

# <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 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

- <sup>5</sup> [Review](#) 
- <sup>6</sup> [Repository](#) 
- <sup>7</sup> [Archive](#) 

---

Editor: [Open Journals](#) 

Reviewers:

- <sup>8</sup> [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

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

## Summary/Abstract

<sup>17</sup> aurel is an open-source Python package designed to automatically calculate *relativistic*  
<sup>18</sup> quantities. It uses an efficient, flexible and user-friendly caching and dependency-tracking  
<sup>19</sup> system, ideal for managing the highly nonlinear nature of general relativity. The package  
<sup>20</sup> supports both symbolic and numerical calculations. The symbolic part extends SymPy with  
<sup>21</sup> additional tensorial calculations. The numerical part computes a wide range of tensorial  
<sup>22</sup> quantities, such as curvature, matter kinematics and much more, directly from any spacetime  
<sup>23</sup> and matter data arrays using finite-difference methods. Inputs can be either generated from  
<sup>24</sup> analytical expressions or imported from numerical relativity simulations. For users of Einstein  
<sup>25</sup> Toolkit, aurel also provides utilities to load 3D data generated with Carpet. Given the  
<sup>26</sup> increasing use of numerical relativity, aurel offers a timely post-processing tool to support the  
<sup>27</sup> popularisation of this field.

## Statement of need

<sup>28</sup> General relativity describes matter as moving according to how distances shrink or expand;  
<sup>29</sup> likewise, the intervals of time and space evolve depending on the distribution of matter.  
<sup>30</sup> Handling this dynamic “mesh” of distances and times requires elaborate tensor algebra that,  
<sup>31</sup> in some cases, can only be managed with symbolic or numerical tools. Naturally, numerical  
<sup>32</sup> relativity has become an essential tool in modern astrophysics, cosmology, and gravitational  
<sup>33</sup> physics, most notably in the modelling of gravitational-wave signals.

<sup>34</sup> While established computational frameworks focus on solving and evolving Einstein’s field  
<sup>35</sup> equations with specific key diagnostics, they leave calculations of the remaining analysis to the  
<sup>36</sup> discretion of the researchers. Newcomers to the field then face a substantial overhead until  
<sup>37</sup> they develop their own personal post-processing codes. Established researchers also face the  
<sup>38</sup> 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.

<sup>39</sup> Therefore, we present aurel, an open-source Python package designed to streamline relativistic  
<sup>40</sup> calculations. It is hosted on [GitHub](#) and is available on [PyPI](#). The documentation is available  
<sup>41</sup> through [GitHub Pages](#).

## Features

<sup>42</sup> aurel provides an intuitive interface for the automatic calculation of general relativistic  
<sup>43</sup> quantities, either symbolically (with AurelCoreSymbolic, built on SymPy ([Meurer et al., 2017](#)))  
<sup>44</sup> or numerically (with AurelCore, which heavily utilises numpy.einsum ([Harris et al., 2020](#)) for  
<sup>45</sup> efficient operations on array data structures).

39 Both require base quantities such as the spacetime coordinates or the parameters of the  
40 Cartesian numerical grid, as well as the spacetime and matter distributions (the Minkowski  
41 vacuum is otherwise assumed). These inputs can either come from analytical expressions,  
42 with a couple of built-in solutions available, or from output data from any numerical relativity  
43 simulations.

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

52 Then, once input data is provided, users can directly request a wide range of relativistic  
53 quantities, including: spacetime; matter (Eulerian, Lagrangian, or conserved); BSSNOK;  
54 constraints; fluid covariant kinematics; null ray expansion; 3- and 4-dimensional curvature;  
55 gravito-electromagnetism; Weyl scalars and invariants (including gravitational waves). To see  
56 a full list of available quantities, see: [descriptions](#). Tools are also provided for spatial and  
57 spacetime covariant derivatives and Lie derivatives. All spatial derivatives are computed by the  
58 `FiniteDifference` class that provides 2nd, 4th, 6th and 8th order schemes, using periodic,  
59 symmetric or one-sided boundary conditions.

## 60 Automatic Computational Pathway

61 The `aurel` automatic process composes a computational pathway at runtime to evaluate  
62 the requested quantities. This is implemented through a lazy-evaluation memoised property  
63 pattern, where each quantity is defined as a method of the core class that may depend on  
64 other quantities.

