# ParticlePhaseSpace: A python package for streamlined import, analysis, and export of particle phase space data

Brendan Whelan [1] **and Leo Esnault** [2]

1 Image-X Institute, School of Health Sciences, University of Sydney **2** Univ. Bordeaux-CNRS-CEA, Centre Lasers Intenses et Applications, UMR 5107, 33405 Talence, France

Summary

In accelerator particle physics, a description of the positions and directions of an ensemble of particles is defined as phase space (Wiedemann (2015)). To describe a given particle at a given time six properties are required, for example, position and momentum: [x, y, z, px, py, pz]. To extend this description to arbitrary particles at any arbitrary point in time and to handle particles of different statistical weight the particle species (e.g. electron, X-ray, etc.), time at which its properties were recorded, and statistical weight of the particle must also be included, resulting in a nine dimensional space. Phase space data is commonly both the input and output of particle accelerator simulations. Unfortunately, there is no widely accepted format for phase space data.

Statement of need

Although the use of phase space data is well established, there is no consistent implementation of phase space data between different programs, as discussed at length in Tessier (2021). To appreciate why this is an issue, one must understand that in a typical accelerator workflow, it is common to utilize several different programs to simulate different phases of particle transport. For any given simulation task, there are many different simulation programs one can use. An incomplete list is supplied in the Wikipedia article "Accelerator Physics Codes" (2022) and the campa consortium data base Consortium for Advanced Modeling of Particle Accelerators (CAMPA) (2022). Each of these programs will typically utilize their own unique format for saving and loading phase space data. This means that getting these programs to 'speak' to each other generally requires a substantial amount of work. The openPMD initiative aims to solve this problem by defining a common open-access metadata standard (Huebl et al. (2015)). However, many codes have yet to adopt this standard, and some commercial codes in particular may never do so. The fragmented nature of phase space data formats has in turn led to fragmented analysis frameworks, with different fields and research groups tending to develop their own, often in-house, code for handling phase space data.

ParticlePhaseSpace aims to solve these issues by providing well documented, extensible mechanisms for the import and export of data in different formats, as well as a library of methods for visualizing, manipulating, characterizing and analyzing phase space data. There are many examples in the scientific literature where ParticlePhaseSpace would be useful, for example, St. Aubin et al. (2010), Whelan et al. (2016), and Esnault (2021). The basic code hierarchy is shown in Figure 1.
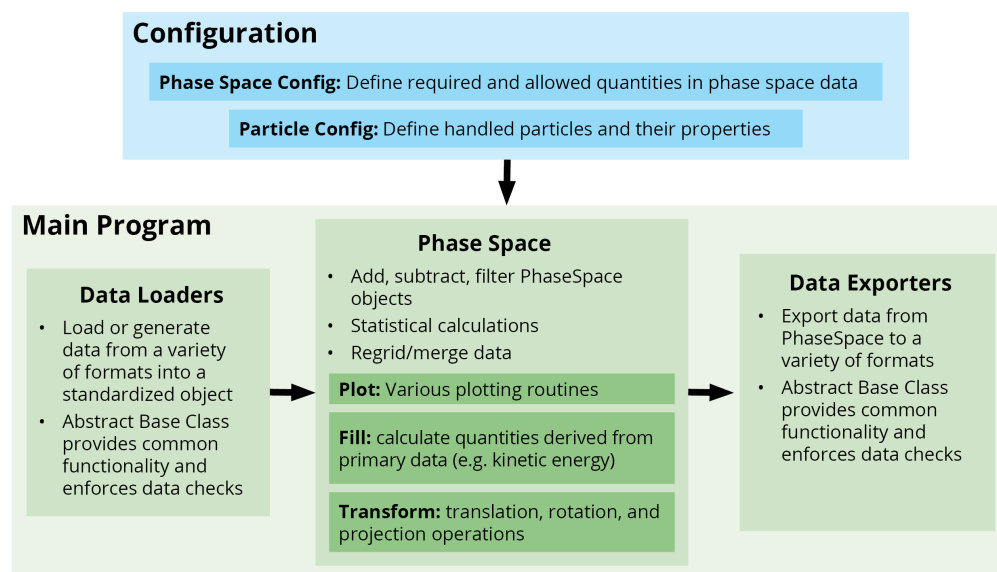

Figure 1: Code structure overview.

