The Journal of Open Source Software

kuibit: Analyzing Einstein Toolkit simulations with Python

Gabriele Bozzola[1]

1 Steward Observatory and Astronomy Department, University of Arizona

DOI: 10.21105/joss.03099

Software

- Review
- Repository
- Archive

Editor: Eloisa Bentivegna

Reviewers:

- @yurlungur
- @eloisabentivegna

Submitted: 30 January 2021
Published: 12 April 2021

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

`kuibit`[1] is a Python library for analyzing simulations performed with the `Einstein Toolkit`[2] Löffler et al. (2012), a free and open-source code for numerical relativity and relativistic astrophysics. Over the past years, numerical simulations like the ones enabled by the `Einstein Toolkit` have become a critical tool in modeling, predicting, and understanding several astrophysical phenomena, including binary black hole or neutron star mergers. As a result of the recent detections of gravitational waves by the LIGO-Virgo collaboration, these studies are at the forefront of scientific research. The package presented in this paper, `kuibit`, provides an intuitive infrastructure to read and represent the output of the `Einstein Toolkit`. This simplifies analyzing simulations and significantly lowers the barrier in learning how to use the tool.

Statement of need

The `Einstein Toolkit` is a software for numerical simulations based on the `Cactus` computational framework and designed to be accessible for both users and developers. Numerical-relativity simulations require large and complex codes, which have to run on the world's largest supercomputers. `Einstein Toolkit` significantly reduces this complexity and improves accessibility by splitting infrastructure code from physics one. On one side, there is memory management, parallelization, grid operations, and all the other low-level details that are needed to successfully perform a simulation but do not strictly depend on the physical system under consideration. On the other, there are the physics modules, which implement the scientific aspects of the simulation. Codes are developed by domain-experts and researchers can focus on their goals without having to worry about the technical details of the implementation. This makes the `Einstein Toolkit` easier to use and extend.

Despite the advancements made by the `Einstein Toolkit`, there is still a big leap between running a simulation and obtaining scientific results. The output from the `Einstein Toolkit` is a collection of files with different formats and structures, with data that is typically spread across multiple files (one or more for each MPI process) in various directories (one per checkpoint). Reading the simulation output and properly combining all the data is a challenging task. Even once the output is read, traditional data structures are not a good representation of the physical quantities. For instance, representing variables defined on an adaptive-mesh-refined grid as simple arrays completely ignores all the information on the grid

[1] A kuibit (harvest pole) is the tool traditionally used by the Tohono O'odham people to reach the fruit of the Saguaro cacti during the harvesting season.

[2] While `kuibit` is designed for the `Einstein Toolkit`, most of its capabilities will work also for all the other codes based on `Cactus`. For instance, it is known that `kuibit` can be used to analyze `Illinois GRMHD` (Duez et al., 2005) simulations.

structure, making some operations impractical or impossible to perform. The lack of suitable interfaces introduces significant friction in exploring the scientific content of a simulation. `kuibit` takes care of both the aspects of reading the simulation data and of providing high-level representations of the data that closely follows what researchers are used to. In addition to this, `kuibit` also includes a set of routines that are commonly used in the field: for example, it handles unit conversion (including from geometrized units to physical), it has the noise curves of known detectors, or it computes gravitational-waves from simulation data.

`kuibit` is based on the same design (and in various cases, implementation details too) of a pre-existing package named `PostCactus` (Kastaun, n.d.). Like `PostCactus`, `kuibit` has two groups of modules. The first is to define custom data-types for time series, Fourier spectra, multipolar decompositions, and grid data (both on uniform grids and mesh-refined ones). The second group consists of the readers, which are a collection of tools to scan the simulation output and organize it. The main reader is a class `SimDir` which provides the interface to access all the data in the simulation. For instance, the `timeseries` attribute in `SimDir` is a dictionary-like object that contains all the time series in the output. When reading data, `kuibit` takes care of all the low-level details, like handling transparently simulation restarts, or merging grid data stored in different files. Therefore, users can easily access the data regardless of how complicated the structure of the output is. Moreover, `kuibit` does not assume any particular organization of the output and uses regular expressions to find the relevant information from filenames or metadata, allowing for flexibility in the simulation workflow.

