aurel: A Python package for automatic relativistic calculations

Robyn L. Munoz [1], **Christian T. Byrnes** [1], **and Will J. Roper** [1]

1 Department of Physics and Astronomy, University of Sussex, Brighton, BN1 9QH, United Kingdom

Software

- Repository

Published: unpublished

Summary

`aurel` is an open-source Python package designed to *au*tomatically calculate *rel*ativistic quantities. It uses an efficient, flexible and user-friendly caching and dependency-tracking system, ideal for managing the highly nonlinear nature of general relativity. The package supports both symbolic and numerical calculations. The symbolic part extends `SymPy` with additional tensorial calculations. The numerical part computes a wide range of tensorial quantities, such as curvature, matter kinematics and much more, directly from any spacetime and matter data arrays using finite-difference methods. Inputs can be either generated from analytical expressions or imported from Numerical Relativity (NR) simulations. For users of `Einstein Toolkit`, `aurel` also provides utilities to load 3D data generated with `Carpet`. Given the increasing use of NR, `aurel` offers a timely post-processing tool to support the popularisation of this field.

Statement of need

General relativity describes matter as moving according to how distances shrink or expand; likewise, the intervals of space and time evolve depending on the distribution of matter. Handling this dynamic "mesh" of distances and times requires elaborate tensor algebra that, in some cases, can only be managed with symbolic or numerical tools. Naturally, NR has become essential for modern astrophysics, cosmology, and gravitational physics, most notably in the modelling of gravitational-wave signals.

While established computational frameworks focus on solving and evolving Einstein's field equations, with specific key diagnostics, they leave calculations of the remaining analysis to the discretion of the researchers. Newcomers to the field then face a substantial overhead until they develop their own personal post-processing codes. Established researchers also face the tedious task of handling intermediary variables and indices when calculating new quantities. The field then suffers from this error-prone, time-consuming process and would benefit from an accessible, open-source, standardised framework to automate these steps.

We therefore present `aurel`, an open-source Python package designed to streamline relativistic calculations. It is hosted on GitHub and is available on PyPI. The documentation is available through GitHub Pages.

State of the field

When looking for general relativity Python packages, there are a number of tools that provide symbolic calculations (Czaja, n.d.; Gourgoulhon, Bejger, and Mancini 2015; Wittig and Grover 2017; Bapat et al. 2020; Martín and Sureda 2022; Della Monica 2025; Shoshany 2025; Hackstein and Hackmann 2025). Or, one may also consider computer algebra systems

(Maplesoft 2025; Wolfram Research 2025; Martín-García et al. 2025). However, when non-linearities become too complex for symbolic packages, NR is used instead.

`Einstein Toolkit` (Löffler et al. 2012; Rizzo et al. 2025) is a large community-driven software whose tools enable the evolution of Einstein's field equations. Diagnostic and further analysis calculations are typically performed on the fly, during simulations. To study the outputs of such simulations, provided by `Carpet`, there are Python reading packages available (Kastaun, n.d.; Bozzola 2021; Radice, n.d.; Ferguson et al. 2025). These extra calculations can slow down the simulation of the spacetime evolution, and if certain relativistic quantities are not available in `Einstein Toolkit`, or in one of the post-processing packages, then the user needs to code that up themselves.

There are a number of other well-established NR codes (Wright 2018; Barrera-Hinojosa and Li 2020; Andrade et al. 2021; Zhang et al. 2025; Palenzuela et al. 2025) that also have their own diagnostic tools. However, these are typically built-in, so going from one code to another, to benchmark or to use their different types of applications, requires learning the ecosystem of each.

To improve the community's versatility and limit the repeated implementation of error-prone calculations, there is a motivation to provide packages for computing relativistic quantities in an NR-code-agnostic way. Especially in the post-processing sense, where all calculations are done from a given NR spacetime and matter solution. A couple of notable packages (Pook-Kolb et al. 2019; Grasso et al. 2021) focus on ray tracing, or apparent-horizon finding, which are currently beyond the scope of `aurel`. While others have more overlap (Munoz and Bruni 2023; Cranganore et al. 2025) in calculating curvature terms, they differ in scope and workflow.

