Ray-trax: Fast, Time-Dependent, and Differentiable Ray Tracing for On-the-fly Radiative Transfer in Turbulent Astrophysical Flows

Lorenzo Branca[1] **Rune Rost**[1] **Tobias Buck**[1]
[1]Interdisciplinary Center for Scientific Computing, Heidelberg University, Germany
rune.rost@stud.uni-heidelberg.de
lorenzo.branca@iwr.uni-heidelberg.de
tobias.buck@iwr.uni-heidelberg.de

Abstract

Radiative transfer is a key bottleneck in computational astrophysics: it is nonlocal, stiff, and tightly coupled to hydrodynamics. We introduce **Ray-trax**, a GPU-oriented, fully differentiable 3D ray tracer written in JAX that solves the *time-dependent* emission–absorption problem and runs directly on turbulent gas fields produced by hydrodynamic simulations. The method favors the widely used on-the-fly *emission–absorption approximation*, which is state of the art in many production hydro codes when scattering is isotropic. Ray-trax vectorizes across rays and sources, supports arbitrarily many frequency bins without architectural changes, and exposes end-to-end gradients, making it straightforward to couple with differentiable hydro solvers while *preserving differentiability*. We validate against analytical solutions, demonstrate propagation in turbulent media, and perform a simple inverse problem via gradient-based optimization. In practice, the memory footprint scales as $\mathcal{O}(N_{\text{src}}\, N_{\text{cells}})$ while remaining highly efficient on accelerators.

1 Introduction

Radiative feedback shapes star formation, galaxy evolution, and the circumgalactic/intergalactic medium [7, 8, 10]. However, radiative transfer (RT) is often the step that limits resolution, cadence, or physics fidelity in large-scale simulations: it introduces global couplings, adds a fast characteristic speed (c), and requires angular and spectral resolution. For many workflows, especially *on-the-fly* coupling to hydrodynamics, an emission–absorption model (isotropic scattering) is the de facto standard [12]; it captures photoheating and attenuation with a clean computational footprint and predictable scaling [6]. This paper presents **Ray-trax**, a compact JAX [1] ray-tracing implementation designed around four goals: (i) *time dependence*, solving the initial-value radiative-transport problem with finite light-travel horizons; (ii) *GPU throughput*, by fully vectorizing the marching loop over rays and sources; (iii) *frequency-bin agnosticism*, treating the frequency dimension as a batch axis so users can add as many radiation bins as needed without changing code paths; and (iv) *differentiability*, enabling gradient-based inference and, crucially, the option to *couple with differentiable hydro* while maintaining end-to-end gradients `https://github.com/lorenzobranca/Ray-trax.git`.

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

2 Method and code structure

We model time-dependent, monochromatic transport of specific intensity $I(\mathbf{x}, \hat{\mathbf{n}}, t)$ via

$$\frac{1}{c}\frac{\partial I}{\partial t} + \hat{\mathbf{n}}\cdot\nabla I \;=\; -\kappa(\mathbf{x})\, I \;+\; j(\mathbf{x}), \tag{1}$$

with space-only coefficients κ and j defined on a regular 3D grid. Along characteristics we march in *space* with step Δs (so one time step of size Δt corresponds to a propagation horizon $c\,\Delta t$). The code uses a semi-analytic/explicit update per substep $k \to k{+}1$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \hat{\mathbf{n}}\Delta s, \qquad \tau_{k+1} = \tau_k + \kappa(\mathbf{x}_k)\,\Delta s, \qquad I_{k+1} = I_k\, e^{-\kappa(\mathbf{x}_k)\Delta s} + j(\mathbf{x}_k)\, e^{-\tau_k}\,\Delta s, \tag{2}$$

i.e., exact attenuation over the step and a first-order quadrature for emission. The number of substeps is $N_s = \lceil c\,\Delta t/\Delta s \rceil$ (or a user-specified cap). Despite the apparent simplicity of the approximation (isotropic scattering), it is state of the art in numerical simulations [5, 9, 2].
We trace N_Ω nearly uniform directions on $\mathbb{S}^2$ using a Fibonacci (golden-angle) lattice well suited to vectorization [3]. The per-ray contributions are summed and scaled by $4\pi/N_\Omega$ to approximate the solid-angle integral.
At each substep we read j via trilinear interpolation [4] and (currently) sample κ with nearest-neighbor interpolation at the cell index used for the previous position, then deposit the *post-attenuation* intensity I_{k+1} back to the grid using the same trilinear weights. This makes the discrete operator smooth in the voxel values and avoids scatter/gather mismatch. Indexing is *clamped* to the domain, so rays exiting the box deposit to the nearest boundary voxels.
The single-source kernel advances the field over one Δt given `(j_map, kappa_map, source_pos, num_rays, step_size, radiation_velocity, time_step)`. Multiple sources are supported by a simple loop-and-sum wrapper.

