

# <sup>1</sup> aurel: A Python package for automatic relativistic calculations

<sup>3</sup> Robyn L. Munoz  <sup>1</sup>, Christian T. Byrnes  <sup>1</sup>, and Will J. Roper  <sup>1</sup>

<sup>4</sup> 1 Department of Physics and Astronomy, University of Sussex, Brighton, BN1 9QH, United Kingdom

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright and release the work under a <sup>16</sup> Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))<sub>17</sub>

## <sup>5</sup> Summary

<sup>6</sup> aurel is an open-source Python package designed to automatically calculate *relativistic* <sup>7</sup> quantities. It uses an efficient, flexible and user-friendly caching and dependency-tracking <sup>8</sup> system, ideal for managing the highly nonlinear nature of general relativity. The package <sup>9</sup> supports both symbolic and numerical calculations. The symbolic part extends SymPy with <sup>10</sup> additional tensorial calculations. The numerical part computes a wide range of tensorial <sup>11</sup> quantities, such as curvature, matter kinematics and much more, directly from any spacetime <sup>12</sup> and matter data arrays using finite-difference methods. Inputs can be either generated from <sup>13</sup> analytical expressions or imported from Numerical Relativity (NR) simulations, with helper <sup>14</sup> functions provided to read in data from standard NR codes. Given the increasing use of NR, <sup>15</sup> aurel offers a timely post-processing tool to support the popularisation of this field.

## <sup>16</sup> Statement of need

<sup>17</sup> General relativity describes matter as moving according to how distances shrink or expand; <sup>18</sup> likewise, the intervals of space and time evolve depending on the distribution of matter. <sup>19</sup> Handling this dynamic “mesh” of distances and times requires elaborate tensor algebra that, in <sup>20</sup> some cases, can only be managed with symbolic or numerical tools. Naturally, NR has become <sup>21</sup> essential for modern astrophysics, cosmology, and gravitational physics, most notably in the <sup>22</sup> modelling of gravitational-wave signals.

<sup>23</sup> While established computational frameworks focus on solving and evolving Einstein’s field <sup>24</sup> equations, with specific key diagnostics, they leave calculations of the remaining analysis to <sup>25</sup> the discretion of the researchers. Newcomers to the field then face a substantial overhead until <sup>26</sup> they develop their own personal post-processing codes. Established researchers also face the <sup>27</sup> tedious task of handling intermediary variables and indices when calculating new quantities. <sup>28</sup> The field then suffers from this error-prone, time-consuming process and would benefit from <sup>29</sup> an accessible, open-source, standardised framework to automate these steps.

<sup>30</sup> We therefore present aurel, an open-source Python package designed to streamline relativistic <sup>31</sup> calculations. It is hosted on [GitHub](#) and is available on [PyPI](#). The documentation is available <sup>32</sup> through [GitHub Pages](#).

## <sup>33</sup> State of the field

<sup>34</sup> When looking for general relativity Python packages, there are a number of tools that provide <sup>35</sup> symbolic calculations ([Bapat et al., 2020](#); [Czaja, n.d.](#); [Della Monica, 2025](#); [Gourgoulhon et al.,](#) <sup>36</sup> [2015](#); [Hackstein & Hackmann, 2025](#); [Martín & Sureda, 2022](#); [Shoshany, 2025](#); [Wittig & Grover,](#) <sup>37</sup> [2017](#)). Or, one may also consider computer algebra systems ([Maplesoft, 2025](#); [Martín-García](#)

& others, 2025; Wolfram Research, 2025). However, when non-linearities become too complex for symbolic packages, NR is used instead.

Einstein Toolkit (Löffler & others, 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, provided by Carpet, there are Python reading packages available (Bozzola, 2021; Ferguson et al., 2025; Kastaun, n.d.; Radice, n.d.). 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 (Andrade et al., 2021; Barrera-Hinojosa & Li, 2020; Palenzuela et al., 2025; Wright, 2018; Zhang 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 (Grasso et al., 2021; Pook-Kolb et al., 2019) focus on ray tracing, or apparent-horizon finding, which are currently beyond the scope of aurel. While others have more overlap (Cranganore et al., 2025; Munoz & Bruni, 2023) 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 & 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); NR formulations; 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

87 spacetime covariant derivatives and Lie derivatives. All spatial derivatives are computed by the  
88 FiniteDifference class that provides 2nd, 4th, 6th and 8th order schemes, using periodic,  
89 symmetric or one-sided boundary conditions.

## 90 Automatic Computational Pathway

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

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

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

## 108 Time dependence

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

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

## 118 Research Impact Statement

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

## 130 AI usage disclosure

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

## 140 Acknowledgements

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

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

145 Andrade, T., Salo, L. A., Aurrekoetxea, J. C., Bamber, J., Clough, K., Croft, R., Jong, E.  
 146 de, Drew, A., Duran, A., Ferreira, P. G., Figueiras, P., Finkel, H., França, T., Ge, B.-X.,  
 147 Gu, C., Helfer, T., Jäykkä, J., Joana, C., Kunesch, M., ... Wong, K. (2021). GRChombo:  
 148 An adaptable numerical relativity code for fundamental physics. *Journal of Open Source  
 149 Software*, 6(68), 3703. <https://doi.org/10.21105/joss.03703>

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

154 Barrera-Hinojosa, C., & Li, B. (2020). GRAMSES: A new route to general relativistic  $N$ -body  
 155 simulations in cosmology. Part i. Methodology and code description. *JCAP*, 01, 007.  
 156 <https://doi.org/10.1088/1475-7516/2020/01/007>

