PlasmaPy: initial development of an open source core Python package for plasma physics

Nicholas A. Murphy, Andrew J. Leonard, Dominik Stańczak, Colby C. Haggerty, Tulasi N. Parashar, Yi-Min Huang, and the PlasmaPy Community

1Harvard-Smithsonian Center for Astrophysics, 2Aperio Software, 3Warsaw University of Technology, 4University of Delaware, 5Princeton University

59th Annual Meeting of the APS Division of Plasma Physics
Milwaukee, Wisconsin, USA
October 23–27, 2017

Link to this poster: http://q-r.to/baoaBr
In recent years, researchers in several different subfields of physics and astronomy have collaboratively developed core Python packages such as Astropy\textsuperscript{1} and SunPy\textsuperscript{2}.

These packages provide core functionality, common frameworks for data analysis and visualization, and educational tools.

A similar open source package for plasma physics would greatly benefit our field.

\textbf{We have begun open development of PlasmaPy, which will be a community-developed and community-driven core Python package for plasma physics.}

\textsuperscript{1}Astropy Collaboration (2013, A&A, 558, 833)
\textsuperscript{2}SunPy Community (2015, CS&D, 8, 014009)
Current status of scientific programming in plasma physics

- Major codes often use low-level languages such as Fortran
- Programmers are often self-taught
- Code is often difficult to read
- Compiling and installing codes is difficult and time-consuming
- Different codes lack interoperability
- Documentation is usually inadequate
- Access to major codes is often restricted in some way
- Somewhat unusual to share code
- Many versions of software do essentially the same thing
- Research is difficult to reproduce

There is a considerable need for open, general-purpose shared software for plasma physics using modern best practices for scientific programming.
Why choose Python?

- Free and open source
- High-level, interpreted language
- Programming style emphasizes readability
- Can “glue” together software written in different languages
- Can reach near-compiled speeds using packages such as Numba and Cython, or by calling compiled routines
- Well-developed numerical and scientific analysis packages
- Active user community
- Can learn from and collaborate with ongoing highly successful projects such as Astropy and SunPy
- Will help students learn programming skills that will be useful in finding employment outside of plasma physics
PlasmaPy is an open source Python 3.6+ package for plasma physics in the early stages of development. The long-term goal of the PlasmaPy community is to facilitate a fully open source Python ecosystem for plasma physics.
PlasmaPy is open source for open and reproducible science

- Some software packages in plasma physics are described as open source, but do not meet the definition\(^3\) set by the Open Source Initiative (OSI) or use an OSI-approved license\(^4\)
- PlasmaPy is under the permissive **BSD 3-clause license** with OSI-approved language to protect against software patents
  - Using a permissive license maximizes compatibility with software under different licenses
  - Permissively licensed code may be incorporated into both proprietary and copyleft software
- Creative works besides source code are usually under Creative Commons licenses
  - The **CC BY 4.0** license allows works to be shared and adapted as long as attribution is given to the original work
  - The **CC BY-SA 4.0** license allows works to be shared and adapted with attribution if derivative works are shared under the same license

\(^3\) Open source definition: [https://opensource.org/osd](https://opensource.org/osd)
\(^4\) OSI-approved licenses: [https://opensource.org/licenses](https://opensource.org/licenses)
PlasmaPy is using best practices for scientific computing\(^6\) to ensure that code is easy-to-use and maintainable

- Simple and intuitive application program interface (API)
- Readable and consistent style (PEP 8 standard)
- Embed documentation in code
- Use modular, object-oriented programming
- Version control with git with useful commit messages
- Avoid prematurely optimizing code
- Use semantic versioning
- Use open Matrix/Gitter channel for real-time communication
- Continuous integration testing with coverage checks
- Issue tracking and code review using GitHub
- Adopt a code of conduct\(^5\) and work toward a welcoming and inclusive community

\(^5\)See CODE_OF_CONDUCT.md in the PlasmaPy repository

\(^6\)Many of these practices are described by G. Wilson et al., “Best Practices for Scientific Computing,” PLOS Biology \textbf{12}, e1001745 (2014)
Organizational development in 2017

- Create organizational infrastructure
  - Set up team communication channels
  - Write vision statement and contribution guide
  - Adopt a code of conduct
  - Appoint the Coordinating Committee
  - Start the PlasmaPy Enhancement Proposals repository
  - Choose license and versioning scheme
  - Begin development roadmap

- Set up documentation structure
  - Enable builds of online documentation using Sphinx that are hosted on Read the Docs
  - Create initial website using Nikola and GitHub Pages

- Plan for release of version 0.1.0 as a prototype and developer’s preview in early 2018.\(^7\)

---

\(^7\)The API of the 0.*.* development releases should be considered unstable. Beginning with version 1.0.0, the API will maintain backward compatibility until the next major release.
Code development began in earnest in April 2017

- Implemented continuous integration testing with Travis CI and coverage testing with Coveralls to find code not covered by tests.
- Began development of core data structures (Plasma class).
- Created atomic subpackage for easy access to atomic data.
- Began physics subpackage for calculating plasma parameters and transport coefficients.
- Began math subpackage for plasma dispersion function, etc.
- Began development of plasma simulation capabilities.
- Implemented a particle pusher.

