Proposal

Artem Havryliuk

Mentors: *Jim Pivarski, Philip James Ilten*

"Array-Oriented Python Interface for the Pythia Event Generator"

Overview

Pythia is a cornerstone C++ package used in particle physics for simulating high-energy collisions. Although Pythia's current Python interface is excellent for prototyping due to its comprehensive feature set, including bi-directional bindings, it becomes inefficient when analyzing large datasets because of its one-event-at-a-time approach. This project aims to develop an array-oriented Python interface for Pythia, leveraging the Awkward Array library to enhance performance and usability, particularly in interactive environments like Jupyter notebooks. By focusing on a streamlined feature set optimized for speed, this new interface will facilitate more efficient data handling, enabling the usage of Pythia through the Python interface for large-scale production-level simulation.

Background

Existing Interfaces and Challenges

Pythia's current Python interface, implemented using pybind11, allows for interaction with its C++ core. However, the performance is hampered by Python's inherent inefficiency in processing large datasets through iterative loops. The necessity to handle complex, nested data structures—such as groups of particles with attributes like positions and momenta—further complicates this process.

Comparative Case: FastJet

A similar challenge was addressed in a previous project involving FastJet, another critical C++ package for jet clustering in particle physics. By adding an array-oriented Python interface, the project significantly improved data handling efficiency. FastJet's success provides a valuable blueprint for enhancing Pythia.

Steps

- Ease of Installation: Establish a streamlined installation process to ensure the new interface is easy to install and use. Implement a continuous deployment (CD) pipeline that allows users to install the interface with a simple ***pip install pythia***. This setup will facilitate quick feedback from users, as they can test new features by installing pre-releases from PyPI.

- Utilize Awkward Array and LayoutBuilder: Leverage the Awkward Array library and its LayoutBuilder to manage the complex, nested data structures inherent to Pythia's outputs. This will enable efficient data transfer between C++ and Python, forming the foundation for the array-oriented interface.
- Develop an Array-Oriented Interface: Create a Python interface for Pythia that processes data in arrays rather than single events. This approach will drastically improve performance by maintaining data in a numerical format suitable for bare-metal processing, thus avoiding the inefficiencies of Python for loops.

Timeline

Duration	***Task***
2 weeks	● *Review Awkward Array Documentation: Gain a comprehensive understanding of Awkward Array and LayoutBuilder to effectively use these tools*
1 week	● *Study Pythia's C++ API: Understand the current implementation and identify the key areas where data handling can be optimized.*
2 weeks	● *Set Up Continuous Deployment (CD) Pipeline: Establish an automated CD pipeline to streamline the installation process. This will allow for routine releases, enabling users to test new features promptly via pip-installing pre-releases from PyPI.* ● *Design the Interface: Outline the architecture of the new interface, ensuring it can handle arrays of events and particles efficiently.*
1 week	● *Implement Data Structures: Use Awkward Array to represent Pythia's complex, nested data structures in a way that is compatible with Python.*
2 weeks	● *Develop the Interface: Implement the array-oriented interface using pybind11 to connect Pythia's C++ core with Python.*
2 weeks	● *Unit Testing: Conduct thorough testing to ensure the interface handles data correctly and efficiently.* ● *Performance Benchmarking: Compare the performance of the new interface with the existing one-event-at-a-time interface to quantify improvements.* ● *Optimization: Identify and implement optimizations to further enhance performance.*

2 weeks	• *Documentation: Create comprehensive documentation to assist users in installing and using the new interface.* • *Package Distribution: Make the interface available as a pip-installable package.*