# Differentiable Astrophysics

**Philip Mocz***
Center for Computational Astrophysics
Flatiron Institute
New York, NY 10010
pmocz@flatironinstitute.org

## Abstract

We introduce `Jaxion`, a simple and extensible simulation library built on JAX for numerical experiments in astrophysics and scientific machine learning. `Jaxion` enables multiphysics simulations of fuzzy dark matter, stars, and gas, combining spectral methods, particle-mesh, and finite volume schemes. Leveraging JAX's automatic differentiation and hardware acceleration, `Jaxion` provides a flexible platform for rapid prototyping and differentiable simulations. This differentiability allows gradients to be computed through entire simulation pipelines, enabling seamless integration with optimization, inference, and machine learning workflows. As a result, users can efficiently run simulations on GPUs and treat them as black-box differentiable functions for inverse problems or hybrid physics-ML modeling.

## 1 Introduction

Astrophysics has long relied on sophisticated numerical simulation codes to model the complex, multi-scale processes that govern the Universe. Established codes such as ATHENA++ [14], AREPO [13], FLASH [5], and RAMSES [15] have enabled detailed studies of gas dynamics, star formation, and cosmological structure formation. These tools typically employ a combination of grid-based, particle-based, and spectral methods to solve the governing equations of hydrodynamics, gravity, and additional physics.

Despite their success, traditional astrophysics codes are often limited in their ability to interface with modern machine learning (ML) techniques and to support differentiable programming paradigms. As the field moves toward integrating ML for tasks such as parameter inference, model discovery, and hybrid physics-ML modeling, there is a growing need for simulation frameworks that are both flexible and differentiable. The future of astrophysical simulation lies in platforms that can leverage automatic differentiation, hardware acceleration, and seamless integration with ML workflows, enabling new approaches to scientific discovery and data-driven modeling. Recent developments of differentiable astrophysics code for various applications ranging from hydrodynamical simulations to modeling gravitational waves include the works of [7, 9, 16].

In this paper, we introduce `Jaxion`, a differentiable simulation library designed for studying fuzzy dark matter (FDM) coupled to baryons (stars and gas). FDM is a plausible dark matter candidate, modeled as a quantum wave-like field, governed by the Schrödinger-Poisson equations, and exhibits unique phenomena such as solitonic cores and granular interference patterns on kiloparsec scales [8]. It is evolved in time via a spectral method in our code. Gas is modeled with the compressible Euler equations solved with a finite volume method. Stars are evolved according to gravity (collisionless) using a particle-mesh scheme. All three components are coupled through the gravitational potential. Classical codes developed to model such FDM systems include PYULTRALIGHT [4] (spectral method

*Corresponding author

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025.

in Python), GAMER-2 [12] (GPU-accelerated simulations with adaptive mesh refinement), and AREPO [11] (spectral method coupled to a moving Voronoi mesh for the gas).

`Jaxion` is built on JAX [1], and leverages its capabilities for automatic differentiation and just-in-time compilation to deploy code on GPUs and TPUs. Autodiff opens new avenues for scientific discovery, such as gradient-based parameter inference, optimization, and hybrid physics-ML modeling.

This paper is organized as follows: in Section 2 we describe the coupled astrophysical equations that `Jaxion` solves, in Section 3 we describe our second-order numerical method, in Section 4 we describe the concept, advantages, and limitations of end-to-end differentiable physics simulations, in Section 5 we demonstrate the simulation code and application to inverse problems, and in Section 6 we offer concluding remarks.

## 2 Physical Equations

FDM is represented by the wave function, $\psi$, normalized such that its density is $\rho_{\rm dm} = |\psi|^2$. The field has a boson mass of $m$ and is evolved according to the Schrödinger-Poisson equations:

$$i\frac{\partial}{\partial t}\psi = -\frac{\hbar}{2m}\nabla^2\psi + \frac{m}{\hbar}V\psi \tag{1}$$

$$\nabla^2 V = 4\pi G\left(\rho_{\rm tot} - \overline{\rho}_{\rm tot}\right) \tag{2}$$

where the gravitational potential $V$ is sourced by the total density of matter: $\rho_{\rm tot} = \rho_{\rm dm} + \rho_{\rm gas} + \rho_{\rm stars}$, and $\overline{\rho}_{\rm tot}$ is the mean density in our periodic box.

