The Journal of Open Source Software

Vectorized, Python Array API Standard Compatible Functions for Quadrature, Series Summation, Differentiation, Optimization, and Root Finding in SciPy

Matt Haberland [1¶], **Albert Steppi** [2], **Pamphile Roy** [3], **and Jake Bowhay** [4]

1 California Polytechnic State University, San Luis Obispo, USA **2** Quansight, Austin, USA **3** Consulting Manao GMBH, Vienna, Austria **4** University of Bristol, Bristol, United Kingdom ¶ Corresponding author

DOI: 10.21105/joss.08027

Software

- Review
- Repository
- Archive

Editor: Vincent Knight

Reviewers:

- @alexhroom
- @pescap
- @hoanganhngo610

Submitted: 08 January 2025
Published: 19 March 2026

Summary

Numerical integration, series summation, differentiation, optimization, and root finding are fundamental problems with applications in essentially all domains of science and engineering. Frequently, such problems do not arise individually, but rather in batches; e.g., differentiation of a single curve at many points or minimization of a function for many values of a parameter. In array computing, operations on batches of values can be vectorized. With NumPy (Harris et al., 2020), operations on arrays expressed in Python can be evaluated as (sequential) loops in efficient native code, or in some cases in parallel with SIMD (Adel et al., 2019). Other Python array libraries such as CuPy (Okuta et al., 2017), PyTorch (Ansel et al., 2024), and JAX (Bradbury et al., 2018) are able to exploit GPUs to parallelize vectorized computations. However, SciPy (Virtanen et al., 2020) – the de facto standard Python library for solving the above problems – offered few functions capable of vectorizing the solution of such problems with NumPy (Harris et al., 2020), let alone with alternative array libraries.

This paper discusses several new features that fill this gap, included in SciPy 1.15.0:

- `scipy.differentiate` (Haberland & others, 2024a), a sub-package for numerical differentiation of scalar or vector-valued functions of one or more variables,
- `scipy.integrate.tanhsinh` (Haberland & others, 2023) for quadrature of scalar or vector-valued integrands of one variable,
- `scipy.integrate.nsum` (Haberland & others, 2024b) for summation of real-valued finite or infinite series,
- `scipy.optimize.elementwise` (Haberland et al., 2023) for finding roots and minimization of single-input, single-output functions.

Although the details of the algorithms are inherently distinct, these features rely on a common framework for elementwise iterative methods and share similar interfaces, and the same implementation works with several Python Array API Standard (Consortium for Python Data API Standards, 2024) compatible arrays, including CuPy and PyTorch. These features will dramatically improve performance of end-user applications, and together, they form the backbone of SciPy's new random variable infrastructure.

Statement of need

Before the release of SciPy 1.15.0, the need for these capabilities was partially met in the scientific Python ecosystem. As popular examples, Numdifftools (Brodtkorb, 2022) and

PyNumDiff (Breugel et al., 2022) provide tools for numerical differentiation, quadpy (Schlömer, 2023) and `scipy.integrate` offer several functions for numerical integration, `mpmath`'s (The mpmath development team, 2023) `nsum` function is for series summation, and `scipy.optimize` offers functions for scalar minimization and root finding. However, to the authors' knowledge, the new implementations are unique in that they offer the following advantages.

- Backend-independence: all of these features are "Array API compatible" in the sense that they rely almost entirely on the Python standard library and calls to Python Array API Standard functions/methods. Consequently, the functions accept objects other than NumPy arrays, such as PyTorch tensors and CuPy arrays, and the calculations are performed by the underlying array library.
- Speed: the features take full advantage of vectorized user callables, avoiding slow Python loops and the excessive overhead of repeatedly calling compiled code.
- Prevalence: SciPy is one of the most popular scientific Python packages. If a scientific Python user needs these features, chances are that they already have SciPy installed, eliminating the need to find and learn a new package.
- Ease-of-use: the API reference for these new functions is thorough, and the interfaces share common features and operate smoothly with other SciPy functions.
- Dependability: as with all new SciPy code, these features were designed and implemented following good software development practices. They have been carefully peer-reviewed, and extensive unit tests protect against backward incompatible changes and regressions.

Acknowledgements

We gratefully acknowledge the support of Chan Zuckerberg Initiative Essential Open Source Software for Science Grant CZI EOSS5-0000000176, "SciPy: Fundamental Tools for Biomedical Research". Thanks to all those who have submitted feature requests, bug reports, and review comments related to these features, and especially to SciPy maintainers who reviewed and merged related work.

References

Adel, S., Picus, M., & Gommers, R. (2019). *NEP 38: Using SIMD optimization instructions for performance*. https://numpy.org/neps/nep-0038-SIMD-optimizations.html

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., & others. (2024). PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 929–947. https://doi.org/10.1145/3620665.3640366

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/jax-ml/jax

Breugel, F. V., Liu, Y., Brunton, B. W., & Kutz, J. N. (2022). PyNumDiff: A Python package for numerical differentiation of noisy time-series data. *Journal of Open Source Software*, *7*(71), 4078. https://doi.org/10.21105/joss.04078

Brodtkorb, P. A. (2022). *Numdifftools 0.9.41*. GitHub. https://github.com/pbrod/numdifftools

Consortium for Python Data API Standards. (2024). *Python array API standard*. https://data-apis.org/array-api/latest/

Haberland, M., & others. (2023). *tanhsinh — SciPy manual*. https://docs.scipy.org/doc/

scipy/reference/generated/scipy.integrate.tanhsinh.html

Haberland, M., & others. (2024a). *Finite difference differentiation (scipy.differentiate) — SciPy manual*. https://docs.scipy.org/doc/scipy/reference/differentiate.html

Haberland, M., & others. (2024b). *nsum — SciPy manual*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.nsum.html

Haberland, M., Steppi, A., & others. (2023). *Elementwise scalar optimization (scipy.optimize.elementwise) — SciPy manual*. https://docs.scipy.org/doc/scipy/reference/optimize.elementwise.html

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., & others. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

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*.

Schlömer, N. (2023). *Quadpy*. GitHub. https://github.com/sigma-py/quadpy

The mpmath development team. (2023). *mpmath: A Python library for arbitrary-precision floating-point arithmetic*. http://mpmath.org/

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