Vectorization, memory, and speed. Rays are independent conditioned on the medium, so we use `vmap` over directions; the marching loop itself is written with unrolled `lax.fori_loop` and JIT compiled. On multi-GPU nodes we optionally shard directions across devices with `NamedSharding` and perform a final on-device sum per shard before reducing across shards; the code requires N_Ω to be divisible by the device count.
The work scales as $\mathcal{O}(N_{\text{src}}\, N_\Omega\, N_s)$. In the current implementation, each ray accumulates into a grid and these are reduced at the end, so the transient memory is $\mathcal{O}(N_\Omega\, N_{\text{cells}})$ (amortized by device sharding); the returned field is one grid per time step (or the sum over sources in the multi-source routine).
On a 128^3 grid, tracing $4{,}096$ rays from $2{,}097$ sources across the full domain completes in $4.49\,\text{s}$ (wall clock) on a single NVIDIA H200 GPU.

Frequency-bin agnosticism. Multi-frequency RT often forces architectural choices. In Ray-trax the number of bins is a batch hyperparameter: adding bins simply increases parallel work. No control-flow branches, no new kernels, and no changes to the differentiation rules are required.

3 Benchmarks in uniform media and turbulent fields

Analytical validation. For constant κ and point-like sources (represented numerically as narrow Gaussians) the reference intensity is

$$J_{\text{ref}}(\mathbf{x}) \;=\; \sum_{i=1}^{N_{\text{src}}} \frac{L_i}{4\pi r_i^2}\, e^{-\kappa r_i}, \qquad r_i = \|\mathbf{x} - \mathbf{x}_i\|. \tag{3}$$

We compare $\log_{10} J$ slices (numeric vs. analytical) and relative-error maps, excluding the central voxel and a one-voxel border from metrics. Convergence is monotonic in both the spatial step Δs and the number of directions N_Ω, as expected for first-order ray marching with uniform angular sampling. The average relative error with respect to the analytical solution is 3.16%. A radial cut through a source validates the $1/r^2$ scaling and the exponential attenuation, shown in the top panel of Fig. 1.

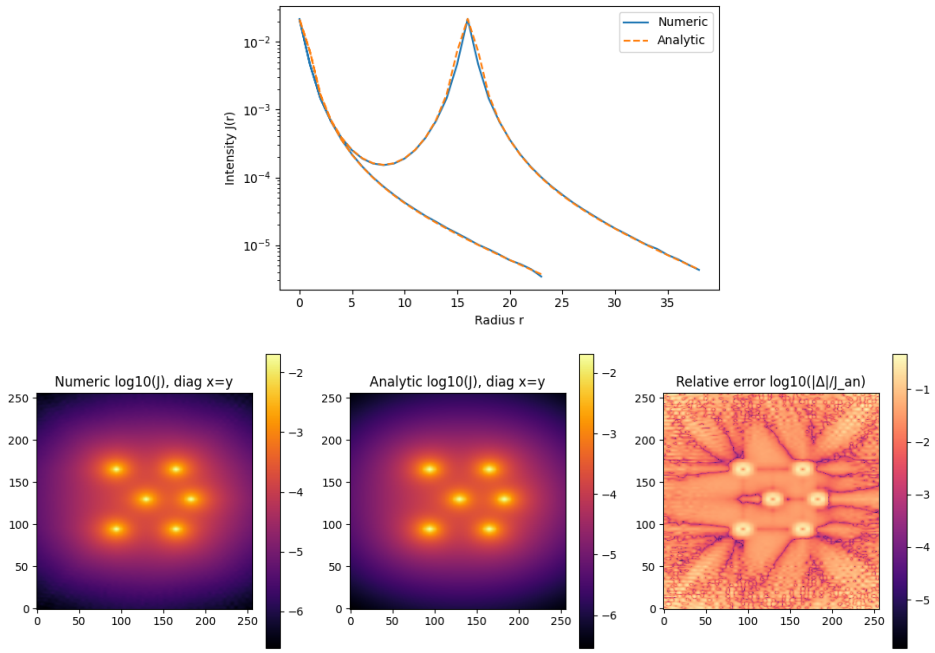

Figure 1: Analytical benchmark in a uniform medium with multiple point sources. **Top:** Specific intensity I along a line of sight through the domain center, shown for both $\pm$ directions; the sightline intersects a second source in one of the directions. **Bottom:** Diagonal slice of I across the volume (left to right: numerical solution, analytical reference, and $\log_{10}$ residuals).