For the gas, we consider the compressible isothermal Euler equations, with density $\rho_{\rm gas} \equiv \rho$, velocity $\mathbf{v}$, and (fixed) sound-speed $c_s$:

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ \rho\mathbf{v}\end{pmatrix} + \nabla\cdot\begin{pmatrix}\rho\mathbf{v}\\ \rho\mathbf{v}\mathbf{v}^T + \rho c_s^2\end{pmatrix} = \begin{pmatrix}0\\ -\rho\nabla V\end{pmatrix} \tag{3}$$

Finally, star particles (position $\mathbf{x}_s$, velocity $\mathbf{v}_s$, mass $m_s$) evolve according to the collisionless Boltzmann equation:

$$\frac{d\mathbf{x}_s}{dt} = \mathbf{v}_s, \quad \frac{d\mathbf{v}_s}{dt} = -\nabla V \tag{4}$$

## 3 Numerical Method

The numerical solver is based on earlier works, re-written in JAX. We employ a second-order ('kick-drift-kick') pseudo-spectral solver for the Schrödinger and Poisson equations in a periodic box [10]. For the hydrodynamics, we use a second-order unsplit Godunov scheme finite volume scheme with local Lax-Friedrichs flux on a uniform mesh, as described in [13]. Finally, for gravitational particles (stars) we employ a second-order particle-mesh method with cloud-in-cell transfer operators [5].

The dark matter, stars, and gas are coupled gravitationally. In each 'kick' half time-step, the gravitational potential is updated by depositing all matter onto the grid to obtain the total density source and solving the Poisson equation (spectrally). Accelerations are then applied to the wavefunction, gas, and particles.

`Jaxion` is open-source and available at `https://github.com/JaxionProject/jaxion`.

## 4 Differentiable Simulations

Fig. 4 illustrates the conceptual parallel between differentiable multiphysics simulation codes and neural networks. In traditional neural network training, model weights are optimized by minimizing a loss function with respect to training data, leveraging gradient information for efficient learning. Similarly, differentiable simulation frameworks enable inverse design, where optimal initial conditions or parameters of a physical system are learned by minimizing an objective function defined on the simulation output. In both scenarios, access to gradients through the computational graph is essential for scalable and efficient optimization, highlighting the transformative role of differentiable programming in both AI and scientific modeling.

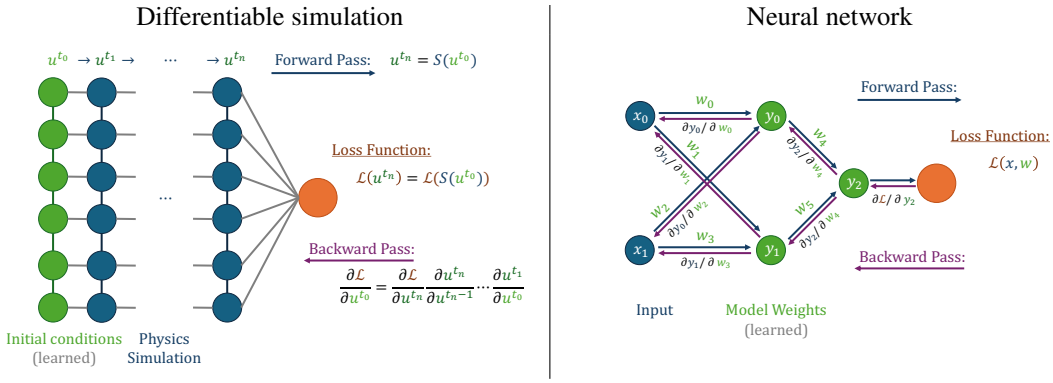


Figure 1: Conceptual parallels between differentiable multiphysics simulation codes and neural networks. Both rely on differentiable frameworks. Finding training model weights in an ML model is analogous to finding optimal initial conditions/parameters of a physical system in an inverse problem.

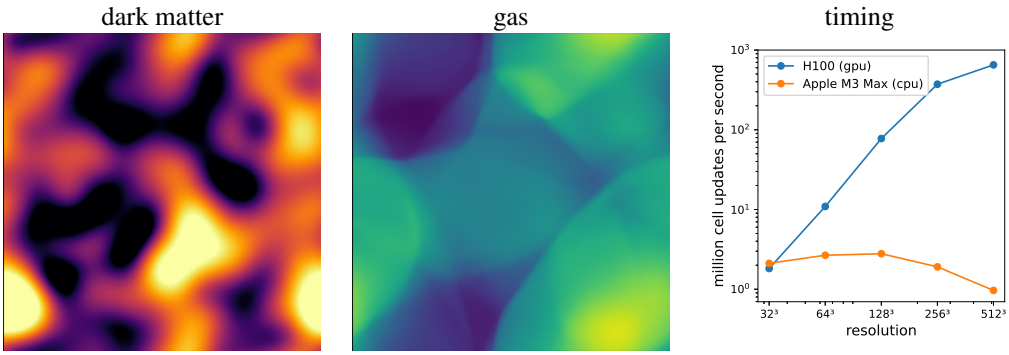


Figure 2: An example of a multiphysics simulation of FDM fluctuations coupling to interstellar gas gravitationally and exciting turbulence. Shown also is the computational scaling of the code as a function of simulation resolution.

