doi:10.5281/zenodo.2578277

Building an open source software ecosystem for cross-disciplinary plasma research and education

Nicholas A. Murphy,[1,2,a)] Dominik Stańczak,[1,3] Andrew J. Leonard,[1,4] Tulasi N. Parashar,[1,5] Pawel M. Kozlowski,[6] B. L. Alterman,[7] D. Aaron Roberts,[8] S. D. Christe,[8] Martin Connors,[9] Monica G. Bobra,[10] James Paul Mason,[8] Will Barnes,[11] Ryan M. McGranaghan,[12] Asti Bhatt,[13] Philip J. Erickson,[14] Frank D. Lind,[14] Ryan Volz,[14] John Swoboda,[14] Nick Hatzigeorgiu,[15] Andrew Inglis,[8] Felipe Nathan deOliveira-Lopes,[16] Jack Ireland,[8] John C. Coxon,[17] Sophie A. Murray,[18,19] Japheth N. Yates,[20] Mark C. M. Cheung,[21,22] Jeff Klenzing,[8] David Stansby,[23] Han He,[24] Yi-Min Huang,[25] Chuanfei Dong,[25] Henry Winter,[2] Juan-Camilo Buitrago-Casas,[15] Manjit Kaur,[26] Sterling Smith,[27] Benjamin Dudson,[28] Daniel B. Seaton,[29,30] Luca Comisso,[31,32] Alexa J. Halford,[33] D. H. Barnak,[6] R. S. Weigel,[34,35] A. Tavant,[36] Jon D. Vandegriff,[37] Miguel de Val-Borro,[8] and Antonia Savcheva[2]

[1)] *PlasmaPy Coordinating Committee*

[2)] *Center for Astrophysics | Harvard & Smithsonian*

[3)] *University of Warsaw*

[4)] *Aperio Software*

[5)] *University of Delaware*

[6)] *Los Alamos National Laboratory*

[7)] *University of Michigan*

[8)] *NASA Goddard Space Flight Center*

[9)] *Athabasca University*

[10)] *Stanford University*

[11)] *Rice University*

[12)] *Atmospheric and Space Technologies Research Associates (ASTRA) LLC*

[13)] *SRI International*

[14)] *MIT Haystack Observatory*

[15)] *Space Sciences Laboratory, University of California Berkeley*

[16)] *Max Planck Institute for Plasma Physics*

[17)] *University of Southampton*

[18)] *Trinity College Dublin*

[19)] *Dublin Institute for Advanced Studies*

[20)] *European Space Agency*

[21)] *Lockheed Martin Solar & Astrophysics Laboratory*

[22)] *Hansen Experimental Physics Laboratory, Stanford University*

[23)] *Imperial College London*

[24)] *National Astronomical Observatories, Chinese Academy of Sciences*

[25)] *Department of Astrophysical Sciences, Princeton University*

[26)] *Department of Physics & Astronomy, Swarthmore College*

[27] *General Atomics*

[28] *University of York, UK*

[29] *Cooperative Institute for Research in Environmental Sciences, University of Colorado*

[30] *National Centers for Environmental Information, National Oceanic and Atmospheric Administration, Boulder, Colorado*

[31] *Department of Astronomy, Columbia University*

[32] *Columbia Astrophysics Laboratory, Columbia University*

[33] *The Aerospace Corporation*

[34] *Department of Physics and Astronomy, George Mason University*

[35] *Space Weather Laboratory, George Mason University*

[36] *Laboratoire de Physique des Plasmas, CNRS, Ecole polytechnique, Saclay, France*

[37] *Johns Hopkins University Applied Physics Laboratory*

[a] Email: namurphy@cfa.harvard.edu

PUBLIC DOMAIN

I. SCIENTIFIC SOFTWARE FOR PLASMA PHYSICS

Software is crucial to all areas of modern plasma science research. Laboratory plasma physicists use software to interpret plasma diagnostics, analyze experimental results, and glean insights using advanced techniques from data science. Space scientists use software to reduce and understand *in situ* observations. Numericists use software to simulate the behavior of laboratory, heliospheric, and astrophysical plasmas, and then analyze or visualize the results. Theorists use symbolic manipulation software to perform or check derivations. Cross-disciplinary research and cross-comparisons between experiments, observations, simulations, and theories all require software. **Despite our heavy reliance on software, funding agencies have traditionally had few avenues available to support development of general purpose research software infrastructure.**[1]

