





PlasmaCalcs: A Consistent Interface for Python Plasma Calculations from Any Inputs

Samuel Evans¹, Rattanakorn Koontaweepunya¹, Alexander Green¹,
and Tess Goodman¹

¹ Boston University Center for Space Physics, United States  Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: 

Submitted: 08 September 2025

Published: unpublished

License

Authors of papers retain copyright
and release the work under a
Creative Commons Attribution 4.0
International License ([CC BY 4.0](#)).

Summary

Plasma makes up 99.9% of the observable universe, and plasma physics drive a wide range of phenomena, from sparks to lightning bolts, stellar formation to supernovae, and planetary atmospheres to interactions in interstellar space ([Contemporary Physics Education Project, 2011](#); [NASA, 2021](#)). Plasma is a state of matter involving at least some charged particles, which can interact with electric and magnetic fields. Vast amounts of research at the forefront of human knowledge involve analyzing theory and data related to plasma physics. PlasmaCalcs provides a consistent interface (in Python) for plasma calculations from any inputs.

Statement of need

Plasma physics contains a myriad of quantities, mathematical operations, and potential data sources. *Quantities* include: electric and magnetic field, mass, charge, density, velocity, temperature, collision frequency, magnetization parameter, Debye length, collisional mean free path, thermal speed, zeroth-order values such as equilibrium temperatures, elemental abundances, and many more. *Operations* include: spatial and temporal derivatives, vector magnitude, cross products and dot products, Fourier transforms, and interpolation from lookup tables. *Data sources* include: observations from spacecraft or telescopes; outputs from various types of simulators, including kinetic codes (e.g., EPPIC ([Oppenheim & Dimant, 2004](#))), multifluid codes (e.g., Ebysus ([Martínez-Sykora et al., 2020](#))), and single-fluid radiative magnetohydrodynamic (MHD) codes (e.g., Bifrost ([Gudiksen et al., 2011](#)), MURAM ([Vögler et al., 2005](#)); and any arbitrary dataset (e.g., an artificially-constructed grid of relevant physical parameters).

Complicating matters further, quantities often have different dimensionality in different data sources. For example, magnetic field is constant in EPPIC, but varies in time and space (which may itself be 2D or 3D) in Bifrost. Also, EPPIC number density varies from species to species, while single-fluid codes like Bifrost instead track the “averaged” single-fluid number density.

PlasmaCalcs helps to manage all of this complexity. This package provides a consistent interface for loading data from any kind of input, has many different available calculations relevant to plasma physics, supports well-labeled multidimensional arrays via xarray ([Hoyer & Hamman, 2017](#)), features many convenient capabilities for data manipulation and exploration, and contains well-documented custom-built help systems. If you use Python to analyze any data related to plasma physics, then this package is for you!

One of the biggest advantages to using PlasmaCalcs is that it separates data inputs from the quantities themselves. For example:

```
import PlasmaCalcs as pc
cc = pc.BifrostCalculator(...) # or, e.g., EppicCalculator or EbysusCalculator
```

`cc('ldebye')`

will get the Debye length (one of the fundamental length scales in plasma physics). The user doesn't need to know that Debye length depends on number density (n) and temperature (T), figure out how to load n and T , and remember which function to plug those values into. The user also doesn't need to consider the number of species involved, nor whether n and T vary across different dimensions; the syntax to get the Debye length is the same for all kinds of input. This greatly reduces the mental load required to do analysis. Users are free to ask interesting questions such as "does my simulation resolve the Debye length?" without always needing to consider more basic questions like "how do I compute the Debye length?" at every step of the way.

Existing packages help with some of these tasks, but PlasmaCalcs provides notable advantages:

- `plasma` ([PlasmaPy Community et al., 2024](#)) provides many plasma formulas, however those formulas all require the user to directly input the relevant physical quantities (e.g., `plasma.formulary.lengths.Debye_length` requires `T_e` and `n_e`).
- `xarray` ([Hoyer & Hamman, 2017](#)) provides a powerful interface for analyzing multidimensional arrays, and is used extensively by PlasmaCalcs. However, PlasmaCalcs provides many methods and quantities specific to plasma physics. PlasmaCalcs also provides additional convenient behaviors for plotting (e.g., put "x" on the x-axis by default) and tools for creating animations (e.g., `arr.pc.image().ani()`), which greatly facilitate exploration of multidimensional data. Additionally, PlasmaCalcs provides convenient access to useful scientific operations missing from `xarray`, such as `arr.pc.fft()` to take the Fourier transform.
- `sunpy` ([The SunPy Community et al., 2020](#)) provides many tools relevant to solar plasma physics, but focuses primarily on acquiring, directly inspecting, and visualizing data, rather than computing a wide variety of plasma quantities.
- `helita` provides interfaces for computing quantities from a variety of input sources, including Ebysus, Bifrost, and MURAM. PlasmaCalcs improves upon `helita` in a variety of ways, including:
 - a more detailed class hierarchy, which reduces repetitive code and encapsulates methods more intuitively.
 - defining one function per quantity, which allows for easier introspection.
 - making fewer assumptions about the underlying data structure (e.g., data does not need to be 3D).

