




# Sarracen: a Python package for analysis and visualization of smoothed particle hydrodynamics data

Andrew Harris<sup>1\*</sup> and Terrence S. Tricco<sup>1\*</sup> 

<sup>1</sup> Memorial University of Newfoundland, Canada  Corresponding author \* These authors contributed equally.

DOI: [10.21105/joss.05263](https://doi.org/10.21105/joss.05263)

## Software

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

---

Editor: [Warrick Ball](#)  

## Reviewers:

- [@hlim88](#)
- [@JBorrow](#)

Submitted: 09 March 2023

Published: 21 June 2023

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

## Summary

Sarracen is a Python package for analyzing and visualizing smoothed particle hydrodynamics (SPH) data. SPH is a method of fluid simulation that discretizes fluid elements into a collection of particles ([Gingold & Monaghan, 1977](#); [Lucy, 1977](#); [Monaghan, 2005](#); [Price, 2012](#)). This approach works well for many astrophysical problems of interest, and as such there are a number of SPH codes widely used for astrophysical simulations, e.g., [Wadsley et al. \(2017\)](#), [Price et al. \(2018\)](#), [Schaller et al. \(2018\)](#). Sarracen offers a variety of SPH interpolation methods to aid in analysis and visualization of SPH data. It is built in Python so that users can leverage the robust scientific libraries that are available. Much of the core of Sarracen is built upon `pandas` and `Matplotlib`. Users familiar with these packages should be able to use Sarracen to do complex analyses without difficulty. Our intended use is for astrophysical SPH data, but anticipate that our package may be useful in other scientific domains.

## Statement of need

`SpIash` ([Price, 2007](#)) is the current standard bearer for visualization of astrophysical SPH data. It is an open-source, command-line visualization tool written in Fortran. It is comprehensive, highly efficient, and can natively read SPH data from a large number of SPH simulation codes. `SpIash` has a large user base for these reasons. The significant shortcoming of `SpIash` is that it has limited capability for analysis of SPH data. Any complicated analysis requires modification of the Fortran code.

There are publicly available Python solutions for visualization or analysis of astrophysical SPH data. However, these are often specific to a single code, such as `SWIFTsimIO` ([Borrow & Borrisov, 2020](#)), which is dedicated to the `Swift` ([Schaller et al., 2018](#)) cosmological code. `yt` ([Turk et al., 2011](#)) is a general purpose analysis and visualization package for volumetric astrophysical data. Originally designed for data from grid-based codes, recent versions have added support to store SPH particle data directly (instead of storing only a mesh interpolation). `ParaView` ([Ayachit, 2015](#)) and `VisIt` ([Childs et al., 2012](#)) are two open-source visualization applications that can be used with SPH data and scale to large data sets. `ParaView` and `VisIt` additionally offer scripting in Python. `Plonk` ([Mentiplay, 2019](#)) is the work most similar to ours, but uses custom data structures for storing particle data and is limited to reading HDF5 data from the `Phantom` ([Price et al., 2018](#)) SPH code.

Our goal with Sarracen is to provide a Python package that implements the robust interpolation and visualization of SPH data offered by `SpIash`, but which can be used for deeper analysis and integrated into a data scientist's Python toolkit. We use `Matplotlib` ([Hunter, 2007](#)) for visualization, and an extension of the `pandas` ([McKinney, 2010](#)) `DataFrame` structure for storing particle data. Using Sarracen should be familiar for most users. Furthermore, Python

has many high-quality scientific libraries for data manipulation and statistical analysis, such as NumPy (Harris et al., 2020) and SciPy (Virtanen et al., 2020). A user will be able to easily write custom analysis scripts specific to their simulation or area of astronomy and astrophysics. These factors should aid in making analyses more reproducible, efficient, and less error prone. Finally, Sarracen can be run interactively inside of a Jupyter notebook environment, which enables results to be easily shared, presented and modified.

## Features

At its core, Sarracen supports the interpolation of SPH particle data via the SPH smoothing kernel. The basic approach for interpolation of a quantity,  $A$ , is

$$A_a = \sum_b \frac{m_b}{\rho_b} A_b W_{ab}(h_b), \quad (1)$$

where the summation is over neighbouring particles,  $m$  is the mass,  $\rho$  is density, and  $W(h_b)$  is the smoothing kernel with smoothing length,  $h$ . Sarracen includes multiple choices for the smoothing kernel, with the cubic spline as default.