Turbulent snapshots from hydro. We apply `Ray-trax` *directly* to turbulent gas fields from hydrodynamic simulations. The opacity is prescribed as $\kappa(\mathbf{x}) \propto \rho(\mathbf{x})$, where ρ is the turbulent density computed with the hydro code `jf1uid` [11]. Radiation sources are placed at the top 0.1% density peaks. This setup is intended to mimic a typical turbulent, star-forming cloud.

Figure 2 shows six x–y slices of $\log_{10} I$ evolving over time: the morphology displays shadowing, porous channels, and illuminated bubbles consistent with the underlying flow structures. For time-resolved visualizations we increase the light-travel horizon step by step to reveal the front progression; for strict time integration we fix Δt and iterate.

On-the-fly suitability. In many production hydro codes, the **emission–absorption** approximation is the *state of the art* for on-the-fly RT: it captures attenuation and local heating without introducing angular coupling from scattering. `Ray-trax` is tailored to this use case: the operator is parallel over rays, frequency bins are batched, and the memory footprint is predictable and linear in N_{src} and N_{cells}. This makes it practical to insert RT substeps between hydro updates without derailing wall-clock budgets.

4 Differentiability: inversion and coupling

Gradient-based inversion. To illustrate end-to-end gradients we recover a scalar source amplitude A with a fixed spatial template $e_0(\mathbf{x})$ from a reference field I_{ref}. We minimize either a plain MSE or a relative MSE using gradients $\partial\mathcal{L}/\partial A$ from `jacfwd`. The 1D loss landscape $A \mapsto \mathcal{L}(A)$ is convex; the gradient solution matches the corresponding closed-form minimizer (under the chosen weighting). This toy example readily extends to a small vector of parameters (e.g., a global κ scale and a few amplitudes), and to multi-bin fits where each bin contributes an additive term in the loss.

Coupling to differentiable hydro. Because `Ray-trax` is built from smooth interpolation/deposition primitives and JAX control flow, its Jacobians with respect to (j, κ) are well defined. When (j, κ) comes from a differentiable hydrodynamics model (e.g., density, temperature, or ioniza-

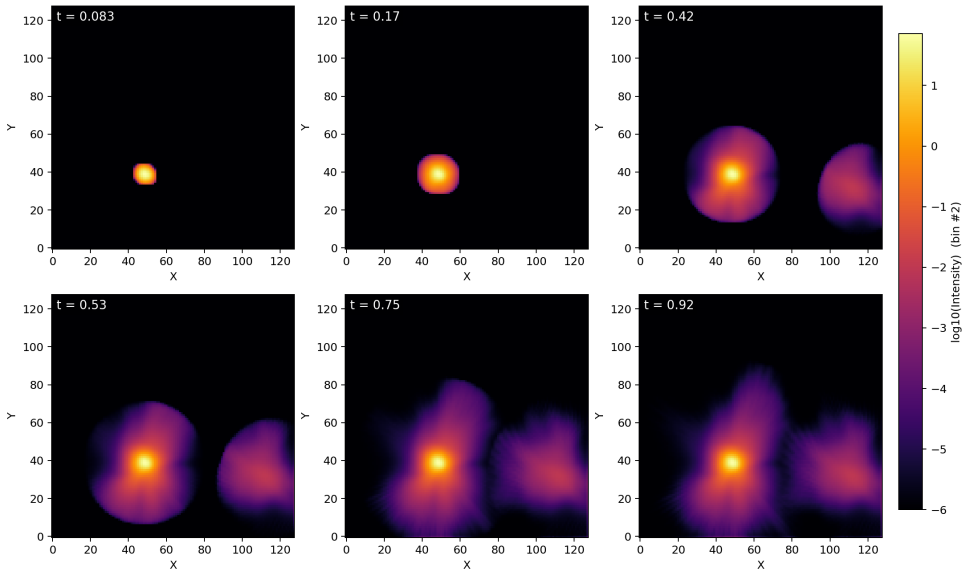

Figure 2: Time-resolved propagation (from top left to bottom right) of specific intensity I through a turbulent opacity field $\kappa(\mathbf{x})$ extracted from a hydrodynamic snapshot. The frequency bin is chosen to highlight shadowing. Time t is in unit of total simulation time.