PlasmaPy’s entire code development history is openly available on our GitHub repository.
PlasmaPy uses the astropy.units package for units

This package creates Quantity objects with attached units.

```python
>>> from astropy import units
>>> distance = 44*units.imperial.mile
>>> time = 30*units.minute
>>> distance/time
<Quantity 88.0 mi / h>
>>> (distance/time).to(units.m/units.s)
<Quantity 39.33952 m / s>
>>> (1.21*units.GW).cgs
<Quantity 1.21e+16 erg / s>
>>> 2*units.m/units.s + 4*units.m/units.s**2
UnitConversionError: Can only apply 'add' function to quantities with compatible dimensions
```

Built-in equivalencies can handle non-standard unit conversions commonly used in plasma physics:

```python
>>> kT = 1.2*units.keV
>>> kT.to(units.K, equivalencies=units.temperature_energy())
<Quantity 13925426.47248121 K>
```

---

8Code inside PlasmaPy strictly uses SI units to avoid confusion and for consistency with established international practices.
Review of PlasmaPy: “Your code has documentation.”

```python
def plasma_dispersion_func(zeta):
    r"""Calculate the plasma dispersion function

    Parameters
    --------
    zeta : complex, float, ndarray, or Quantity
        Argument of plasma dispersion function.

    Returns
    -------
    Z : complex, float, or ndarray
        Value of plasma dispersion function.

    Raises
    ------
    TypeError
        If the argument is invalid.
    UnitsError
        If the argument is a Quantity but is not dimensionless
    ValueError
        If the argument is not entirely finite

    Notes
    -----
    The plasma dispersion function is defined as:
    .. math::
        Z(\zeta) = \sqrt{\pi}\int_{-\infty}^{+\infty}dx
            \frac{e^{-x^2}}{x-\zeta}

    where the argument is a complex number
    [fried.conte-1961].

    In plasma wave theory, the plasma dispersion function appears frequently when the background medium has a Maxwellian distribution function. The argument of this function then refers to the ratio of a wave's phase velocity to a thermal velocity.

    References
    --------
    .. [fried.conte-1961]
    Fried, Burton D. and Samuel D. Conte. 1961.

    Examples
    ------
    >>> plasma_dispersion_func(1j)
    0.75787215614131187j
    >>> plasma Dispersion_func(-1.52+0.47j)
    (0.6088889572342553+0.33494583882874029j)

    ▶ We use the numpydoc docstring format and sometimes put more effort into writing documentation than writing code.
```
What does PlasmaPy need to succeed?

- Open development
  - Low barrier to entry
  - Inviting new contributors
  - Open data policies for major experiments
- A welcoming and inclusive environment
  - Provide a culture of appreciation for contributors to PlasmaPy
  - Use a code of conduct
- A sustainable funding model\(^9\)
  - Astropy development is mostly a volunteer, grassroots effort
  - Most work on Astropy has been done by graduate students and postdocs, with little direct funding support
  - There is a need for funding agencies and large institutions to support open development of general purpose software

---

\(^9\)This issue is described thoroughly by D. Muna et al. in *The Astropy Problem* (arXiv:1610.03159)
We have begun open development of PlasmaPy, which will be a community-developed and community-driven core Python package for plasma physics.

Our goals for the next year include:

- Release prototype version 0.1.0 in early 2018
- Add fluid and particle simulation capabilities
- Develop analysis tools for experimental and space data
- Expand core functionality (e.g., a Grad-Shafranov solver, a dispersion solver, and topology analysis tools)
- Join NumFOCUS and organize Software Carpentry workshops

New contributors are welcome and can become involved by:

- Joining our email list and conversation on Matrix
- Raising issues on GitHub with new ideas for code development
- Contributing code, such as those labeled "Good first contribution"
- Contributing documentation
- Becoming an early adopter and providing feedback
PlasmaPy Websites

- PlasmaPy’s GitHub repository is:
  
  https://github.com/PlasmaPy/plasmapy

- Our Matrix channel for real-time communication is:
  
  https://riot.im/app/#/room/#plasmapy:matrix.org

- Sign up for the PlasmaPy email list at:
  
  https://groups.google.com/d/forum/plasmapy

- We are developing online documentation at:
  
  http://plasmapy.readthedocs.io/en/latest/

- We are developing our webpage at:
  
  http://www.plasmapy.org/