For 3D data, a quantity may be interpolated to a 3D fixed grid, to a 2D grid representing a slice through the data, or to a 1D line that cuts through the volume. Column integrated line-of-sight interpolation is included. Interpolation of 2D data is also supported. Additionally, Sarracen implements the mapping method of Petkova et al. (2018), which exactly computes the volume-averaged quantity within each cell of a fixed grid by analytically computing the integral of the kernel function over the volume of each cell. Sarracen can render interpolated grids with Matplotlib using API syntax inspired by Seaborn (Waskom, 2021). Vector quantities, such as velocity, can be rendered with streamlines or arrow plots. Sarracen uses SciPy to support view rotation, with the rotation specified by Euler angles, a rotation vector, rotation matrix or quaternions.

The interpolation routines in Sarracen use Numba (Lam et al., 2015) to implement multi-threaded CPU parallelization and CUDA-enabled GPU acceleration. Furthermore, Numba translates these routines into optimized machine code using just-in-time compilation. Additionally, operations on data are vectorized by using NumPy.

## Acknowledgements

Thank you to Rebecca Nealon for testing Sarracen and providing feedback and suggestions. This research was enabled in part by support provided by the Digital Research Alliance of Canada. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

## References

- Ayachit, U. (2015). *The paraview guide: A parallel visualization application*. Kitware, Inc.
- Borrow, J., & Borrisov, A. (2020). swiftsimio: A Python library for reading SWIFT data. *The Journal of Open Source Software*, 5(52), 2430. <https://doi.org/10.21105/joss.02430>
- 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., Rübél, 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). <https://doi.org/10.1201/b12985>

- Gingold, R. A., & Monaghan, J. J. (1977). Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, *181*, 375–389. <https://doi.org/10.1093/mnras/181.3.375>
- 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. *Nature*, *585*(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering*, *9*(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A LLVM-based Python JIT Compiler. *Proc. Second Workshop on the LLVM Compiler Infrastructure in HPC*, 1–6. <https://doi.org/10.1145/2833157.2833162>
- Lucy, L. B. (1977). A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, *82*, 1013–1024. <https://doi.org/10.1086/112164>
- McKinney, W. (2010). Data Structures for Statistical Computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). <https://doi.org/10.25080/Majora-92bf1922-00a>
- Mentiplay, D. (2019). Plonk: Smoothed particle hydrodynamics analysis and visualization with Python. *The Journal of Open Source Software*, *4*(44), 1884. <https://doi.org/10.21105/joss.01884>
- Monaghan, J. J. (2005). Smoothed particle hydrodynamics. *Reports on Progress in Physics*, *68*(8), 1703–1759. <https://doi.org/10.1088/0034-4885/68/8/R01>
- Petkova, M. A., Laibe, G., & Bonnell, I. A. (2018). Fast and accurate Voronoi density gridding from Lagrangian hydrodynamics data. *Journal of Computational Physics*, *353*, 300–315. <https://doi.org/10.1016/j.jcp.2017.10.024>
- Price, D. J. (2007). splash: An Interactive Visualisation Tool for Smoothed Particle Hydrodynamics Simulations. *Publications of the Astronomical Society of Australia*, *24*(3), 159–173. <https://doi.org/10.1071/AS07022>
- Price, D. J. (2012). Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics*, *231*(3), 759–794. <https://doi.org/10.1016/j.jcp.2010.12.011>
- Price, D. J., Wurster, J., Tricco, T. S., Nixon, C., Toupin, S., Pettitt, A., Chan, C., Mentiplay, D., Laibe, G., Glover, S., Dobbs, C., Nealon, R., Liptai, D., Worpel, H., Bonnerot, C., Dipierro, G., Ballabio, G., Ragusa, E., Federrath, C., ... Lodato, G. (2018). Phantom: A Smoothed Particle Hydrodynamics and Magnetohydrodynamics Code for Astrophysics. *Publications of the Astronomical Society of Australia*, *35*, e031. <https://doi.org/10.1017/pasa.2018.25>
- Schaller, M., Gonnet, P., Draper, P. W., Chalk, A. B. G., Bower, R. G., Willis, J., & Hausammann, L. (2018). *SWIFT: SPH With Inter-dependent Fine-grained Tasking* (p. ascl:1805.020). Astrophysics Source Code Library, record ascl:1805.020.
- 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. *Astrophysical Journal Supplement Series*, *192*(1), 9. <https://doi.org/10.1088/0067-0049/192/1/9>
- 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>

Wadsley, J. W., Keller, B. W., & Quinn, T. R. (2017). Gasoline2: a modern smoothed particle hydrodynamics code. *Monthly Notices of the Royal Astronomical Society*, 471(2), 2357–2369. <https://doi.org/10.1093/mnras/stx1643>

Waskom, M. (2021). seaborn: statistical data visualization. *The Journal of Open Source Software*, 6(60), 3021. <https://doi.org/10.21105/joss.03021>