Paicos: A Python package for analysis of (cosmological) simulations performed with Arepo

Thomas Berlok [1,2,¶], **Léna Jlassi** [2], **Ewald Puchwein** [2], **and Troels Haugbølle** [1]

1 Niels Bohr Institute, University of Copenhagen, Denmark **2** Leibniz-Institut für Astrophysik Potsdam, Germany ¶ Corresponding author

Summary

Cosmological simulations evolve dark matter and baryons subject to gravitational and hydrodynamic forces (Vogelsberger et al., 2020). The simulations start at high redshift and capture hierarchical structure formation where small structures form first and later assemble to larger structures (Springel et al., 2005). The Arepo code is a versatile finite-volume code which can solve the magnetohydrodynamic equations on an unstructured Voronoi mesh in a cosmologically comoving frame (Springel, 2010; Weinberger et al., 2020).

Here we present Paicos, a new object-oriented Python package for analyzing simulations performed with Arepo. Paicos strives to reduce the learning curve for students and researchers getting started with Arepo simulations. As such, Paicos includes many examples in the form of Python scripts and Jupyter notebooks (Kluyver et al., 2016) as well as an online documentation describing the installation procedure and recommended first steps.

Paicos' main features are automatic handling of cosmological and physical units, computation of derived variables, 2D visualization (slices and projections), 1D and 2D histograms, and easy saving and loading of derived data including units and all the relevant metadata.

Paicos relies heavily on well-established open source software libraries such as NumPy (Harris et al., 2020; van der Walt et al., 2011), Scipy (Virtanen et al., 2020), h5py (Collette, 2013), Cython (Behnel et al., 2011) and astropy (Astropy Collaboration, 2013) and contains features for interactive data analysis inside Ipython terminals (Perez & Granger, 2007) and Jupyter notebooks (Kluyver et al., 2016), e.g., tab completion of data keywords and Latex rendering of data units. Paicos also contains a number of tests that are automated using pytest (Krekel et al., 2004), CircleCi and GitHub workflows.

Statement of need

The Arepo code stores its data output as HDF5 files, which can easily be loaded as NumPy arrays using h5py. However, data visualization of the unstructured mesh used in Arepo is non-trivial and keeping track of the units used in the data outputs can also be a tedious task. The general purpose visualization and data analysis software package yt[1] (Turk et al., 2011) and the visualization package py-sphviewer (Benitez-Llambay, 2015) are both able to perform visualizations of Arepo simulations. Paicos provides an alternative, which is specifically tailored to Arepo simulations. It is in this regard similar to swiftsimio (Borrow & Borrisov, 2020), which was developed specifically for SWIFT simulations (Schaller & others, 2023).

---

[1]See http://yt-project.org/.

We have developed Paicos because we identified a need for an analysis code that simultaneously fulfills the following requirements: 1) is specifically written for analysis of Arepo simulations discretized on a Voronoi mesh 2) provides safeguards against errors related to conversions of cosmological and physical units 3) facilitates working with large data sets by supporting the saving and loading of reduced/derived data including units and metadata 4) contains enough functionality to be useful for practical research tasks while still being light-weight and well-documented enough that it can be installed, used and understood by a junior researcher with little or no assistance.

Overview of key Paicos features

The key Paicos features, which were implemented to fulfill the above requirements, are as follows:

- Functionality for reading cell/particle and group/subhalo catalog data as saved by Arepo in the HDF5 format.
- Unit-handling via astropy with additional support for the a and h factors that are used in cosmological simulations.[2]
- Functionality for automatically obtaining derived variables from the variables present in the Arepo snapshot (e.g. the cell volume, temperature, or magnetic field strength). Within Jupyter notebooks or an Ipython terminal: Tab completion to show which variables are available for a given data group (either by directly loading from the HDF5-file or by automatically calculating a derived variable).
- Functionality for saving and loading data including cosmological and physical units in HDF5-files that automatically include all metadata from the original Arepo snapshot (i.e. the Header, Config and Param groups).
- Functionality for creating slices and projections and saving them in a format that additionally includes information about the size, center and orientation of the image.
- Functionality for creating 1D and 2D histograms (e.g. radial profiles and ρ-T phase-space plots) of large data sets (using OpenMP-enabled Cython).
- Functionality for user customization, e.g. adding new units or functions for computing derived variables.
- Functionality for selecting only a part of a snapshot for analysis and for storing this selection as a new reduced snapshot.

Finally, we also make public GPU implementations of projection functionalities, which are much faster than the OpenMP-parallel CPU implementations described above. The GPU implementations include a mass-conserving SPH-like projection and a ray tracing implementation using a bounding volume hierarchy (BVH) in the form of a binary radix tree for nearest neighbor searches. These GPU implementations use Numba (Lam et al., 2015) and CuPy (Okuta et al., 2017) and require that the user has a CUDA-enabled GPU. Our binary tree implementation follows the the GPU-optimized tree-construction algorithm described in (Karras, 2012) and the implementation of it found in the publicly available Cornerstone Octree GPU-library (Keller et al., 2023). Using our SPH-like implementation on an Nvidia A100 GPU, it takes less than a second to project 100 mio. particles onto an image plane with 4096^2 pixels. This speed enables interactive data exploration. We provide an example IPython widget illustrating this feature. Finally, we note that the returned images include physical units and can be used for scientific analysis.