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

159 Cranganore, S. S., Bodnar, A., Berzins, A., & Brandstetter, J. (2025). Einstein fields: A neural  
 160 perspective to computational general relativity. <https://arxiv.org/abs/2507.11589>

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

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

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

168 Gourgoulhon, E., Bejger, M., & Mancini, M. (2015). Tensor calculus with open-source  
 169 software: The SageManifolds project. *Journal of Physics: Conference Series*, 600(1),  
 170 012002. <https://doi.org/10.1088/1742-6596/600/1/012002>

171 Grasso, M., Villa, E., Korzyński, M., & Matarrese, S. (2021). Isolating nonlinearities of  
 172 light propagation in inhomogeneous cosmologies. *Phys. Rev. D*, 104(4), 043508. <https://doi.org/10.1103/PhysRevD.104.043508>

174 Hackstein, J. P., & Hackmann, E. (2025). GREOPy: A python package for solving the

- 175        emitter-observer problem in general relativity. *Journal of Open Source Software*, 10(112),  
 176        8765. <https://doi.org/10.21105/joss.08765>
- 177        Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D.,  
 178        Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk,  
 179        M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., ... Oliphant,  
 180        T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- 182        Kastaun, W. (n.d.). *PostCactus*. GitHub. <https://github.com/wokast/PyCactus>
- 183        Löffler, F., & others. (2012). The einstein toolkit: A community computational infrastructure  
 184        for relativistic astrophysics. *Class. Quant. Grav.*, 29, 115001. <https://doi.org/10.1088/0264-9381/29/11/115001>
- 186        Maplesoft, a. division of W. M. Inc. (2025). *Maple, version 2025.2*. <https://www.maplesoft.com/products/maple/>
- 188        Martín, M. S., & Sureda, J. (2022). Pytearcat: PYthon TEnsor AlgebRa calCuLATOR a python  
 189        package for general relativity and tensor calculus. *Astronomy and Computing*, 100572.  
 190        <https://doi.org/10.1016/j.ascom.2022.100572>
- 191        Martín-García, J. M., & others. (2025). *xAct: Efficient tensor computer algebra for the  
 192        wolfram language*. <https://www.xact.es/>
- 193        Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar,  
 194        A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller,  
 195        R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., ... Scopatz, A.  
 196        (2017). SymPy: Symbolic computing in Python. *PeerJ Computer Science*, 3, e103.  
 197        <https://doi.org/10.7717/peerj-cs.103>
- 198        Munoz, R. L., & Bruni, M. (2023). EBWeyl: A code to invariantly characterize numerical  
 199        spacetimes. *Classical and Quantum Gravity*, 40(13), 135010. <https://doi.org/10.1088/1361-6382/acd6cf>
- 201        Palenzuela, C., Bezares, M., Liebling, S., Schianchi, F., Abalos, J. F., Aguilera-Miret, R.,  
 202        Bona, C., Carretero, J. A., Massò, J., Smith, M. P., Amponsah, K., Kornet, K., Miñano,  
 203        B., Pareek, S., & Radia, M. (2025). *MHDuet : A high-order general relativistic radiation  
 204        MHD code for CPU and GPU architectures*. <https://arxiv.org/abs/2510.13965>
- 205        Pook-Kolb, D., Birnholtz, O., Krishnan, B., & Schnetter, E. (2019). Existence and stability  
 206        of marginally trapped surfaces in black-hole spacetimes. *Phys. Rev. D*, 99(6), 064005.  
 207        <https://doi.org/10.1103/PhysRevD.99.064005>
- 208        Radice, D. (n.d.). *Scidata*. Bitbucket. <https://bitbucket.org/dradice/scidata/src/master>
- 209        Rizzo, M., Haas, R., Brandt, S. R., Etienne, Z., Ferguson, D., Sanches, L. T., Tsao, B.-  
 210        J., Werneck, L., Boyer, D., Bozzola, G., Cheng, C.-H., Cupp, S., Diener, P., Jacques,  
 211        T. P., Ji, L., Macpherson, H., Markin, I., Schnetter, E., Tichy, W., ... Zink, B. (2025).  
 212        *The einstein toolkit* (The "Martin D. Kruskal" release, ET\_2025\_05). Zenodo. <https://doi.org/10.5281/zenodo.15520463>
- 214        Shoshany, B. (2025). OGRePy: An object-oriented general relativity package for python.  
 215        *Journal of Open Research Software*, 13. <https://doi.org/10.5334/jors.558>
- 216        Wittig, A. N., & Grover, J. (2017). *PyHole: General relativity ray tracing and analysis tool*.  
 217        <https://eprints.soton.ac.uk/453123/>
- 218        Wolfram Research, Inc. (2025). *Mathematica, version 14.3*. <https://www.wolfram.com/mathematica>
- 220        Wright, A. (2018). *AlexJamesWright/METHOD: Initial public release* (Version 1.0). Zenodo.  
 221        <https://doi.org/10.5281/zenodo.1404697>

- <sup>222</sup> Zhang, H., Li, B., Weinzierl, T., & Barrera-Hinojosa, C. (2025). ExaGRyPE: Numerical general  
<sup>223</sup> relativity solvers based upon the hyperbolic PDEs solver engine ExaHyPE. *Comput. Phys. Commun.*, 307, 109435. <https://doi.org/10.1016/j.cpc.2024.109435>  
<sup>224</sup>

DRAFT