Vector: JIT-compilable mathematical manipulations of ragged Lorentz vectors

Saransh Chopra [1,2*], **Henry Schreiner** [2*], **Eduardo Rodrigues** [3], **Jonas Eschle** [4], and **Jim Pivarski** [2*¶]

1 University College London **2** Princeton University **3** University of Liverpool **4** Syracuse University ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.07791

Software

- Review
- Repository
- Archive

Editor: Axel Donath

Reviewers:

- @AnnikaStein
- @rafaelab

Submitted: 12 October 2024
Published: 23 May 2025

Summary

Mathematical manipulation of vectors is a crucial component of data analysis pipelines in high energy physics, enabling physicists to transform raw data into meaningful results that can be visualized. More specifically, high energy physicists work with 2D and 3D Euclidean vectors, and 4D Lorentz vectors that can be used as physical quantities, such as position, momentum, and forces. Given that high energy physics data are not uniform, vector manipulation frameworks or libraries are expected to work readily on non-uniform or ragged data, data with variable-sized rows (or a nested data structure with variable-sized entries); thus, the library is expected to perform operations on an entire ragged structure in minimum passes. Furthermore, optimizing memory usage and processing time has become essential with the increasing computational demands at the Large Hadron Collider (LHC), the world's largest particle accelerator. Vector is a Python library for creating and manipulating 2D, 3D, and Lorentz vectors, especially arrays of vectors, to solve common physics problems in a NumPy-like (Harris et al., 2020) way. The library enables physicists to operate on high energy physics data in a high level language without compromising speed. The library is already in use at LHC and is a part of frameworks, like Coffea (Gray et al., 2023), employed by physicists across multiple high energy physics experiments.

Statement of need

Vector is one of the few Lorentz vector libraries that offer a Pythonic interface but a compiled computational backend, with the others being Coffea's vector module (depends on Vector), PyROOT's (Brun et al., 2020) LorentzVectors and TLorentzVector classes, and HEPvector (Schreiner, n.d.) (deprecated in favor of Vector). Although Vector was written with high energy physics in mind, it is a general-purpose library that can be used for any scientific or engineering application. The library houses a set of diverse backends, three numerical backends for experimental physicists and one symbolic backend for theoretical physicists. These backends are:

- a pure Python object (builtin) backend for scalar computations,
- a NumPy backend for computations on regular collection-type data,
- a SymPy (Meurer et al., 2017) backend for symbolic computations, and
- an Awkward (Pivarski et al., 2018) backend for computations on ragged collection-type data

Moreover, Vector is the first Lorentz vector library to offer multiple computational backends, as well as both numerical and symbolic backends. Furthermore, akin to PyROOT and LorentzVectorHEP.jl (Ling et al., 2023), Vector supports just-in-time compilation through

Numba extensions (Lam et al., 2015), implemented for both the object and Awkward backends. Vector also includes support for JAX (Bradbury et al., 2018) and Dask (Rocklin, 2015) for the Awkward backend, enabling the library to support automatic differentiation and parallel computing, which are required for introducing automatic differentiation in Analysis Grand Challenge (Held & Shadura, 2022) and to meet the computational needs of High Luminosity LHC (Aberle et al., 2020).

Impact

Besides PyROOT's LorentzVectors and TLorentzVector, Vector has become a popular choice for mathematical manipulations in Python-based high energy physics analysis pipelines. Along with being utilized directly in analysis pipelines at LHC (Held et al., 2024; Kling et al., 2023; Qu et al., 2022), the library is also being used in other high energy physics experiments and as a dependency in other user-facing frameworks, such as, Coffea, MadMiner (Brehmer et al., 2020), FastJet (Roy et al., 2023), Spyral (McCann, n.d.), Weaver (Qu et al., n.d.), and pylhe (Heinrich et al., n.d.). The library is also used in multiple teaching materials for graduate courses and workshops. Finally, given the generic nature of the library, it is often used in non-high-energy physics use cases.

Acknowledgements

The work on Vector was supported by NSF cooperative agreements OAC-1836650 (IRIS-HEP) and PHY-2323298 (IRIS-HEP). We would additionally like to thank the contributors of Vector and the Scikit-HEP community for their support.