Here, `aurel` innovates in its automatic design, which is easily extendable and provides flexibility and robustness with a large and ever-growing catalogue of relativistic quantities. A precursor to this package was `EBWeyl` (Munoz and Bruni 2023), as it provided calculations of gravito-electromagnetic contributions from base spacetime and matter quantities. `aurel` now has a completely different structure (relying on the automatic dependency resolution), provides calculations of many more terms, over time, and has entirely new features as described in the following section.

Software Design

`aurel` provides an intuitive interface for the automatic calculation of general relativistic quantities, either symbolically (with `AurelCoreSymbolic`, built on `SymPy` (Meurer et al. 2017)) or numerically (with `AurelCore`, which heavily utilises `numpy.einsum` (Harris et al. 2020) for efficient operations on array data structures).

Both require base quantities such as the spacetime coordinates or the parameters of the Cartesian numerical grid, as well as the spacetime and matter distributions (the Minkowski vacuum is otherwise assumed). These inputs can either come from analytical expressions, with a couple of built-in solutions available, or from output data from any NR simulations; they just need to be passed as numpy arrays.

Specifically, for simulations run with `Carpet` in the `Einstein Toolkit`, the `reading` module provides helper functions to load and organise the 3D data. These can read the parameter file, summarise available iterations and variables, and handle data separated across restarts, chunks, or refinement levels for normal `Carpet` data files or checkpoint files. To speed up repeated data reading, `read_data` can also split the data per iteration, instead of per variable.

Then, once input data is provided, users can directly request a wide range of relativistic quantities, including: spacetime; matter (Eulerian, Lagrangian, or conserved); BSSNOK formulation; constraints; fluid covariant kinematics; null ray expansion; 3- and 4-dimensional curvature; gravito-electromagnetism; Weyl scalars and invariants (including gravitational waves).

To see a full list of available quantities, see: descriptions. Tools are also provided for spatial and spacetime covariant derivatives and Lie derivatives. All spatial derivatives are computed by the `FiniteDifference` class that provides 2nd, 4th, 6th and 8th order schemes, using periodic, symmetric or one-sided boundary conditions.

Automatic Computational Pathway

The `aurel` automatic process composes a computational pathway at runtime to evaluate the requested quantities. This is implemented through a lazy-evaluation memoised property pattern, where each quantity is defined as a method of the core class that may depend on other quantities. This design has been chosen for its flexibility and accessibility while remaining robust under future extensions.

Quantities are requested via a user-friendly dictionary-style access, e.g. `rel["s_RicciS"]`, which triggers the lazy memoised check to see if this is already cached. If yes, then the result is directly returned. If not, then the corresponding method is called, which recursively triggers the calculation of dependencies (e.g. `rel["s_Ricci_down3"]`). This continues until the requested quantities can be calculated and so returned.

To avoid redundant computations, each result is cached, which builds up a cache memory that needs to be efficiently managed. So, inspired by Python's garbage collection, `aurel` uses an intelligent eviction policy that tracks memory footprint, evaluation counts, and last-access times. When the configurable thresholds are exceeded, the older and heavier cached quantities are removed, while safeguarding protected base quantities. Throughout this process, `aurel` keeps the user informed on progress by providing verbose updates on the computation and caching workflow.

Time dependence

All calculations within the `AurelCore` class are evaluated at a single fixed time, corresponding to one slice in time, so that individual time steps can be treated independently. For multiple time steps, an `AurelCore` object needs to be created and the requested quantities collected for each.

To streamline this process, `aurel` provides the `over_time` function to do exactly this, and also compute summary statistics over the grid domain (e.g., max/min) at each time step. By design, it is easily extensible, so `over_time` also accepts custom functions of new relativistic quantities and summary statistics. This makes `aurel` versatile, supporting an infinite number of ways to view a problem and develop diagnostic tools.

