VISUALPIC: A NEW DATA VISUALIZER AND POST-PROCESSOR FOR PARTICLE-IN-CELL CODES*

A. Ferran Pousa†,[1], R. Assmann, A. Martinez de la Ossa[1], DESY, 22607 Hamburg, Germany
[1] also at Universität Hamburg, 22761 Hamburg, Germany

Abstract

Numerical simulations are heavily relied on for evaluating optimal working points with plasma accelerators and for predicting their performance. These simulations produce high volumes of complex data, which is often analyzed by scientists with individually prepared software and analysis tools. As a consequence, there is a lack of a commonly available, quick, complete and easy-to-use data visualizer for Particle-In-Cell simulation codes. VisualPIC is a new application created with the aim of filling that void, providing a graphical user interface with advanced tools for 2D and 3D data visualization, post-processing and particle tracking. The program is developed under the principles of open source and with a modular design, an approach and architecture which allow interested scientists to contribute by adding new features or compatibility for additional simulation codes.

INTRODUCTION

VisualPIC is a new software for data visualization and analysis specifically designed to work with Particle-In-Cell (PIC) [1] simulation codes, mainly for its application in plasma wakefield acceleration [2].

Figure 1: VisualPIC logo.

The original aim of VisualPIC was to provide a flexible and easy-to-use interface for data analysis, allowing the user to visualize the simulation results without having to write any code. This reduces the need of custom made scripts which, even if very efficient for specific cases, can easily tend to become quite cluttered and unpractical when used as the only tool for data visualization.

The program has been developed as an open-source project and it's written in Python [3] due to the maturity of the available plotting libraries like matplotlib [4] and its cross-platform and open nature. Furthermore, using an object-oriented programming language means that the code can be designed in a modular way, in which different classes perform very specific and well defined tasks. For example, the data-reading process has been isolated into 3 main classes, making the rest of the program virtually independent of the simulation code used to produce the data files. This implies that scientists interested in taking advantage of the features in VisualPIC only have to implement the data readers for their PIC code of choice.

The main capabilities of the program include 2D and 3D visualization of fields and particle data, particle tracking through the simulation, the creation of snapshots and animations, as well as a dedicated visualizer for making eye-catching 3D renders of the simulation.

The program is still on an Alpha development stage and only data readers for the PIC code OSIRIS [5] are implemented, but more are planned for the future.

DEPENDENCIES AND USED LIBRARIES

At the time of writing, the program is designed to run in Python 3.5, using Qt 5.7 [6] for the Graphical User Interface (GUI). The main Python libraries needed to run VisualPIC include:

- NumPy 1.12 with MKL [7].
- SciPy 0.18 [8].
- Matplotlib 2.0.
- PyQt5 [9].
- H5py 2.6 [10].
- VTK 7.0 [11].
- Pillow [12].

Also, the software FFmpeg [13] is used to create the animations and needs to be installed and added to the system PATH.

These libraries and software packages are available for all 3 main operating systems, effectively making VisualPIC a cross-platform application.

DATA READING AND VISUALIZATION

All the user has to do in order to analyze a simulation is to specify the path of the root folder containing the data. Once this is done, the program will automatically scan the folder and populate the GUI with the available data sets.

After that, the user can select what to visualize and easily jump between time steps with a slider placed at the bottom of the window, as seen in Fig. 2.

The individual plots can be edited to change the data units, axes labels, colormaps, etc. and saved in a number of different formats, including PNG and PDF. Also, animations of the data can be easily made with a dedicated tool.

* This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 653782.
† angel.ferran.pousa@desy.de

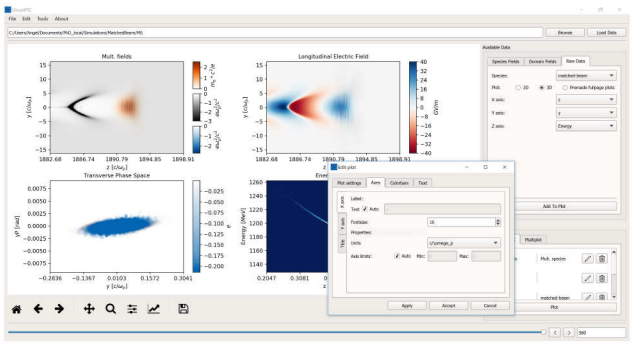

Figure 2: Main VisualPIC interface for data visualization.

PARTICLE TRACKING

PIC codes can typically allow particle tracking by assigning individual labels or tags to the macroparticles in the simulation. This can be useful for studying beam dynamics, laser-plasma interactions, self-injection of electrons, or other processes that might require a detailed study.

In order to take advantage of this feature, VisualPIC automatically detects whether the particle data from a simulation includes tags and offers a specific interface for particle tracking, as seen in Fig. 3.

