PlasmaPy: beginning a community developed Python package for plasma physics

Nicholas A. Murphy,[1] Yi-Min Huang,[2]
and the PlasmaPy Community

[1]Harvard-Smithsonian Center for Astrophysics
[2]Princeton University

58th Annual Meeting of the APS Division of Plasma Physics
San Jose, California, USA
October 31 – November 4, 2016

Introduction

- **We propose the open development of PlasmaPy: a community-developed and community-driven core Python package for plasma physics**
- In recent years, researchers in several different subfields of physics and astronomy have collaboratively developed core Python packages such as Astropy,[1] SunPy,[2] and SpacePy[3]
- These packages provide core functionality, common frameworks for data analysis and visualization, and educational tools
- A similar package for plasmas would greatly benefit our field
- The goals of this poster are to:
  - Make the case for the creation/open development of PlasmaPy
  - Recruit plasma physicists to join the PlasmaPy project from the very beginning

[1] Astropy Collaboration (2013, A&A, 558, 833)
[2] SunPy Community (2015, CS&D, 8, 014009)
[3] Morley et al. (2014, ASCL:1401.002)

Current status of scientific programming in plasma physics

- Major codes often use low-level languages such as Fortran
- Programmers are often self-taught
- 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 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 C or Fortran routines
- Well-developed numerical and scientific analysis packages
- Active user community
- Can learn from and collaborate with ongoing highly successful projects such as Astropy, SunPy, and SpacePy
- Will help users learn programming skills that will be useful in finding employment outside of plasma physics

PlasmaPy will use best practices for scientific computing[4] to ensure that code is easy-to-use and maintainable

- Simple and intuitive application program interface (API)
- Readable and consistent style (such as PEP 8 standard)
- Embed documentation in code
- Use modular, object-oriented programming
- Version control with git and GitHub
- Avoid prematurely optimizing code
  - Use high-level languages when possible
- Use Slack for text-based chat team communication and community building, with in-person development meetings
- Use automated unit testing, issue tracking tools, and pre-merge code reviews
- Ensure that the community is welcoming and inclusive

[4] G. Wilson et al., "Best Practices for Scientific Computing," PLOS Biology **12**, e1001745 (2014)

Initial development plan

- Short-term development priorities
  - Create a `plasma` class that allows easy calculation of plasma parameters (using `units` module from Astropy)
  - Implement commonly used analytical functions
  - Create simple tools for analyzing magnetic field data
- Long-term development possibilities
  - Standardize data representations
  - Build tools for analysing and visualizing experimental results
  - Implement a flexible Grad-Shafranov solver
  - Incorporate easy-to-use fluid and particle-in-cell simulation capabilities
  - Design tools for the analysis of magnetic topology
- Follow Astropy model by using main package for core functionality, and affiliated packages for extensions
- The development plan is still under development, so please share ideas!

Goals for upcoming year

- Recruit team members from a variety of subfields within plasma physics
- Host initial discussions on Slack and telecons
- Survey existing Python software for plasma physics, and contemplate ways to unify efforts
- Figure out short-term and long-term development plans, and begin development in earnest
- Decide on an organizational structure and open source-license
- Implement unit testing (for example, with Travis CI)
- Have an in-person development meeting
- Find long-term funding mechanisms
- Host a Python training or Software Carpentry workshop at next year's APS DPP meeting?

What does PlasmaPy need to succeed?

- Open development
  - Need a critical mass of developers
  - Low barrier to entry
- A welcoming and inclusive environment
  - Provide a culture of appreciation for contributors to PlasmaPy
  - Use the Contributor Covenant[5] as the initial code of conduct and anti-harassment policy
- A sustainable funding model[6]
  - 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

---

[5] Online at `http://contributor-covenant.org/version/1/4/`

[6] This issue is described thoroughly by D. Muna et al. in *The Astropy Problem* (arXiv:1610.03159)

Becoming involved

- Please contact Nick Murphy at `namurphy@cfa.harvard.edu` or Yi-Min Huang at `yiminh@princeton.edu` to join the PlasmaPy team on Slack and email list
  - People at all levels of experience with Python are welcome
- GitHub repository:

  `https://github.com/PlasmaPy/`
- Sign up for the PlasmaPy email list at:

  `https://groups.google.com/d/forum/plasmapy`
- The website will eventually most likely be:

  `http://www.plasmapy.org`

Summary

- **We propose that our community begins open development of PlasmaPy: a core Python package for plasma physics**
- PlasmaPy should be useful for experimentalists, theorists, numericists, and observers in plasma astrophysics, space physics, heliophysics, and laboratory plasma physics
- The success of PlasmaPy depends on active community participation, a welcoming and inclusive environment, and a sustainable funding/development model

Acknowledgements: The authors are incredibly grateful to all who have contributed to the long history of open source scientific software. We thank Sumana Harihareswara for advice on beginning an open source project, and acknowledge helpful discussions with many friends and colleagues in plasma physics, solar & space physics, and astronomy. The authors acknowledge support from the NSF/DOE Partnership in Basic Plasma Science and Engineering through a component of DOE grant DE-SC0016363.