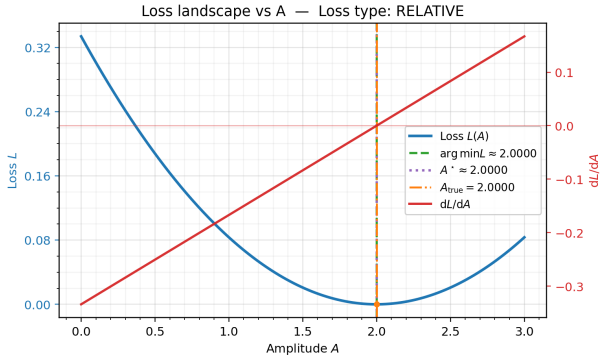

Figure 3: End-to-end differentiability enables straightforward parameter recovery.

tion fractions produced by a differentiable PDE integrator; see [11]), *the composed system remains differentiable*. This opens the door to physics-constrained learning, gradient-based calibration of sub-grid models, and end-to-end system identification where radiative observables inform hydrodynamic parameters.

5 Discussion and conclusions

We presented `Ray-trax`, a fast, *time-dependent*, and fully differentiable ray tracer for on-the-fly emission–absorption RT on GPUs. It operates directly on turbulent hydro snapshots, supports arbitrarily many frequency bins without code changes, and, despite a memory cost that scales as $\mathcal{O}(N_{\text{src}}\, N_{\text{cells}})$ when per-source fields are retained, remains highly efficient in practice due to full vectorization and sharding. Analytical tests verify accuracy and convergence; turbulent applications demonstrate realism; and differentiability enables both inverse problems and seamless coupling to differentiable hydro while preserving gradients. We view `Ray-trax` as a minimal, composable building block for modern astrophysical pipelines where RT can no longer be the bottleneck. The code is publicly available at `https://github.com/lorenzobranca/Ray-trax.git`.

6 Acknowledgments

This work is funded by the Carl-Zeiss-Stiftung through the NEXUS program. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2181/1 - 390900948 (the Heidelberg STRUCTURES Excellence Cluster). We acknowledge the usage of the AI-clusters Tom and Jerry funded by the Field of Focus 2 of Heidelberg University.

References

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[2] D. Decataldo, A. Lupi, A. Ferrara, A. Pallottini, and M. Fumagalli. Shaping the structure of a GMC with radiation and winds. *Monthly Notice of the Royal Astronomical Society*, 497(4):4718–4732, October 2020.

[3] Álvaro González. Measurement of areas on a sphere using Fibonacci and latitude–longitude lattices. *Mathematical Geosciences*, 42(1):49–64, January 2010.

[4] Herman Hansson Söderlund, Alex Evans, and Tomas Akenine-Möller. Ray tracing of signed distance function grids. *Journal of Computer Graphics Techniques (JCGT)*, 11(3):94–113, 2022.

[5] Rahul Kannan, Mark Vogelsberger, Federico Marinacci, Ryan McKinnon, Rüdiger Pakmor, and Volker Springel. AREPO-RT: radiation hydrodynamics on a moving mesh. *Monthly Notice of the Royal Astronomical Society*, 485(1):117–149, May 2019.

[6] D. Mihalas and B. W. Mihalas. *Foundations of radiation hydrodynamics*. 1984.

[7] Masahiro Nagashima, Naoteru Gouda, and Norimasa Sugiura. Effects of the UV background radiation on galaxy formation. *Monthly Notice of the Royal Astronomical Society*, 305(2):449–456, April 1999.

[8] Aura Obreja, Fabrizio Arrigoni Battaia, Andrea V. Macciò, and Tobias Buck. AGN radiation imprints on the circumgalactic medium of massive galaxies. *Monthly Notice of the Royal Astronomical Society*, 527(3):8078–8102, January 2024.

[9] J. Rosdahl, J. Blaizot, D. Aubert, T. Stranex, and R. Teyssier. RAMSES-RT: radiation hydrodynamics in the cosmological context. *Monthly Notice of the Royal Astronomical Society*, 436(3):2188–2231, December 2013.

[10] Piyush Sharda and Shyam H. Menon. Population III star formation in the presence of turbulence, magnetic fields, and ionizing radiation feedback. *Monthly Notice of the Royal astronomical Society*, 540(2):1745–1764, June 2025.

[11] Leonard Storcks and Tobias Buck. Differentiable Conservative Radially Symmetric Fluid Simulations and Stellar Winds – jf1uids. *arXiv e-prints*, page arXiv:2410.23093, October 2024.

