The Journal of Open Source Software

Odisseo: A Differentiable N-body Code for Gradient-Informed Galactic Dynamics

Giuseppe Viterbo [1,2] and **Tobias Buck** [1,2]

1 Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg, Im Neuenheimer Feld 205, D-69120 Heidelberg, Germany **2** Universität Heidelberg, Zentrum für Astronomie, Institut für Theoretische Astrophysik, Albert-Ueberle-Straße 2, D-69120 Heidelberg, Germany

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 22 July 2025
Published: unpublished

Background

N-body simulations, which model the interactions within a system of particles, are a fundamental tool in computational physics with widespread applications [e.g. planetary science, stellar cluster dynamics, cosmology, molecular dynamics]. While Odisseo (Optimized Differentiable Integrator for Stellar Systems Evolution of Orbits) can be used in any context where an N-body simulation is useful, a key motivating application in galactic astrophysics is the study of stellar streams. Stellar streams are the fossilized remnants of dwarf galaxies and globular clusters that have been tidally disrupted by the gravitational potential of their host galaxy. These structures, observed as coherent filaments of stars in the Milky Way and other nearby galaxies, are powerful probes of astrophysics. Their morphology and kinematics encode detailed information about the host galaxy's gravitational field, making them ideal for mapping its shape. Furthermore, studying the properties of streams and their progenitors is key to unraveling the hierarchical assembly history of galaxies.

The standard workflow has involved running computationally expensive simulations, comparing their projected outputs to observational data via summary statistics, and then using statistical methods like MCMC to explore the vast parameter spaces. While successful, this approaches looses information by compressing rich datasets into simple statistics, and struggles with the high-dimensional parameter spaces required by increasingly complex models. With Odisseo we aim to explore how the new paradigm of differentiable simulations and automatic-differentiation can be adopted to challenge many-body problems.

Statement of Need

Inspired by the work of (Alvey et al., 2024) and (Nibauer et al., 2024) on stellar stream differentiable simulators, with Odisseo we intend to offer a general purpose, highly modular, direct N-body simulator package that can be used for detail inference pipeline by taking advantage of the full information present in the phase-space. The main goal is to explore the joint posterior distribution of progenitor and external potential parameters in the context of galactic dynamics. As demonstrated by recent developments, a promising path for inverse modeling techniques lies in leveraging differentiable programming and modern simulation-based inference (SBI) techniques (Holzschuh & Thuerey, 2024).

By providing a fully differentiable N-body simulator built on JAX (Bradbury et al., 2018), Odisseo directly addresses the key bottlenecks of the standard inference pipeline (MCMC). Its differentiability allows for the direct use of simulation gradients to guide parameter inference, enabling a move from inefficient parameter searches to highly efficient, gradient-informed methods.

Odisseo is designed with open-source, community-driven development in mind, providing a robust and accessible foundation that can be extended with new physics models and numerical methods.

Software Design

Odisseo is a Python package written in a purely functional style to integrate seamlessly with the JAX ecosystem. Its design philosophy is to provide a simple, flexible, and powerful tool for inference-focused N-body simulations. Key features include:

- **End-to-End Differentiable**: The entire simulation pipeline is differentiable. The final state of the particles is differentiable with respect to the initial parameters, including initial conditions, total time of integration, particle masses, and parameters of the external potentials.
- **Modularity and Extensibility**: The code is highly modular. The functional design allows for individual components —such as integrators, external potentials, or initial condition generators— to be easily swapped or extended by the user. This facilitates rapid prototyping and model testing.
- **JAX Native and Cross-Platform**: Built entirely on JAX, Odisseo enables end-to-end Just In Time (JIT) compilation for high performance (`jax.jit`), automatic vectorization (`jax.vmap`, or `jax.lax.map` for limited hardware resources) for trivial parallelization, and automatic differentiation (`jax.grad`). This ensures high performance across diverse hardware, including CPUs, GPUs, and TPUs. The use
- **External Potentials**: Odisseo allows for the inclusion of arbitrary external potentials. This is essential for realistically modeling the tidal disruption of satellite systems in a Milky Way-like environment. It can be trivially generalized to physical settings where external potentials are important (e.g. molecular dynamics).
- **Direct acceleration**: The pairwise particle forces are exact, down to a softening length that is used to avoid numerical errors.
- **Unit Conversion**: The conversion between physical and simulation units is handled with a simple `CodeUnits` class that wraps around `astropy` functionality (Astropy Collaboration et al., 2022).