The lack of investment in community-wide software infrastructure has resulted in several adverse consequences. Different researchers and groups frequently duplicate software with essentially the same functionality. These different software packages typically lack interoperability, thus impeding interdisciplinary and cross-disciplinary collaboration. Access to these codes is frequently restricted in some way. Packages tend to be difficult to install, especially if they depend on external libraries. Pressure to publish leads to code written in a rush to get the next research paper out. Documentation is frequently a low priority. Packages often lack a testing framework. Most scientific programmers are self-taught, and constant time pressure prevents us from taking time to learn new programming skills and best practices for scientific computing. The combination of all of these factors leads to the following consequences.

1. **It is difficult for newcomers to begin research in plasma physics because of the state of software.**
2. **Cross-disciplinary research is hampered due to the lack of interoperability.**
3. **Reproducing plasma research is difficult and time-consuming.**

Over the past decade, researchers in different scientific disciplines have collaboratively developed open source Python packages for their fields.[2] These packages provide essential functionality, frameworks for data analysis and visualization, and educational resources. Projects such as Astropy[3] are fundamentally transforming the way scientific research is being done in astronomy by making it more open, collaborative, and reproducible. The Astropy core package contains the functionality needed by most astronomers across subfields. Astropy's affiliated packages contain more specialized functionality. Astropy's coordination committee regulates intercompatibility among the core and affiliated packages. **Astropy and its affiliated packages constitute a software ecosystem for astronomy.**

Solar physicists are following Astropy's model for open development through SunPy[4] and affiliated packages. SunPy is intercompatible with Astropy to enable cross-disciplinary studies. The Python in Heliophysics Community began in 2018 to promote interoperability and reduce duplication of functionality to create an open source software ecosystem for the whole heliosphere.[5] NASA provided seed funding for the first coordination meeting.

The open source revolution that is well underway in astronomy and heliophysics has only just begun in plasma science. The PlasmaPy project began in 2017 as a community effort to create a shared Python package for plasma physics. PlasmaPy[6] aims to provide the foundation for an open source software ecosystem for plasma science that is intercompatible with the astronomy and heliophysics ecosystem. The version 0.1 development release of PlasmaPy in 2018 serves an invitation to the plasma physics community to contribute to this project.[7] **The continued development of an open source software ecosystem for plasma science hinges on both community participation and stable funding.**

Because of the importance of open scientific software infrastructure, we suggest a series of recommendations for the plasma science decadal review. For more contextual information and related recommendations, we encourage the committee to refer to Refs. 8–11.

II. PROPOSED RECOMMENDATIONS

Conclusion: Software infrastructure is a critical enabling technology for modern scientific research. In order for research to be reproducible, the software used to analyze results must be openly available for use, modification, and redistribution. Plasma physics will not be able to perform at its full potential without investment in open source software infrastructure using modern best practices for scientific programming.

A fully open source software ecosystem for plasma physics research and education will have numerous benefits. Research will be more reproducible and transparent by making software open for inspection and modification. Adopting best practices for scientific programming will improve code readability, reliability, and maintainability. Barriers to entry for newcomers to the field will be reduced by providing thorough documentation, access to an active user/developer community, and an intuitive interface. Students and scientists will be introduced to collaborative code development practices that are widely used in industry. Costly duplication of functionality will decrease. Interoperability between different software packages will improve, thus enabling both interdisciplinary and cross-disciplinary collaborations. An open, well-documented code base will reduce the financial burden associated with developing software for new experiments.

Recommendation: The plasma physics community should collaboratively develop an open source software ecosystem for research and education.