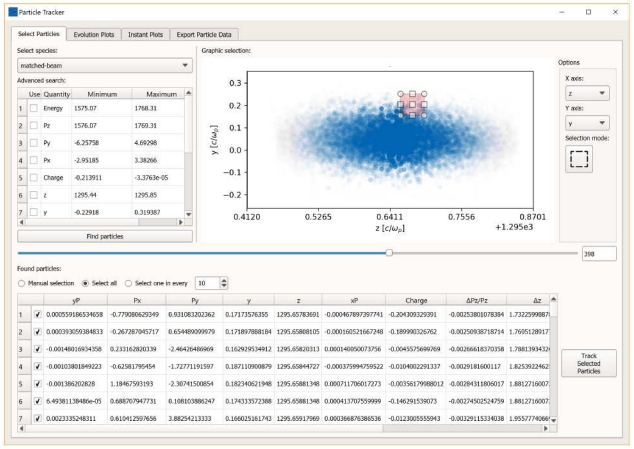

Figure 3: Particle tracking window in VisualPIC. The plot on the top right shows the particle distribution of an electron beam at a certain time step. Particles to track can be selected manually with the rectangle tool on the plot or by specifying the search criteria on the left.

The window for particle tracking allows the user to visualize the state of the species at any time step and offers two different ways of selecting the particles to track:

- Graphic selection: the user can visualize in a 2D scatter plot the state of the species at any time step and in any of the available variables, like position, momentum or energy. Then, by simply drawing a rectangle over the area of interest, a list of the particles contained in there will be populated, allowing their individual selection.
- Precise search: the user can manually specify a range in any of the available quantities and find the particles within the given values. From the obtained list, individual particles can be selected for tracking.

After the selection has been made, the code will go through all the time steps in the simulation and retrieve the information of the tracked particles. This data can then be plotted within the program and exported in HDF5 format [14].

3D VISUALIZATION

VisualPIC also contains a specific module for 3D visualization whose purpose is to create visually appealing 3D renders of the simulation.

One of the features of this module is that the data does not need to be in 3D, instead, the program can also accept data from 2D simulations and efficiently reconstruct a 3D field by assuming cylindrical symmetry.

The 3D renders are made with the VTK library, allowing interactive visualization as well as modifying the visual properties of the fields in real time.

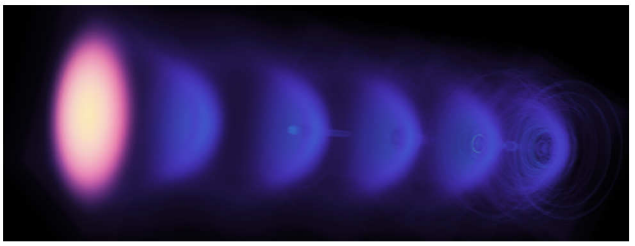

Figure 4: 3D render made from a 2D simulation of a laser-driven plasma accelerating stage. The laser is traveling to the left and is displayed in bright yellow-red. The background plasma can be seen in blue.

SOURCE CODE

VisualPIC is an open-source project and is freely available under the GNU General Public License v3.0. The source code and documentation can be found both on the DESY Stash repository and on GitHub:

`https://stash.desy.de/users/ferran/repos/visualpic`

`https://github.com/AngelFP/VisualPIC`

If you use VisualPIC to produce plots or figures for any scientific work, please provide a reference to this publication.

SUMMARY

VisualPIC is a convenient tool for performing the most common tasks when analyzing data from PIC simulations, as well as producing 3D figures and animations.

The GUI allows users to easily visualize and interact with the data without needing to do any scripting, making the process very quick and agile.

At the time of writing, only OSIRIS data is supported, but thanks to the modular design and open-source nature of VisualPIC, any interested scientists can easily add compatibility for additional PIC codes.

REFERENCES

[1] D. Tskhakaya, K. Matyash, R. Schneider and F. Taccogna, "The Particle-In-Cell Method", Contributions to Plasma Physics, vol. 47, no. 8-9, pp. 563–594, 2007.

[2] T. Tajima and J. M. Dawson, "Laser electron accelerator", Phys. Rev. Lett., vol. 43, no. 4, p. 267, Jul. 1979.

[3] Python Software Foundation. Python Language Reference, version 3.5, `https://www.python.org`.

[4] J. D. Hunter, "Matplotlib: A 2D graphics environment", Computing In Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007.

[5] R. Fonseca et al., "OSIRIS: a three-dimensional, fully relativistic particle in cell code for modeling plasma based accelerators", in *Proc. ICCS 2002*, Amsterdam, The Netherlands, Apr. 2002, pp. 342-351.

[6] Qt: Cross-platform software development for embedded & desktop, `https://www.qt.io`.

[7] S. van der Walt, S. C. Colbert and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation", Computing in Science & Engineering, vol. 13, pp. 22–30, 2011.

[8] E, Jones et. al, "SciPy: Open source scientific tools for Python", `http://www.scipy.org/`.

[9] PyQt: Python bindings for The Qt Company's application framework, `https://www.riverbankcomputing.com/software/pyqt/intro`.

[10] A. Collette, *Python and HDF5*, O'Reilly, 2013.

[11] W. Schroeder et al., *The Visualization Toolkit* (4th ed.), Kitware, 2006.

[12] Pillow: The friendly PIL fork, `https://python-pillow.org/`.

[13] FFmpeg: Cross-platform solution to record, convert and stream audio and video, `https://ffmpeg.org/`.

[14] The HDF Group. "Hierarchical data format version 5", 2000-2017, `http://www.hdfgroup.org/HDF5`.