saransh0701@gmail.com github/Saransh-cpp linkedin/saransh-cpp twitter/saranshchopra7 medium/@whiteviolin

IRIS-HEP Fellowship Proposal

Vector: Constructors, documentation, and benchmarks
Mentors: Henry Schreiner and Jim Pivarski
Saransh Chopra

1 Abstract

Vector is a Python library that allows working with 2D, 3D, and Lorentz vectors, to solve common physics problems in a NumPy-like way. With Vector, users can create vectors in a variety of coordinate systems, including Cartesian, cylindrical, spherical, and any combination of these with time or proper time for Lorentz vectors. In all, there are 12 coordinate systems: {x-y vs ρ - ϕ in the azimuthal plane} × {z vs θ vs η longitudinally} × {t vs τ temporally}.

Vector comes loaded with 3 + 2 backends; a pure Python object backend, a NumPy backend, an Awkward Array backend, an Object-Numba, and an Awkward-Numba backend to leverage JIT (Just In Time) compiled calculations on vectors. Other potential future vanilla backends include Tensorflow and JAX, and other possible future *Numba*-backends include Numba-NumPy.

The user-facing API of Vector is a bit confusing as it does not expose its true API, which are the Python classes. The current API provides wrapper functions (named as *obj*, *arr*, and *awk* functions) to users which are also shown in the internal classes' *__repr__* methods, making these constructor functions even more ambiguous.

Additionally, the vector library lacks proper user as well as API documentation. Currently, there are no detailed explanations in the existing tutorials, and most of the functions and classes do not contain docstrings, which automatically render when displayed on the documentation website. Furthermore, as discussed with the mentors, benchmarking would be a valuable addition to the Vector project, which could then be propagated to all Scikit-HEP packages in the future.

2 Project and deliverables

2.1 Constructors

The current user API of Vector needs to be changed to expose the classes and their constructors to the users. Following this, a user will be able to and will be encouraged to use the class constructors rather than the overloaded *obj*, *arr*, or *awk* functions.

- Exposing all classes and their __init__ methods to the user API.
- Adding keyword arguments to all the __init__ methods.
- Making sure all the backends work with the proposed changes.

2.2 Documentation

The current documentation lacks docstrings (API documentation) and tutorials. This project aims to add most of it from a user's perspective.

- Adding API documentation for the already existing user-facing functionalities.
- Adding API documentation for the functionalities added during the fellowship period.
- Adding tutorials for users.

2.3 Benchmarks

Vector currently has no benchmarks. This project will aim to create a basic benchmark suite for Vector which can then be propagated to other scikit-hep packages in the future.

- Adding benchmarks using *ASV*.
- Automating the benchmarks using GitHub Actions.
- Deploying an easy-to-access website for the benchmarks.

3 Timeline

Time zone: GMT +5:30 hours (Indian Standard Time)
Fellowship duration: June, July, and August (~ 3 months full time)
University examinations: My university exams will take place from the 9th to the 24th of May; therefore, I will be starting the fellowship tenure on the 1st of June.

I will be able to devote 40-45 hours every week to the fellowship throughout June, July, and August. I will be available via all communication channels, including video conferencing, email, and chat. My university will be starting regular offline classes from the 17th of July for my fifth semester, and due to this, my work timings will shift to evenings in IST. This shift might make my work timings slightly less flexible, but I will be available for video conferences throughout the day.

Timeline for June, July and August (12 weeks full time ~ 3 months full time)

- Weeks 1 and 2
  - Get familiar with Vector and its backends.
  - Add and fix the documentation for the existing API.
  - Possibly address issue #136.

- Weeks 3 and 4
  - Get PR #89 merged which had 2D constructors only.
  - Start working on the *Object* backend to expose constructors to the user API (3D and Lorentz).
  - Add tests, documentation, tutorials, and wrap up the *Object* backend.

- Weeks 5 and 6
  - Start working on the *NumPy* backend to expose constructors to the user API.
  - Add tests, documentation, tutorials, and wrap up the *NumPy* backend.

- Weeks 7 and 8
  - Start working on the *Awkward* backend to expose constructors to the user API.
  - Add tests, documentation, tutorials, and wrap up the *Awkward* backend.

- Weeks 9 and 10
  - Buffer period: Add missing documentation and if any documentation was unknowingly missed while implementing the constructors
  - Create basic benchmarks and add CI for the same.
  - Target for the 1.0 release.

- Weeks 11 and 12
  - Add more benchmarks and improve the CI for the same.
  - Create a template-like structure for benchmarks which could be then utilized by other Scikit-HEP projects.
  - If time permits, add more tutorials to the documentation.

---

4 About me

I am Saransh, a sophomore at the **University of Delhi**, pursuing a major in Information Technology and Mathematics with a minor in Computational Biology. In daylight, I work towards my academic skills and professional commitments, and by night, I develop and maintain open-source research software, which I believe are the key to collaborative and reproducible research. Currently, I am responsible for the development and maintenance of PyBaMM (150,000+ installs), BattBot (120+ followers), liionpack (350+ installs), and my contributions range from DeepXDE (250,000+ installs) to Colour (1,300,000+ installs).

In the summer of 2021, I worked as a Google Summer of Code student developer under **PyBaMM**, **NumFOCUS**, where I worked on mathematical modeling of batteries using Python. My other research and development experiences include working under my Mathematics professor Dr. Shoba Bagai (University of Delhi), to solve higher dimensional partial differential equations using Physics-Informed Neural Networks and working as an intern at **AiView** to foster the independence of visually impaired people using Computer Vision. At present, I am more interested in developing the infrastructure of research software, and I currently maintain the infrastructure of PyBaMM and liionpack. I will also be leading a workshop at PyCon Italia this summer titled *"Code coverage through unit tests running in sub-processes/threads: Locally and automated on GitHub"*.

In the near future, I see myself as a graduate student pursuing academic research in Mathematics and Computer Science, and I believe the IRIS-HEP fellowship will help me take a step towards it by giving me a good head-start and an unforgettable research and development experience.