Paicos is hosted on GitHub at https://github.com/tberlok/paicos. We strongly encourage contributions to Paicos by opening issues and submitting pull requests.

---

[2]See e.g. Pakmor & Springel (2013) and Berlok (2022) for detailed discussions and a derivation of the comoving magnetohydrodynamic equations used in cosmological simulations including magnetic fields.

References

Astropy Collaboration. (2013). Astropy: A community Python package for astronomy. *Astronomy and Astrophysics*, *558*. https://doi.org/10.1051/0004-6361/201322068

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython: The Best of Both Worlds. *Computing in Science and Engineering*, *13*(2), 31–39. https://doi.org/10.1109/MCSE.2010.118

Benitez-Llambay, A. (2015). *Py-sphviewer: Py-SPHViewer v1.0.0* (Version v1.0.0). Zenodo. https://doi.org/10.5281/zenodo.21703

Berlok, T. (2022). Hydromagnetic waves in an expanding universe - cosmological MHD code tests using analytic solutions. *515*(3), 3492–3511. https://doi.org/10.1093/mnras/stac1882

Borrow, J., & Borrisov, A. (2020). Swiftsimio: A python library for reading SWIFT data. *Journal of Open Source Software*, *5*(52), 2430. https://doi.org/10.21105/joss.02430

Collette, A. (2013). *Python and HDF5*. O'Reilly.

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

Karras, T. (2012). Maximizing parallelism in the construction of BVHs, octrees, and k-d trees. *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*, 33–37. https://doi.org/10.2312/EGGH/HPG12/033-037

Keller, S., Cavelan, A., Cabezon, R., Mayer, L., & Ciorba, F. M. (2023). Cornerstone: Octree Construction Algorithms for Scalable Particle Simulations. *arXiv e-Prints*, arXiv:2307.06345. https://doi.org/10.48550/arXiv.2307.06345

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., & team, J. development. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides & B. Scmidt (Eds.), *Positioning and power*

in academic publishing: Players, agents and agendas (pp. 87–90). IOS Press. https://doi.org/10.3233/978-1-61499-649-1-87

Krekel, H., Oliveira, B., Pfannschmidt, R., Bruynooghe, F., Laugher, B., & Bruhin, F. (2004). Pytest 7.4. https://github.com/pytest-dev/pytest

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A LLVM-based python JIT compiler. Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC. https://doi.org/10.1145/2833157.2833162

Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). CuPy: A NumPy-compatible library for NVIDIA GPU calculations. Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Thirty-First Annual Conference on Neural Information Processing Systems (NIPS). http://learningsys.org/nips17/assets/papers/paper_16.pdf

Pakmor, R., & Springel, V. (2013). Simulations of magnetic fields in isolated disc galaxies. 432(1), 176–193. https://doi.org/10.1093/mnras/stt428

Perez, F., & Granger, B. E. (2007). IPython: A System for Interactive Scientific Computing. Computing in Science and Engineering, 9(3), 21–29. https://doi.org/10.1109/MCSE.2007.53

Schaller, M., & others. (2023). Swift: A modern highly-parallel gravity and smoothed particle hydrodynamics solver for astrophysical and cosmological applications. arXiv e-Prints, arXiv:2305.13380. https://doi.org/10.48550/arXiv.2305.13380

Springel, V. (2010). E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh. 401(2), 791–851. https://doi.org/10.1111/j.1365-2966.2009.15715.x

Springel, V., White, S. D. M., Jenkins, A., Frenk, C. S., Yoshida, N., Gao, L., Navarro, J., Thacker, R., Croton, D., Helly, J., Peacock, J. A., Cole, S., Thomas, P., Couchman, H., Evrard, A., Colberg, J., & Pearce, F. (2005). Simulations of the formation, evolution and clustering of galaxies and quasars. 435(7042), 629–636. https://doi.org/10.1038/nature03597

Turk, M. J., Smith, B. D., Oishi, J. S., Skory, S., Skillman, S. W., Abel, T., & Norman, M. L. (2011). yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data. The Astrophysical Journal Supplement Series, 192, 9. https://doi.org/10.1088/0067-0049/192/1/9

van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science and Engineering, 13(2), 22–30. https://doi.org/10.1109/MCSE.2011.37

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1. 0 Contributors. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods, 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Vogelsberger, M., Marinacci, F., Torrey, P., & Puchwein, E. (2020). Cosmological simulations of galaxy formation. Nature Reviews Physics, 2(1), 42–66. https://doi.org/10.1038/s42254-019-0127-2

Weinberger, R., Springel, V., & Pakmor, R. (2020). The AREPO Public Code Release. 248(2), 32. https://doi.org/10.3847/1538-4365/ab908c