Running a simulation

Four main components are needed to run a simulation :

- **Configuration**: it handles the shapes of the arrays in the simulations (e.g. number of particles, number of time steps) and all the components for which recompilation would be required if changed.
- **Parameters**: it contains the physical parameters with respect to which we can differentiate through the time stepping.
- **Initial conditions**: the initial state of the simulation, it contains the `positions` and `velocities` of all the particles. The `masses` are a separate array with the same length of `initial conditions`.
- **Time integration**: the main function that perform the evolution of the particles state.

Examples on how to set up different problems are presented in the documentation.

Research Impact Statement

Odisseo has already demonstrated concrete research impact through peer-reviewed articles and active use in astrophysical research. The presentation paper was accepted at the Differentiable Systems and Scientific Machine Learning (DiffSciML) workshop at EurIPS2025 (Copenhagen)(Viterbo & Buck, 2025a). Moreover, the code was used to generate a realistic, simulation-based training set for the analysis of the GD-1 stellar stream, for an astrophysical article currently under review at Astronomy & Astrophysics ((Viterbo & Buck, 2025b)).

AI usage disclosure

Generative AI was used to generate documentation and part of the code. The quality and correcteness of AI was assess by mock examples that compares to established dynamical code such as `Gala` (Price-Whelan, 2017), `Galpy`[Bovy:2015] and Galax [Starkman:2024].

Acknowledgements

This project was made possible by funding from the Carl Zeiss Stiftung.

References

Alvey, J., Gerdes, M., & Weniger, C. (2024). *Albatross: A scalable simulation-based inference pipeline for analysing stellar streams in the milky way*. https://doi.org/10.1093/mnras/stad2458

Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., Earl, N., Starkman, N., Bradley, L., Shupe, D. L., Patil, A. A., Corrales, L., Brasseur, C. E., Nöthe, M., Donath, A., Tollerud, E., Morris, B. M., Ginsburg, A., Vaher, E., Weaver, B. A., Tocknell, J., Jamieson, W., … Astropy Project Contributors. (2022). The Astropy Project: Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package. *935*(2), 167. https://doi.org/10.3847/1538-4357/ac7c74

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/jax-ml/jax

Holzschuh, B., & Thuerey, N. (2024). *Flow matching for posterior inference with simulator feedback*. https://arxiv.org/abs/2410.22573

Nibauer, J., Bonaca, A., Spergel, D. N., Price-Whelan, A. M., Greene, J. E., Starkman, N., & Johnston, K. V. (2024). *`StreamSculptor`: Hamiltonian perturbation theory for stellar streams in flexible potentials with differentiable simulations*. https://arxiv.org/abs/2410.21174

Price-Whelan, A. M. (2017). Gala: A Python package for galactic dynamics. *The Journal of Open Source Software*, *2*, 388. https://doi.org/10.21105/joss.00388

Viterbo, G., & Buck, T. (2025a). Differentiable N-body code for Galactic Dynamics – Odisseo. In *arXiv e-prints* (p. arXiv:2511.22468). https://doi.org/10.48550/arXiv.2511.22468

Viterbo, G., & Buck, T. (2025b). The dynamical memory of tidal stellar streams: Joint inference of the Galactic potential and the progenitor of GD-1 with flow matching. In *arXiv e-prints* (p. arXiv:2512.04600). https://doi.org/10.48550/arXiv.2512.04600