The Journal of Open Source Software

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

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary/Abstract

`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 simulations. For users of `Einstein Toolkit`, `aurel` also provides utilities to load 3D data generated with `Carpet`. Given the increasing use of numerical relativity, `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 time and space 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, numerical relativity has become an essential tool in 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.

Therefore, we 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.

Features

`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 numerical relativity simulations.

Specifically, for simulations run with `Carpet` in the `Einstein Toolkit`, the `reading` module provides helper functions to load data in that specific format. These can read the parameter file, create summarising files of the iterations and data variables available. Then the `read_data` function can handle data separated into different simulation restarts, join different multiprocessing chunks, and read different refinement levels. Since the raw data can sometimes take a long time to open, `read_data` can also split the data per iteration, instead of per variable, for quicker access in future reads. Additionally, `read_data` also has the capacity to read in data directly from checkpoint files instead.

Then, once input data is provided, users can directly request a wide range of relativistic quantities, including: spacetime; matter (Eulerian, Lagrangian, or conserved); BSSNOK; 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.

Users request quantities via a 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.

The Journal of Open Source Software

Related packages

`EBWeyl` (Munoz & Bruni, 2023) was a precursor to `aurel` as it provided calculations of gravito-electromagnetic contributions from base spacetime and matter quantities. `aurel` has a completely different structure (relying on the automatic dependency resolution), provides calculations of many more terms, over time, and has entirely new features: symbolic calculations, reading `Einstein Toolkit` data, and built-in solutions. These capabilities, therefore, overlap with many other codes; below, we list only Python packages.

`aurel` provides symbolic calculations, so may overlap with other general relativity symbolic packages, see: `GraviPy` (Czaja, n.d.), `SageManifolds` (Gourgoulhon et al., 2015), `EinsteinPy` (Bapat et al., 2020), `Pytearcat` (Martín & Sureda, 2022), and `OGRePy` (Shoshany, 2025).

`aurel` can work with data from any origin, whether analytically generated or from a numerical relativity simulation; they just need to be passed as numpy arrays. But `aurel` also provides tools to read 3D data from `Carpet Einstein Toolkit` simulations (Löffler & others, 2012; Rizzo et al., 2025). Therefore, there is also an overlap with: `PostCactus` (Kastaun, n.d.), `kuibit` (Bozzola, 2021), `scidata` (Radice, n.d.), and `mayawaves` (Ferguson et al., 2025).

Then, several notable Python packages for general relativity focus on different aspects beyond the current targets of `aurel`: - code generation: `NRPy` (Ruchlin et al., 2018) - evolution codes: `COFFEE` (Doulis et al., 2019) and `Engrenage` (Clough & Aurrekoetxea, n.d.) - geodesic and raytracing: `PyHole` (Wittig & Grover, 2017), `EinsteinPy` (Bapat et al., 2020), `GREOPy` (Hackstein & Hackmann, 2025), `PyGRO` (Della Monica, 2025) - marginaly trapped outer surfaces: `distorted-motsfinder` (Pook-Kolb et al., 2019)

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.

Bapat, S., Saha, R., Bhatt, B., Jain, S., Jain, A., Vela, S. O., Khandelwal, P., Shivottam, J., Ma, J., Ng, G. S., Kerhalkar, P., Sarode, H. S., Sharma, R., Gupta, M., Gupta, D., Tyagi, T., Rustagi, T., Singh, V., Bansal, S., … Chan, M. Y. (2020). *EinsteinPy: A community python package for general relativity*. https://arxiv.org/abs/2005.11288

Bozzola, G. (2021). Kuibit: Analyzing einstein toolkit simulations with python. *Journal of Open Source Software*, *6*(60), 3099. https://doi.org/10.21105/joss.03099

Clough, K., & Aurrekoetxea, J. (n.d.). *Engrenage*. GitHub. https://github.com/GRTLCollaboration/engrenage

Czaja, W. (n.d.). *GraviPy, tensor calculus package for general relativity*. https://pypi.python.org/pypi/GraviPy

Della Monica, R. (2025). PyGRO: A python integrator for general relativistic orbits. *Astronomy &Amp; Astrophysics*, *698*, A193. https://doi.org/10.1051/0004-6361/202554300

Doulis, G., Frauendiener, J., Stevens, C., & Whale, B. (2019). COFFEE—an MPI-parallelized python package for the numerical evolution of differential equations. *SoftwareX*, *10*, 100283. https://doi.org/https://doi.org/10.1016/j.softx.2019.100283

Ferguson, D., Anne, S., Gracia-Linares, M., Iglesias, H., Jan, A., Martinez, E., Lu, L., Meoni, F., Nowicki, R., Trostel, M., Tsao, B.-J., & Valorz, F. (2025). *Mayawaves* (Version v2025.12). Zenodo. https://doi.org/10.5281/zenodo.17981058

Gourgoulhon, E., Bejger, M., & Mancini, M. (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

Hackstein, J. P., & Hackmann, E. (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, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Kastaun, W. (n.d.). *PostCactus*. GitHub. https://github.com/wokast/PyCactus

Löffler, F., & others. (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

Martín, M. S., & Sureda, J. (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

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A. (2017). SymPy: Symbolic computing in Python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

Munoz, R. L., & Bruni, M. (2023). EBWeyl: A code to invariantly characterize numerical spacetimes. *Classical and Quantum Gravity*, *40*(13), 135010. https://doi.org/10.1088/1361-6382/acd6cf

Pook-Kolb, D., Birnholtz, O., Krishnan, B., & Schnetter, E. (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, D. (n.d.). *Scidata*. Bitbucket. https://bitbucket.org/dradice/scidata/src/master/

Rizzo, M., Haas, R., Brandt, S. R., Etienne, Z., Ferguson, D., Sanches, L. T., Tsao, B.-J., Werneck, L., Boyer, D., Bozzola, G., Cheng, C.-H., Cupp, S., Diener, P., Jacques, T. P., Ji, L., Macpherson, H., Markin, I., Schnetter, E., Tichy, W., … Zink, B. (2025). *The einstein toolkit* (The "Martin D. Kruskal" release, ET_2025_05). Zenodo. https://doi.org/10.5281/zenodo.15520463

Ruchlin, I., Etienne, Z. B., & Baumgarte, T. W. (2018). SENR/NRPy+: Numerical relativity in singular curvilinear coordinate systems. *Physical Review D*, *97*(6). https://doi.org/10.1103/physrevd.97.064036

Shoshany, B. (2025). OGRePy: An object-oriented general relativity package for python. *Journal of Open Research Software*, *13*. https://doi.org/https://doi.org/10.5334/jors.558

Wittig, A. N., & Grover, J. (2017). *PyHole: General relativity ray tracing and analysis tool*. https://eprints.soton.ac.uk/453123/