Import/Export of different data formats is facilitated through the use of Abstract Base Classes to generate new `DataLoader` and `DataExporter` objects, which ensures consistent data formatting and easy extensibility. The underlying PhaseSpace data is stored in a `pandas` dataframe with configurable and clearly defined required and allowed (optional) quantities. All optional quantities must have an associated method defining their calculation inside `PhaseSpace.fill`. If this method is not defined and callable the code tests will fail. Similarly, all allowed particles and allowed quantities must be documented, or the code tests will fail. The main `PhaseSpace` object contains various methods for plotting (one and two dimensional histograms, scatter plots, etc.), transforming (translation, rotation, and re-gridding), and filtering (boolean index, time) Phase Space data. `PhaseSpace` objects can also be added or subtracted from each other. Users can work with a wide range of pre-defined unit sets as well as define new units. The particles which are handled by this code are defined inside `ParticleConfig` which enables simple extension to arbitrary particles. Figure 2 shows examples of some of the plots which can be generated using `ParticlePhaseSpace` using data from the X-ray collimator described in Whelan et al. (2022).
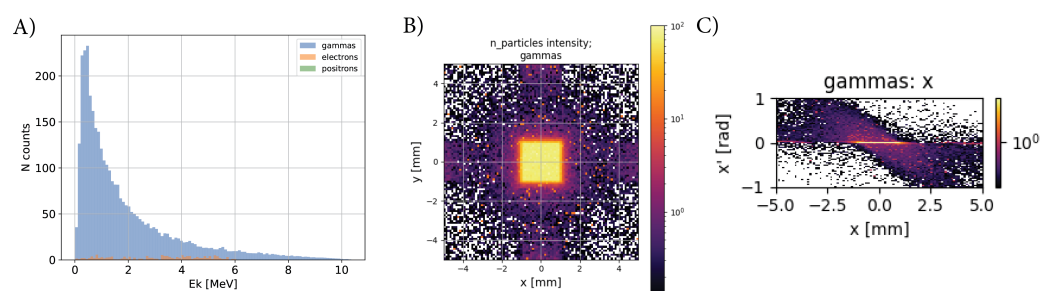

Figure 2: Examples of plots from ParticlePhaseSpace. A) Multi-particle energy histogram. B) 2D intensity histogram of gamma particles. C) Trace-space in x of gamma particles.

There are some additional open source codes providing similar functionality to this code, including `p2sat` (Esnault (2021)) and `postpic` (Kuschel (2014)). The codes Lehe et al.

(2016) and Mayes (2020) have been specifically designed to work with the openPMD standard. `ParticlePhaseSpace` provides an additional tool in this landscape and provides extension mechanisms for loading and exporting arbitrary data formats, a testing framework with continuous integration, multi-particle support in the same Phase Space object, many useful plotting and analysis routines, and automatic code documentation. The major limitation of the code at the time of writing is the ability to handle data larger than memory. Future versions could address this limitation by adding well supported 'chunking' mechanisms or using libraries such as dask Dask Development Team (2016) to enable distributed operations.

Acknowledgements

References

Accelerator physics codes. (2022). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Accelerator_physics_codes&oldid=1099830164

Consortium for Advanced Modeling of Particle Accelerators (CAMPA). (2022). Accelerator Simulation Codes. In *Accelerator Simulation Codes*. https://campa-consortium.github.io/accelerator_simulation_codes/

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. https://dask.org

Esnault, L. (2021). *Particle Phase Space Analysis Toolkit*. https://github.com/lesnat/p2sat

Huebl, A., Lehe, R., Vay, J.-L., Grote, D. P., Sbalzarini, I., Kuschel, S., Sagan, D., Mayes, C., Pérez, F., Koller, F., & Bussmann, M. (2015). *openPMD: A meta data standard for particle and mesh based data*. https://github.com/openPMD; Zenodo. https://doi.org/10.5281/zenodo.591699

Kuschel, S. (2014). *Postpic*. https://github.com/skuschel/postpic

Lehe, R., Huebl, A., Jalas, S., & Thévenet, M. et. al. (2016). *openPMD-viewer*. openPMD. https://github.com/openPMD/openPMD-viewer

Mayes, C. (2020). *openPMD-beamphysics*. https://github.com/ChristopherMayes/openPMD-beamphysics

St. Aubin, J., Steciw, S., Kirkby, C., & Fallone, B. G. (2010). An integrated 6 MV linear accelerator model from electron gun to dose in a water tank. *Medical Physics*, *37*(5), 2279–2288. https://doi.org/10.1118/1.3397455

Tessier, F. (2021). Proposal for an extensible IAEA phase space format · Discussion #712 · nrc-cnrc/EGSnrc. In *GitHub*. https://github.com/nrc-cnrc/EGSnrc/discussions/712

Whelan, B., Gierman, S., Holloway, L., Schmerge, J., Keall, P., & Fahrig, R. (2016). A novel electron accelerator for MRI-Linac radiotherapy. *Medical Physics*, *43*(3), 1285–1294. https://doi.org/10.1118/1.4941309

Whelan, B., Trovati, S., Wang, J., Fahrig, R., Maxim, P. G., Hanuka, A., Shumail, M., Tantawi, S., Merrick, J., & Perl, J. (2022). Bayesian optimization to design a novel x-ray shaping device. *Medical Physics*, *49*(12), 7623–7637. https://doi.org/10.1002/mp.15887

Wiedemann, H. (2015). Particle Beams and Phase Space. In H. Wiedemann (Ed.), *Particle Accelerator Physics* (pp. 213–251). Springer International Publishing. https://doi.org/10.1007/978-3-319-18317-6_8