Reference

Aberle, O., Béjar Alonso, I., Brüning, O., Fessia, P., Rossi, L., Tavian, L., Zerlauth, M., Adorisio, C., Adraktas, A., Ady, M., Albertone, J., Alberty, L., Alcaide Leon, M., Alekou, A., Alesini, D., Ferreira, B. A., Lopez, P. A., Ambrosio, G., Andreu Munoz, P., … Zurbano Fernandez, I. (2020). *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*. CERN. https://doi.org/10.23731/CYRM-2020-0010

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

Brehmer, J., Kling, F., Espejo, I., & Cranmer, K. (2020). MadMiner: Machine learning-based inference for particle physics. *Computing and Software for Big Science*, *4*(1), 3. https://doi.org/10.1007/s41781-020-0035-2

Brun, R., Rademakers, F., Canal, P., Naumann, A., Couet, O., Moneta, L., Vassilev, V., Linev, S., Piparo, D., GANIS, G., Bellenot, B., Guiraud, E., Amadio, G., wverkerke, Mato, P., TimurP, Tadel, M., wlav, Tejedor, E., … Isemann, R. (2020). *Root-project/root: v6.18/02* (Version v6-18-02). Zenodo. https://doi.org/10.5281/zenodo.3895860

Gray, L., Smith, N., Novak, A., Fackeldey, P., Tovar, B., Chen, Y.-M., Watts, G., & Krommydas, I. (2023). *coffea* (Version 0.7.21). https://doi.org/10.5281/zenodo.7733568

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, 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

Heinrich, L., Feickert, M., & Rodrigues, E. (n.d.). *pylhe*. https://doi.org/10.5281/zenodo.1217031

Held, A., Kauffman, E., Shadura, O., & Wightman, A. (2024). Physics analysis for the HL-LHC: Concepts and pipelines in practice with the Analysis Grand Challenge. *EPJ Web of Conferences*, *295*, 06016. https://doi.org/10.1051/epjconf/202429506016

Held, A., & Shadura, O. (2022). The IRIS-HEP Analysis Grand Challenge. *Proceedings of Science*, *ICHEP2022*, 235. https://doi.org/10.22323/1.414.0235

Kling, F., Kuo, J.-L., Trojanowski, S., & Tsai, Y.-D. (2023). FLArE up dark sectors with EM form factors at the LHC forward physics facility. *Nuclear Physics B*, *987*, 116103. https://doi.org/10.1016/j.nuclphysb.2023.116103

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*, 1–6. https://doi.org/10.1145/2833157.2833162

Ling, J., Terwissen, M., & Stewart, G. A. (2023). *LorentzVectorHEP.jl* (Version 0.1.16). https://github.com/JuliaHEP/LorentzVectorHEP.jl

McCann, G. (n.d.). *spyral-utils*. https://github.com/ATTPC/spyral-utils

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A. (2017). SymPy: Symbolic computing in Python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

Pivarski, J., Osborne, I., Ifrim, I., Schreiner, H., Hollands, A., Biswas, A., Das, P., Roy Choudhury, S., Smith, N., & Goyal, M. (2018). *Awkward Array*. https://doi.org/10.5281/zenodo.4341376

Qu, H., Duarte, J., Chao, S., & sunwayihep. (n.d.). *weaver-core*. https://github.com/hqucms/weaver-core

Qu, H., Li, C., & Qian, S. (2022). Particle Transformer for jet tagging. *Proceedings of the 39th International Conference on Machine Learning*, 18281–18292. https://arxiv.org/abs/2202.03772

Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling. *Proceedings of the 14th Python in Science Conference*. https://doi.org/10.25080/Majora-7b98e3ed-013

Roy, A., Pivarski, J., Papageorgakis, C., Duarte, J., Gray, L., Schreiner, H., Kansal, R., Feickert, M., Lieret, K., & ssrothman. (2023). *Scikit-hep/fastjet*. Zenodo. https://doi.org/10.5281/zenodo.7504167

Schreiner, H. (n.d.). *HEPvector: NumPy based vectors for general purpose calculation and physics*. https://github.com/henryiii/hepvector