Research Impact Statement

`aurel` is a specialist tool for general relativity researchers and streamlines numerical relativists' post-processing workflow. Through conference interactions and collaborations involving the authors, this package has gradually been disseminated to individual researchers who appreciate the effortless integration, satisfying dependency resolution and substantial reduction to post-processing overhead. Indeed, in ongoing studies involving NR simulations of primordial black hole formation, `aurel` has increased capacity and redirected repetitive and error-prone development efforts towards exploring a broader range of simulated scenarios. Additionally, for master students, the straightforward and transparent design has provided an easy gateway for them to analyse NR simulations and so quickly get results within the duration of their projects. Going forward, awareness of this code will build upon publication, reaching a wider audience and supporting the popularisation of NR.

AI usage disclosure

GitHub Copilot Claude Sonnet 4 was used for the development and documentation of this package. Autocompletion suggestions were accepted via the VSCode Copilot plugin, and upon the developer's request, edits and code snippets were generated via the large language model's user interface. The most significant AI contributions came in drafting the docstrings and scaffolding the test suite, both of which are essential for the accessibility and robustness of this package. Each and every suggestion or contribution was meticulously reviewed and adjusted before being included by the authors, who made all core design decisions and innovated the original structural concept. Finally, this paper was prepared without the use of generative language models, solely with grammar checkers.

Acknowledgements

We thank Nat Kemp for being one of the first testers of `aurel`. We thank Ian Hawke for support and suggestions.

RM and WR are supported by an STFC grant ST/X001040/1. CB is supported by STFC grants ST/X001040/1 and ST/X000796/1.

Andrade, Tomas, Llibert Areste Salo, Josu C. Aurrekoetxea, Jamie Bamber, Katy Clough, Robin Croft, Eloy de Jong, et al. 2021. "GRChombo: An Adaptable Numerical Relativity Code for Fundamental Physics." *Journal of Open Source Software* 6 (68): 3703. https://doi.org/10.21105/joss.03703.

Bapat, Shreyas, Ritwik Saha, Bhavya Bhatt, Shilpi Jain, Akshita Jain, Sofía Ortín Vela, Priyanshu Khandelwal, et al. 2020. "EinsteinPy: A Community Python Package for General Relativity." https://arxiv.org/abs/2005.11288.

Barrera-Hinojosa, Cristian, and Baojiu Li. 2020. "GRAMSES: A New Route to General Relativistic N-Body Simulations in Cosmology. Part i. Methodology and Code Description." *JCAP* 01: 007. https://doi.org/10.1088/1475-7516/2020/01/007.

Bozzola, Gabriele. 2021. "Kuibit: Analyzing Einstein Toolkit Simulations with Python." *Journal of Open Source Software* 6 (60): 3099. https://doi.org/10.21105/joss.03099.

Cranganore, Sandeep Suresh, Andrei Bodnar, Arturs Berzins, and Johannes Brandstetter. 2025. "Einstein Fields: A Neural Perspective to Computational General Relativity." https://arxiv.org/abs/2507.11589.

Czaja, Wojciech. n.d. "GraviPy, Tensor Calculus Package for General Relativity." https://pypi.python.org/pypi/GraviPy.

Della Monica, Riccardo. 2025. "PyGRO: A Python Integrator for General Relativistic Orbits." *Astronomy &Amp; Astrophysics* 698 (June): A193. https://doi.org/10.1051/0004-6361/202554300.

Ferguson, Deborah, Surendra Anne, Miguel Gracia-Linares, Hector Iglesias, Aasim Jan, Erick Martinez, Lu Lu, et al. 2025. "Mayawaves." Zenodo. https://doi.org/10.5281/zenodo.17981058.

Gourgoulhon, Eric, Michal Bejger, and Marco Mancini. 2015. "Tensor Calculus with Open-Source Software: The SageManifolds Project." *Journal of Physics: Conference Series* 600 (1): 012002. https://doi.org/10.1088/1742-6596/600/1/012002.

Grasso, Michele, Eleonora Villa, Mikołaj Korzyński, and Sabino Matarrese. 2021. "Isolating Nonlinearities of Light Propagation in Inhomogeneous Cosmologies." *Phys. Rev. D* 104 (4): 043508. https://doi.org/10.1103/PhysRevD.104.043508.