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

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

## 77 Time dependence

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

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

## 87 Related packages

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

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

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

102 Then, several notable Python packages for general relativity focus on different aspects beyond  
103 the current targets of aurel: - code generation: NRPy ([Ruchlin et al., 2018](#)) - evolution  
104 codes: COFFEE ([Doulis et al., 2019](#)) and Engrenage ([Clough & Aurrekoetxea, n.d.](#)) - geodesic  
105 and raytracing: PyHole ([Wittig & Grover, 2017](#)), EinsteinPy ([Bapat et al., 2020](#)), GRE0Py  
106 ([Hackstein & Hackmann, 2025](#)), PyGR0 ([Della Monica, 2025](#)) - marginaly trapped outer surfaces:  
107 distorted-motsfinder ([Pook-Kolb et al., 2019](#))

## 108 Acknowledgements

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

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

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

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

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

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

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

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

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

131 Gourgoulhon, E., Bejger, M., & Mancini, M. (2015). Tensor calculus with open-source

- 132 software: The SageManifolds project. *Journal of Physics: Conference Series*, 600(1),  
133 012002. <https://doi.org/10.1088/1742-6596/600/1/012002>
- 134 Hackstein, J. P., & Hackmann, E. (2025). GREOPy: A python package for solving the  
135 emitter-observer problem in general relativity. *Journal of Open Source Software*, 10(112),  
136 8765. <https://doi.org/10.21105/joss.08765>
- 137 Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D.,  
138 Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk,  
139 M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., ... Oliphant,  
140 T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- 141 Kastaun, W. (n.d.). *PostCactus*. GitHub. <https://github.com/wokast/PyCactus>
- 142 Löffler, F., & others. (2012). The Einstein Toolkit: a community computational infrastructure  
143 for relativistic astrophysics. *Class. Quant. Grav.*, 29, 115001. <https://doi.org/10.1088/0264-9381/29/11/115001>
- 144 Martín, M. S., & Sureda, J. (2022). Pytearcat: PYthon TEtensor AlgebRa calCuLATOR a python  
145 package for general relativity and tensor calculus. *Astronomy and Computing*, 100572.  
146 <https://doi.org/10.1016/j.ascom.2022.100572>
- 147 Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar,  
148 A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller,  
149 R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., ... Scopatz, A.  
150 (2017). SymPy: Symbolic computing in Python. *PeerJ Computer Science*, 3, e103.  
151 <https://doi.org/10.7717/peerj-cs.103>
- 152 Munoz, R. L., & Bruni, M. (2023). EBWeyl: A code to invariantly characterize numerical  
153 spacetimes. *Classical and Quantum Gravity*, 40(13), 135010. <https://doi.org/10.1088/1361-6382/acd6cf>
- 154 Pook-Kolb, D., Birnholtz, O., Krishnan, B., & Schnetter, E. (2019). Existence and stability  
155 of marginally trapped surfaces in black-hole spacetimes. *Phys. Rev. D*, 99(6), 064005.  
156 <https://doi.org/10.1103/PhysRevD.99.064005>
- 157 Radice, D. (n.d.). *Scidata*. Bitbucket. <https://bitbucket.org/dradice/scidata/src/master/>
- 158 Rizzo, M., Haas, R., Brandt, S. R., Etienne, Z., Ferguson, D., Sanches, L. T., Tsao, B.-  
159 J., Werneck, L., Boyer, D., Bozzola, G., Cheng, C.-H., Cupp, S., Diener, P., Jacques,  
160 T. P., Ji, L., Macpherson, H., Markin, I., Schnetter, E., Tichy, W., ... Zink, B. (2025).  
161 *The einstein toolkit* (The "Martin D. Kruskal" release, ET\_2025\_05). Zenodo. <https://doi.org/10.5281/zenodo.15520463>
- 162 Ruchlin, I., Etienne, Z. B., & Baumgarte, T. W. (2018). SENR/NRPy+: Numerical relativity  
163 in singular curvilinear coordinate systems. *Physical Review D*, 97(6). <https://doi.org/10.1103/physrevd.97.064036>
- 164 Shoshany, B. (2025). OGRePy: An object-oriented general relativity package for python.  
165 *Journal of Open Research Software*, 13. <https://doi.org/https://doi.org/10.5334/jors.558>
- 166 Wittig, A. N., & Grover, J. (2017). *PyHole: General relativity ray tracing and analysis tool*.  
167 <https://eprints.soton.ac.uk/453123/>