The computational cost of gradient computation in differentiable simulations is similar to that of the forward simulation, thanks to reverse-mode automatic differentiation [6]. This efficient scaling enables practical use of gradient-based optimization and inference in scientific workflows without significant overhead. The main price to pay is in the memory overhead needed for the computational graph for automatic differentiation. Checkpointing is a technique to reduce memory usage [6].

## 5 Numerical Examples

We briefly present an example of a multiphysics simulation and computational scaling, as well as an inverse-problem solved with `Jaxion`. More examples and code can be found at the code repository `https://github.com/JaxionProject/jaxion`.

### 5.1 Gas Heating from Fuzzy Dark Matter Fluctuations

Fig. 5.1 demonstrates a multiphysics simulation of FDM and gas. The simulation captures a patch of a dark matter halo + galaxy, where interference fluctuations in the FDM density field gravitationally heat the gas [2], leading to the development of small-scale structure and turbulence. The simulation is initialized with dark matter (FDM boson mass $m = 10^{-22}$ eV) having a fixed velocity dispersion, plus a uniform gas density representative of the interstellar medium. The simulation reveals an interesting phenomenon that FDM can induce turbulence in the gas, which may have implications for star formation and galaxy evolution, and merits further study. A plot demonstrating the timings of running the simulation code is also presented, where it can be seen that running the code on GPUs can be over 100x faster than on CPUs.

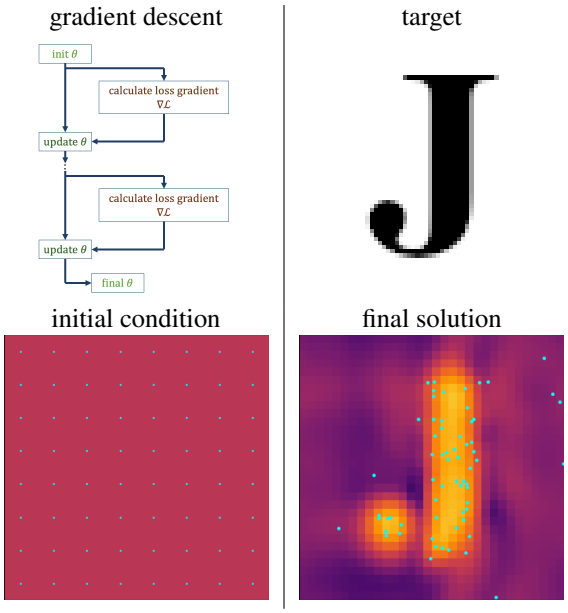


Figure 3: Inverse-problem: in a system with initially uniform dark matter and stars, what initial velocities for the stars will cause the dark matter to evolve into the shape of the letter 'J' (target) in a given amount of time? The bottom right shows the result of a discovered solution after 100 gradient descent-type steps applied to our differentiable physics simulation, found in 511 seconds on an Nvidia H100.

### 5.2 Logo Inverse-Problem

Fig. 5.2 shows an example of an inverse-problem solved with `Jaxion`. In a system with initially uniform dark matter and stars, what initial velocities for the star particles ($64 \times 3$ unknowns) lead to the dark matter ($32^3$ grid) to evolve into the shape of the letter 'J' (projected density) after 300 timesteps? The bottom right shows the result of a discovered solution after 100 Limited-memory BFGS optimization steps (using the `optax` library [3]) applied to our differentiable physics simulation, found in 511 seconds on an Nvidia H100 GPU.

This example showcases the potential of differentiable simulations for solving inverse problems in astrophysics to efficiently navigate the complex landscape of high-dimensional parameter spaces to achieve desired outcomes in the simulation.

## 6 Concluding Remarks

The `Jaxion` code is in its first release phase to introduce a simple, flexible differentiable astrophysics code to the community. Lots of future features are in store, including adding cosmological co-moving factors, sub-grid physics including ML-tuned closure models, multiple axion fields, and scaling of the code to multiple GPUs. Future enhancements may include adaptive mesh refinement (an open area of research for differentiable codes), support for additional physics modules (e.g., magnetohydrodynamics, radiative transfer), improved I/O and visualization tools, integration with probabilistic programming frameworks, and user-friendly APIs for custom model development.

The modern ML Python ecosystem has matured to the point where developing high-performance, feature-rich simulation codes requires 10 times less code than in previous decades. As a result, new ideas can be rapidly prototyped and deployed, accelerating progress in scientific computing. The core of `Jaxion` is only around 1000 lines of code.