Hackstein, Jan P., and Eva Hackmann. 2025. "GREOPy: A Python Package for Solving the Emitter-Observer Problem in General Relativity." *Journal of Open Source Software* 10 (112): 8765. https://doi.org/10.21105/joss.08765.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array Programming with NumPy." *Nature* 585 (7825): 357–62. https://doi.org/10.1038/s41586-020-2649-2.

Kastaun, Wolfgang. n.d. "PostCactus." GitHub. https://github.com/wokast/PyCactus.
Löffler, Frank et al. 2012. "The Einstein Toolkit: A Community Computational Infrastructure for Relativistic Astrophysics." *Class. Quant. Grav.* 29: 115001. https://doi.org/10.1088/0264-9381/29/11/115001.
Maplesoft, a division of Waterloo Maple Inc. 2025. "Maple, Version 2025.2." https://www.maplesoft.com/products/maple/.
Martín, M. San, and J. Sureda. 2022. "Pytearcat: PYthon TEnsor AlgebRa calCulATor a Python Package for General Relativity and Tensor Calculus." *Astronomy and Computing*, 100572. https://doi.org/10.1016/j.ascom.2022.100572.
Martín-García, José M. et al. 2025. "xAct: Efficient Tensor Computer Algebra for the Wolfram Language." https://www.xact.es/.
Meurer, Aaron, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, et al. 2017. "SymPy: Symbolic Computing in Python." *PeerJ Computer Science* 3: e103. https://doi.org/10.7717/peerj-cs.103.
Munoz, Robyn L., and Marco Bruni. 2023. "EBWeyl: A Code to Invariantly Characterize Numerical Spacetimes." *Classical and Quantum Gravity* 40 (13): 135010. https://doi.org/10.1088/1361-6382/acd6cf.
Palenzuela, Carlos, Miguel Bezares, Steven Liebling, Federico Schianchi, Julio Fernando Abalos, Ricard Aguilera-Miret, Carles Bona, et al. 2025. "MHDuet : A High-Order General Relativistic Radiation MHD Code for CPU and GPU Architectures." https://arxiv.org/abs/2510.13965.
Pook-Kolb, Daniel, Ofek Birnholtz, Badri Krishnan, and Erik Schnetter. 2019. "Existence and Stability of Marginally Trapped Surfaces in Black-Hole Spacetimes." *Phys. Rev. D* 99 (6): 064005. https://doi.org/10.1103/PhysRevD.99.064005.
Radice, David. n.d. "Scidata." Bitbucket. https://bitbucket.org/dradice/scidata/src/master/.
Rizzo, Maxwell, Roland Haas, Steven R. Brandt, Zachariah Etienne, Deborah Ferguson, Lucas Timotheo Sanches, Bing-Jyun Tsao, et al. 2025. "The Einstein Toolkit." Zenodo. https://doi.org/10.5281/zenodo.15520463.
Shoshany, Barak. 2025. "OGRePy: An Object-Oriented General Relativity Package for Python." *Journal of Open Research Software* 13. https://doi.org/10.5334/jors.558.
Wittig, Alexander Nicolaus, and Jai Grover. 2017. "PyHole: General Relativity Ray Tracing and Analysis Tool." https://eprints.soton.ac.uk/453123/.
Wolfram Research, Inc. 2025. "Mathematica, Version 14.3." https://www.wolfram.com/mathematica.
Wright, Alex. 2018. "AlexJamesWright/METHOD: Initial Public Release." Zenodo. https://doi.org/10.5281/zenodo.1404697.
Zhang, Han, Baojiu Li, Tobias Weinzierl, and Cristian Barrera-Hinojosa. 2025. "ExaGRyPE: Numerical General Relativity Solvers Based Upon the Hyperbolic PDEs Solver Engine ExaHyPE." *Comput. Phys. Commun.* 307: 109435. https://doi.org/10.1016/j.cpc.2024.109435.