Currently, `kuibit` is the only available package for quantitative analysis of simulations that is free to use and that comes with documentation, tutorials, and examples. Tools like `VisIt` (Childs et al., 2012) or `rugutils` (Guercilena, n.d.) focus only on visualizing grid data, while other packages like `POWER` (Johnson et al., 2018), or `pyGWAnalysis` (Reisswig, n.d.) only on gravitational-wave data. Capabilities similar to those of `kuibit` are offered by `Simulatio nTools` (Hinder & Wardell, 2012), that runs on the proprietary Wolfram Mathematica, and by `PostCactus` (Kastaun, n.d.) and `scidata` (Radice, n.d.), which, at the moment, do not support Python3 and do not have documentation. In addition to this, several research groups develop their own private analysis software.

`kuibit` embraces the core principles of the `Einstein Toolkit`: On one side, `kuibit` solves the engineering problems of reading and representing `Einstein Toolkit` data, so that researchers can directly pursue their scientific goals without having to worry about how the data is stored. With `kuibit`, the entry barrier into using the `Einstein Toolkit` is the lowest it has ever been, and students and researchers can inspect and visualize simulations in just a few lines of code. On the other side, `kuibit` is designed to be a code for the community: it is free and does not require any proprietary software to run, it is openly developed with an emphasis on readability and maintainability, and it encourages contributions.

Acknowledgments

Gabriele Bozzola is supported by by the Frontera Fellowship by the Texas Advanced Computing Center (TACC). Frontera (Stanzione et al., 2020) is founded by NSF grant OAC-1818253. This work was in part supported by NSF Grant PHY-1912619 to the University of Arizona and made use of computational resources provided by the Extreme Science and Engineering Discovery Environment (XSEDE) under grant number TG-PHY190020. XSEDE is supported by the NSF grant No. ACI-1548562. Gabriele Bozzola wishes to thank Wolfgang Kastaun for publicly releasing his `PostCactus` package (Kastaun, n.d.) without which, `kuibit` would not exist.

References

Brandt, S. R., Brendal, B., Gabella, W. E., Haas, R., Karakaş, B., Kedia, A., Rosofsky, S. G., Schaffarczyk, A. P., Alcubierre, M., Alic, D., Allen, G., Ansorg, M., Babiuc-Hamilton, M., Baiotti, L., Benger, W., Bentivegna, E., Bernuzzi, S., Bode, T., Bruegmann, B., … Zlochower, Y. (2020). *The einstein toolkit* (The "Turing" release, ET_2020_05). Zenodo. https://doi.org/10.5281/zenodo.3866075

Childs, H., Brugger, E., Whitlock, B., Meredith, J., Ahern, S., Pugmire, D., Biagas, K., Miller, M., Harrison, C., Weber, G. H., Krishnan, H., Fogal, T., Sanderson, A., Garth, C., Bethel, E. W., Camp, D., Rübel, O., Durant, M., Favre, J. M., & Navrátil, P. (2012). VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight* (pp. 357–372).

Duez, M. D., Liu, Y. T., Shapiro, S. L., & Stephens, B. C. (2005). Relativistic magnetohydrodynamics in dynamical spacetimes: Numerical methods and tests. *Phys. Rev. D*, *72*, 024028. https://doi.org/10.1103/PhysRevD.72.024028

Guercilena, F. (n.d.). PyCactus. In *BitBucket repository*. BitBucket. https://bitbucket.org/relastro/rugutils

Hinder, I., & Wardell, B. (2012). SimulationTools. In *BitBucket repository*. BitBucket. https://simulationtools.org

Johnson, D., Huerta, E. A., & Haas, R. (2018). Python Open source Waveform ExtractoR (POWER): an open source, Python package to monitor and post-process numerical relativity simulations. *Classical and Quantum Gravity*, *35*(2), 027002. https://doi.org/10.1088/1361-6382/aa9cad

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

Löffler, F., Faber, J., Bentivegna, E., Bode, T., Diener, P., Haas, R., Hinder, I., Mundim, B. C., Ott, C. D., Schnetter, E., Allen, G., Campanelli, M., & Laguna, P. (2012). The Einstein Toolkit: A Community Computational Infrastructure for Relativistic Astrophysics. *Class. Quantum Grav.*, *29*(11), 115001. https://doi.org/doi:10.1088/0264-9381/29/11/115001

Radice, D. (n.d.). Scidata. In *BitBucket repository*. BitBucket. https://bitbucket.org/dradice/scidata

Reisswig, C. (n.d.). pyGWAnalysis. In *SVN repository*. Einstein Toolkit. https://svn.einsteintoolkit.org/pyGWAnalysis/

Stanzione, D., West, J., Evans, R. T., Minyard, T., Ghattas, O., & Panda, D. K. (2020). Frontera: The evolution of leadership computing at the national science foundation. *Practice and Experience in Advanced Research Computing*, 106–111. https://doi.org/10.1145/3311790.3396656