The creation of this software ecosystem should follow the open development model used by Astropy. Software should be written using modern best practices for scientific code development,[12,13] such as prioritizing readability, implementing a straightforward and intuitive user interface, optimizing code only if necessary, using a version control system, using continuous integration testing, adopting test-driven development, turning bugs into test cases, performing code reviews, and producing useful error messages. Documentation should be prioritized, and be clear and understandable enough for a student taking their first class in plasma physics. Open source projects should create materials that can be used in plasma education, including examples in formats such as Jupyter notebooks. Packages in this ecosystem must be released under a license approved by the Open Source Initiative (OSI). All projects must have a code of conduct.

Python is recommended as a primary programming language of this software ecosystem for the following reasons.[14] The scientific Python community is highly active and there exist many well-developed Python packages for numerical and scientific analysis and visualization. Choosing Python will enable intercompatibility with the existing Python ecosystems for astronomy and heliophysics. Python can be used without paying the subscription fees often required for proprietary languages. Python can call code written in Fortran, C, C++, and Julia. High performance can be achieved by calling compiled code or by using a just-in-time compiler. These features have made Python one of the most commonly used languages in industry, and have brought Python to the forefront of the data science and Big Data revolutions. Building this software ecosystem using Python will enable use of these technologies in plasma science and equip students with the tools necessary to tackle the technological challenges of this century.

Recommendation: Funding agencies should invest in open source software infrastructure for plasma physics and related fields.

A longstanding problem has been that funding agencies have had few mechanisms to support open development and maintenance of research software infrastructure.[1] Agencies should create, modify, or expand funding opportunities to fulfill this community need.

DOE and NSF should establish a new competitive funding opportunity with the goal of developing and sustaining open source plasma science software infrastructure.[15] Each awardee would be responsible for developing and maintaining a component of a new open source software ecosystem. Awardees would be required to form a team and work together to coordinate interoperability (including with software packages widely used in fields related to plasma physics), reduce software duplication, identify and address gaps in functionality, establish community code development guidelines, and seek feedback from the broader plasma physics community. This team should hold biannual in-person meetings, with representation from funding agencies. These meetings and other communications should be open to members of the broader plasma community, including prospective users and developers.

Software that is not actively maintained will gradually decrease in usability and usefulness due to changes in external dependencies, evolving user needs, and lost expertise. Open source projects that become essential to the community should be supported by longer-term contracts in order to provide stability and enable continued development.

Recommendation: Software projects, funding agencies, and the broader plasma science community should jointly take responsibility for software sustainability.

Software sustainability describes the technical, institutional, and cultural practices that allow software to continue operating as expected in the future. Software sustainability increases research trustworthiness, scientific return on investment, and the rate of discovery.[9] The responsibility for software sustainability should be shared among all stakeholders.

Individual projects should support software sustainability by creating transparent organizational infrastructure, involving the community, providing guidance to new contributors, releasing sufficient documentation, making their software products citable and discoverable, and fostering a welcoming and inclusive environment.

Funding agencies should act as stewards of the open source software ecosystem for plasma physics. These agencies should regularly evaluate the status of this software ecosystem and ensure that important packages do not become abandoned. They should identify gaps in functionality with help from the community, and tailor funding opportunities to address these gaps. DOE should clarify guidance on export control restrictions related to making software for plasma science open source.

Research and educational institutions should enact policies that enable contributions to open source projects under any license approved by OSI. Journals should encourage authors to cite important software used in a project. Authors should archive data and software used in publications so that they become citable. Institutions should hold workshops that cover best practices for scientific computing such as those by the Carpentries. Contributions to open source software projects and efforts to make research reproducible should be recognized as strengths in decisions on hiring and tenure.

Recommendation: Funding agencies and the greater plasma physics community should invest in data science innovations enabled by an open and collaborative software ecosystem.

The growth in size and complexity of data sets produced by plasma physics experiments, observations, and simulations highlights how critical openly available data science tools are to the infrastructure of plasma physics research.[16,17] Examples of data science tools which could be built within an open ecosystem include, but are not limited to: automated and reproducible feature detection and tracking in images (such as x-ray radiographs), rapid exploration of multi-dimensional parameter problems (as in designing inertial confinement fusion experiments), automated and reproducible processing of atomic spectra, big data processing using results from high repetition rate facilities, and searches for unexpected patterns in large data sets. Automation efforts would save scientists time from conducting repetitive tasks, and would enable real-time analysis of data during experimental campaigns. The development of these tools would integrate existing open source data science libraries such as scikit-learn, TensorFlow, and Keras into the plasma science software ecosystem. This effort would further the role of plasma science in pushing the frontiers of data science and computational techniques.