Research Projects Utilizing PlasmaCalcs

PlasmaCalcs has contributed to a variety of recent scientific research endeavors by the authors, with corresponding manuscripts currently being prepared for publication. It has also been utilized in several recent publications studying plasma instabilities, including Evans et al. (2025) and Koontaweepunya et al. (2024).

Acknowledgements

We acknowledge Meers Oppenheim, Juan Martínez-Sykora, and Yakov Dimant, for many helpful conversations which contributed to our understanding of plasma physics and informed some of our choices for which plasma physics quantities to implement while developing this project. We also acknowledge the developers of the `helita` package, which inspired various aspects of the PlasmaCalcs code design.

References

- Contemporary Physics Education Project. (2011). *Fusion: Physics of a Fundamental Energy Source*. <https://newsite.cpepphysics.org/elementor-1179/>
- Evans, S., Oppenheim, M., Martínez-Sykora, J., & Dimant, Y. (2025). Multifluid and Kinetic 2D and 3D Simulations of Thermal Farley–Buneman Instability Turbulence in the Solar Chromosphere. *986*(1), 23. <https://doi.org/10.3847/1538-4357/adcd70>
- Gudiksen, B. V., Carlsson, M., Hansteen, V. H., Hayek, W., Leenaarts, J., & Martínez-Sykora, J. (2011). The stellar atmosphere simulation code Bifrost. Code description and validation. *Astronomy & Astrophysics*, *531*, A154. <https://doi.org/10.1051/0004-6361/201116520>
- Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, *5*(1). <https://doi.org/10.5334/jors.148>
- Koontaweeponya, R., Dimant, Y. S., & Oppenheim, M. M. (2024). Non-Maxwellian ion distribution in the equatorial and auroral electrojets. *Frontiers in Astronomy and Space Sciences*, *11*, 1478536. <https://doi.org/10.3389/fspas.2024.1478536>
- Martínez-Sykora, J., Szydlarski, M., Hansteen, V. H., & De Pontieu, B. (2020). On the Velocity Drift between Ions in the Solar Atmosphere. *The Astrophysical Journal*, *900*(2), 101. <https://doi.org/10.3847/1538-4357/ababa3>
- NASA. (2021). *Plasma, Plasma, Everywhere!* <https://www.nasa.gov/podcasts/curious-universe/plasma-plasma-everywhere>
- Oppenheim, M. M., & Dimant, Y. S. (2004). Ion thermal effects on E-region instabilities: 2D kinetic simulations. *Journal of Atmospheric and Solar-Terrestrial Physics*, *66*(17), 1655–1668. <https://doi.org/10.1016/j.jastp.2004.07.007>
- PlasmaPy Community, Murphy, N. A., Everson, E. T., Stańczak-Marikin, D., Heuer, P. V., Kozłowski, P. M., Johnson, E. T., Malhotra, R., Schaffner, D. A., Vincena, S. T., Abler, M., Addison, J., Ahamed, A. F., Alarcon, P. V., Antognetti, B., Arran, C., Bagherianlemraski, H., Beckers, J., Bedmutha, M., ... Zhang, C. (2024). *PlasmaPy* (Version v2024.10.0). Zenodo. <https://doi.org/10.5281/zenodo.14010450>
- The SunPy Community, Barnes, W. T., Bobra, M. G., Christe, S. D., Freij, N., Hayes, L. A., Ireland, J., Mumford, S., Perez-Suarez, D., Ryan, D. F., Shih, A. Y., Chanda, P., Glogowski, K., Hewett, R., Hughitt, V. K., Hill, A., Hiware, K., Inglis, A., Kirk, M. S. F., ... Dang, T. K. (2020). The SunPy Project: Open Source Development and Status of the Version 1.0 Core Package. *The Astrophysical Journal*, *890*, 68–68. <https://doi.org/10.3847/1538-4357/ab4f7a>
- Vögler, A., Shelyag, S., Schüssler, M., Cattaneo, F., Emonet, T., & Linde, T. (2005). Simulations of magneto-convection in the solar photosphere. Equations, methods, and results of the MURaM code. *Astronomy & Astrophysics*, *429*, 335–351. <https://doi.org/10.1051/0004-6361:20041507>