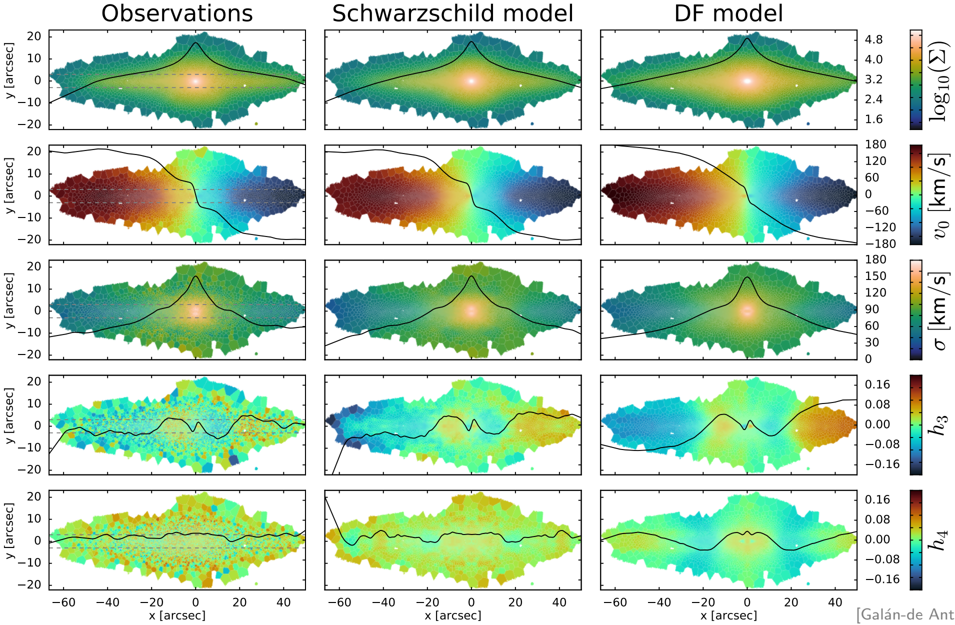
orbit LOSVDs (line-of-sight velocity distributions)



target LOSVD

Example application

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



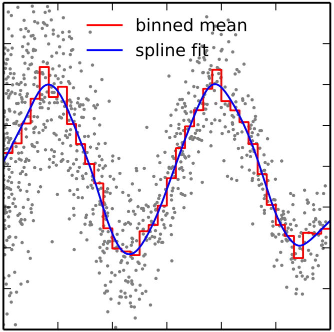
[Galán-de Anta+ 2023]

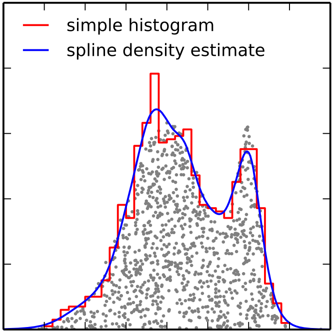
Other features

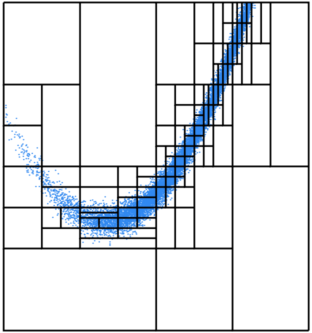
- AGAMA is similar in scope to other galactic-dynamics software packages – GALPY [Bovy 2015] and GALA [Price-Whelan 2017], and provides some interoperability with both (e.g., potentials that can be used in other packages), however, it is typically more computationally efficient in similar tasks.
- AGAMA is *not* an N-body simulation code, but it can be used to set up initial conditions for galaxy simulations, provides external gravitational potential for simulation codes in NEMO and AMUSE frameworks, and assists in the analysis (e.g., extract potential from snapshots, integrate orbits, etc.)
- AGAMA *is* actually a tool for simulating the dynamical evolution of star clusters using the Monte Carlo stellar-dynamical approach (RAGA [Vasiliev 2015]) and the Fokker–Planck method (PHASEFLOW [Vasiliev 2017]).

Various mathematical methods

- spline interpolation (B-spline/cubic/quintic, 1d/2d/3d);
- penalized (i.e., with automatic optimal smoothing) spline fitting and density estimation;
- N-dimensional integration (using `cubature` or `cuba` libraries);
- drawing uniform-weight samples from an arbitrary N-dim probability function (rejection sampling with adaptive domain refinement).







Some technical challenges

- OpenMP parallelization of computationally intensive tasks in the C++ code:
  + efficiency gain with little* extra effort;
  − special measures to ensure independence of result from execution order.
- Python interface uses plain Python C API (no `ctypes`, `boost` or `SWIG`):
  + full control over details, support user-defined Python callback functions;
  − lots of boilerplate code; unwieldy (∼8 000 lines mixture of C & C++)...
- Global Interpreter Lock (GIL) in CPython vs. OpenMP parallelization:
  − Python callbacks cannot be called from multiple threads simultaneously;
  + create native C++ object approximating Python-defined functions ($\rho, \Phi, \dots$)
- Custom installation script `setup.py`:
  + compiles not only the Python extension module, but also C++ executables;
  − nightmare to support various operating systems and Python distributions.
- Deliberately avoid integration with `Astropy`:
  − users need to care about units and coordinate conversion themselves;
  + avoid [sometimes significant] overheads – maximize computational efficiency.

Possible future improvements

- Even with all the documentation and example scripts, the learning curve is quite steep: prepare `Jupyter` notebook tutorials (one is ready, need more).
- Simplify installation by providing pre-built binary packages for `Anaconda`.
- User-defined `Python` callback functions incur a huge performance drop: investigate the possibility of compiling them as `Cython` code and using natively from the `C++` core.
- GPU acceleration of the most performance-critical parts.
- Differentiable programming (e.g., to use with Hamiltonian Monte Carlo).

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;
- Numerical orbit integration;
- 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: spline-based interpolation, 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`