[12] Richard Wünsch. Radiation transport methods in star formation simulations. *Frontiers in Astronomy and Space Sciences*, 11:1346812, March 2024.

A Notation

We summarize here the main symbols and terms used throughout the paper.

Geometry and coordinates.

$\mathbf{x} = (x, y, z)$ Spatial position on a regular 3D Cartesian grid.

$\hat{\mathbf{n}}$ Unit vector specifying a propagation direction on the unit sphere $\mathbb{S}^2$.

$r_i = |\mathbf{x} - \mathbf{x}_i|$ Distance between position $\mathbf{x}$ and source position $\mathbf{x}_i$.

Radiation field.

$I(\mathbf{x}, \hat{\mathbf{n}}, t)$ Specific intensity (monochromatic), i.e., radiant energy per unit time, area, solid angle, and frequency, at position $\mathbf{x}$, in direction $\hat{\mathbf{n}}$, and time t.

$J(\mathbf{x}, t)$ Angle-integrated (or accumulated) intensity at position $\mathbf{x}$ and time t, e.g., $J = \int I, \mathrm{d}\Omega$ or its discrete approximation via the sum over sampled directions.

$J_{\mathrm{ref}}(\mathbf{x})$ Reference (analytical) solution for J used in benchmarks.

I_{ref} Reference intensity field used in inversion experiments.

Material properties and sources.

$\kappa(\mathbf{x})$ Absorption coefficient (opacity) at position $\mathbf{x}$.

$j(\mathbf{x})$ Emissivity at position $\mathbf{x}$ (source term in the emission–absorption equation).

$\rho(\mathbf{x})$ Gas density field; in turbulent tests we set $\kappa(\mathbf{x}) \propto \rho(\mathbf{x})$.

L_i Luminosity (normalization) of the i-th point-like source in the analytical benchmark.

$N_{\mathbf{src}}$ Number of radiation sources.

Transport equation and marching scheme.

c Signal speed in the transport equation (speed of light or chosen radiation propagation speed).

t Physical time.

Δt Time-step size for the macroscopic evolution or light-travel horizon.

Δs Spatial step along a ray; we typically choose $\Delta s = c, \Delta t / N_s$ or an equivalent prescription.

N_s Number of substeps along each ray segment, $N_s = \lceil c, \Delta t / \Delta s \rceil$ (or user-specified).

k Substep index in the marching scheme.

$\mathbf{x}k$ Ray position at substep k.

I_k Specific intensity at substep k along a ray.

τ_k Optical depth accumulated up to substep k, $\tau k + 1 = \tau_k + \kappa(\mathbf{x}_k), \Delta s$.

Angular discretization.

N_Ω Number of discrete propagation directions sampled on $\mathbb{S}^2$ (Fibonacci / golden-angle lattice).

$4\pi / N_\Omega$ Solid angle weight used to approximate the integral over directions by a finite sum.

Discretization and complexity.

$N_{\mathbf{cells}}$ Total number of grid cells in the 3D domain.

$\mathcal{O}(\cdot)$ Big-O notation for asymptotic computational or memory complexity.

$\mathcal{O}(N_{\mathbf{src}}, N_\Omega, N_s)$ Work per time step: sources $\times$ directions $\times$ substeps.

$\mathcal{O}(N_{\mathbf{src}}, N_{\mathbf{cells}})$ Memory scaling when storing per-source fields before summation.

Inverse problems and differentiability.

A Scalar amplitude parameter (e.g., global scaling of a fixed emissivity template) recovered in gradient-based inversion.

$e_0(\mathbf{x})$ Fixed emissivity template used in inversion experiments; the effective emissivity is $A, e_0(\mathbf{x})$.

$\mathcal{L}$ Loss function used for calibration or inversion (e.g., mean squared error or relative error between I/J and a reference).

$\partial \mathcal{L} / \partial A$ Gradient of the loss with respect to A, obtained via automatic differentiation.

Implementation-level symbols (informal).

`j_map` Discrete grid representation of $j(\mathbf{x})$.

`kappa_map` Discrete grid representation of $\kappa(\mathbf{x})$.

`source_pos` Source position(s) in grid coordinates.

`num_rays` Implementation parameter corresponding to N_Ω.

`step_size` Implementation parameter corresponding to Δs.

`radiation_velocity` Code parameter corresponding to c.

`time_step` Code parameter corresponding to Δt.

This notation table is intended to make the manuscript accessible to readers who are not specialists in radiative transfer while remaining consistent with standard usage in the field.