`Jaxion` showcases the potential of differentiable programming for astrophysical simulations, combining end-to-end differentiability, hardware acceleration, and modular multiphysics. This open-source platform enables new approaches to scientific discovery in astrophysics. We invite the community to explore and build upon the open-source project.

## References


[1] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

[2] B. V. Church, P. Mocz, and J. P. Ostriker. Heating of Milky Way disc stars by dark matter fluctuations in cold dark matter and fuzzy dark matter paradigms. *MNRAS*, 485(2):2861–2876, May 2019. doi: 10.1093/mnras/stz534.

[3] DeepMind, I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, A. Dedieu, C. Fantacci, J. Godwin, C. Jones, R. Hemsley, T. Hennigan, M. Hessel, S. Hou, S. Kapturowski, T. Keck, I. Kemaev, M. King, M. Kunesch, L. Martens, H. Merzic, V. Mikulik, T. Norman, G. Papamakarios, J. Quan, R. Ring, F. Ruiz, A. Sanchez, L. Sartran, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, M. Stanojević, W. Stokowiec, L. Wang, G. Zhou, and F. Viola. The DeepMind JAX Ecosystem, 2020. URL `http://github.com/google-deepmind`.

[4] F. Edwards, E. Kendall, S. Hotchkiss, and R. Easther. PyUltraLight: a pseudo-spectral solver for ultralight dark matter dynamics. *Journal of Cosmology and Astroparticle Physics*, 2018(10):027, Oct. 2018. doi: 10.1088/1475-7516/2018/10/027.

[5] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo. FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes. *ApJS*, 131(1):273–334, Nov. 2000. doi: 10.1086/317361.

[6] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[7] B. Horowitz and Z. Lukic. Differentiable Cosmological Hydrodynamics for Field-Level Inference and High Dimensional Parameter Constraints. *arXiv e-prints*, art. arXiv:2502.02294, Feb. 2025. doi: 10.48550/arXiv.2502.02294.

[8] L. Hui, J. P. Ostriker, S. Tremaine, and E. Witten. Ultralight scalars as cosmological dark matter. *Phys. Rev. D*, 95(4):043541, Feb. 2017. doi: 10.1103/PhysRevD.95.043541.

[9] D. Lanzieri, F. Lanusse, and J.-L. Starck. Hybrid Physical-Neural ODEs for Fast N-body Simulations. In *Machine Learning for Astrophysics*, page 60, July 2022. doi: 10.48550/arXiv.2207.05509.

[10] P. Mocz, M. Vogelsberger, V. H. Robles, J. Zavala, M. Boylan-Kolchin, A. Fialkov, and L. Hernquist. Galaxy formation with BECDM - I. Turbulence and relaxation of idealized haloes. *MNRAS*, 471(4): 4559–4570, Nov. 2017. doi: 10.1093/mnras/stx1887.

[11] P. Mocz, A. Fialkov, M. Vogelsberger, F. Becerra, M. A. Amin, S. Bose, M. Boylan-Kolchin, P.-H. Chavanis, L. Hernquist, L. Lancaster, F. Marinacci, V. H. Robles, and J. Zavala. First Star-Forming Structures in Fuzzy Cosmic Filaments. *Phys. Rev. Lett.*, 123(14):141301, Oct. 2019. doi: 10.1103/PhysRevLett.123.141301.

[12] H.-Y. Schive, J. A. ZuHone, N. J. Goldbaum, M. J. Turk, M. Gaspari, and C.-Y. Cheng. GAMER-2: a GPU-accelerated adaptive mesh refinement code - accuracy, performance, and scalability. *MNRAS*, 481(4): 4815–4840, Dec. 2018. doi: 10.1093/mnras/sty2586.

[13] V. Springel. E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh. *MNRAS*, 401(2):791–851, Jan. 2010. doi: 10.1111/j.1365-2966.2009.15715.x.

[14] J. M. Stone, K. Tomida, C. J. White, and K. G. Felker. The Athena++ Adaptive Mesh Refinement Framework: Design and Magnetohydrodynamic Solvers. *ApJS*, 249(1):4, July 2020. doi: 10.3847/1538-4365/ab929b.

[15] R. Teyssier. Cosmological hydrodynamics with adaptive mesh refinement. A new high resolution code called RAMSES. *A&A*, 385:337–364, Apr. 2002. doi: 10.1051/0004-6361:20011817.

[16] K. W. K. Wong, M. Isi, and T. D. P. Edwards. Fast Gravitational-wave Parameter Estimation without Compromises. *ApJ*, 958(2):129, Dec. 2023. doi: 10.3847/1538-4357/acf5cd.