III. FINAL THOUGHTS

A software ecosystem for plasma research and education will be most beneficial if it is compatible with open source frameworks for related fields such as astronomy[3] and heliophysics.[4,5] Plasma science would benefit from functionality that already exists in other fields, and related fields would benefit from functionality that we create. The resulting intercompatibility would simplify cross-disciplinary research. The fusion plasma community would ideally participate as well.

Plasma education will benefit from investment in open source software infrastructure. Well-documented software with an intuitive interface will let students hit the ground running when beginning their first research projects. The need to re-write functionality that already exists but is hard to read, inadequately documented, and/or closed source will be reduced. Students who enter industry upon graduation will have a job search advantage if they have experience with Python and collaborative code development practices. Examples provided by open source projects may be used in courses to help students understand plasma phenomena.

Even if an open source ecosystem is not funded, research software will continue to be paid for through various grants and projects. Indeed, it will be paid for multiple times as different groups duplicate, triplicate, and quadruplicate the same functionality. Packages that are created will continue to lack interoperability. If the decadal review is silent on general-purpose research software needs, then it risks committing our field to not funding open research software infrastructure for the next ten years.[1] **A software ecosystem for plasma research and education is a sound investment for our community with very tangible benefits and few drawbacks that will reduce the time, effort, and frustration needed to reach scientific understanding.**

REFERENCES

[1]D. Muna et al., "The Astropy Problem," arXiv:1610.03159 (2016).

[2]Examples include Astropy for astronomy; SunPy for solar physics; HelioPy, SpacePy, and pysat for space physics; MetPy for meteorology; SatPy for earth-observing satellite data processing; ObsPy for seismology; pyQuil for quantum computing; MDAnalysis for molecular dynamics; and QuantEcon for economics.

[3]Astropy Collaboration et al., "The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package," Astronomical Journal **156**, 123 (2018), see also `astropy.org`.

[4]SunPy Community et al., "SunPy – Python for solar physics," Computational Science and Discovery **8**, 1 (2015), see also `sunpy.org`.

[5]A. Burrell et al., "Snakes on a Spaceship – An Overview of Python in Heliophysics," Journal of Geophysical Research **123**, 10384 (2018), see also `heliopython.org`.

[6]PlasmaPy Community et al., "PlasmaPy: an open source community-developed Python package for plasma physics," 10.5281/zenodo.1238132 (2018), Zenodo, see also `docs.plasmapy.org`.

[7]Please let us know if you have ideas or would like to contribute!

[8]National Academies of Sciences, Engineering, and Medicine, *Open Source Software Policy Options for NASA Earth and Space Sciences* (National Academies Press, 2018).

[9]S. Hettrick, "Research Software Sustainability: Report on a Knowledge Exchange Workshop," (2016).

[10]Mark D. Wilkinson et al., "The FAIR Guiding Principles for scientific data management and stewardship," Scientific Data **3**, sdata201618 (2016).

[11]Engineering and Physical Sciences Research Council, "Software as an infrastructure," (2012).

[12]G. Wilson et al., "Best Practices for Scientific Computing," PLOS Biology **12**, 1–7 (2014).

[13]A. Scopatz and K. D. Huff, *Effective Computation in Physics: Field Guide to Research with Python* (O'Reilly Media, 2015).

[14]Julia, which to our knowledge is the only interactive dynamically typed language to achieve petascale computing, is also a valid choice. Because Julia can be called from Python and Python can be called from Julia, these two languages can co-exist in a software ecosystem.

[15]This suggested opportunity is inspired by NASA's Living With a Star Targeted Research & Technology program and PyHC.

[16]B. K. Spears et al., "Deep learning: A guide for practitioners in the physical sciences," Physics of Plasmas **25**, 080901 (2018).

[17]D. R. Smith et al., "Highlights from the community white paper 'Enhancing US fusion science with data-centric technologies'," in *APS DPP Meeting Abstracts* (2018).