frontiers

Supplementary Material

Making an Executable Paper with the Python in Heliophysics Community to Foster Open Science and Improve Reproducibility

1 Appendix A. Team Member Descriptions

Eric Grimes

Eric Grimes received his M.S. in Physics from Auburn University in 2012. Since 2012, he has been a Programmer/Analyst at the UCLA Institute of Geophysics and Planetary Physics, specializing in scientific software development. He has developed analysis and visualization tools for several NASA Heliophysics missions, including Magnetospheric Multiscale Mission (MMS), Time History of Events and Macroscale Interactions during Substorms (THEMIS), and more, and currently leads development of the pySPEDAS project. He was responjsible for all the pySPEDAS work during this project.

Jonathan Niehof

Jonathan Niehof earned his Ph.D. from Boston University in 2011 with investigations of energetic ions in Earth's magnetospheric cusps. Since then, he has led data processing and analysis for several missions, including the ECT suite on the Van Allen Probes mission, the ISOIS suite on Parker Solar Probe, and the upcoming Helioswarm mission. He is currently at the University of New Hampshire. He was responsible for all the SpacePy work during this project.

Lutz Rastaetter

Lutz Rastaetter received his Ph.D. in Physics from the Ruhr University Bochum, Germany, in 1997 on numerically modeling the magnetic environment of a rotating star loaded by an accretion disk. He joined NASA GSFC in Greenbelt, Maryland as a post-doctoral researcher working on current sheet thinning and reconnection in the Earth’s magnetosphere. Since the inception of the CCMC in 2000, he has been working on model onboarding, online visualization, model-data validation, and the development of postprocessing tools for use by the science community. He leads the Kamodo model access software development at the CCMC and contributed his science expertise to this work, coordinated the production of the OpenGGCM model data used, and assisted Rebecca in incorporating the OpenGGCM model data into Kamodo's flythrough capability.

Nicholas Murphy

Nick Murphy is an astrophysicist in the Solar & Stellar X-Ray Group at the Smithsonian Astrophysical Observatory (SAO) and a Lecturer in Harvard University's Department of Astronomy. He is also a developer of PlasmaPy, a community-developed core Python package for plasma physics. He was responsible for all the PlasmaPy work during this project.

Rebecca Ringuette

Rebecca Ringuette graduated in 2014 with her doctoral degree in physics from Louisiana State University on detecting terrestrial gamma-ray flashes from the rooftops of LSU. After teaching physics and astronomy at the high school and university levels for a few years, Rebecca transitioned into X-ray astrophysics in 2018, studying astrophysical observations of solar wind charge exchange and developing pipeline software in Python for the HaloSat CubeSat mission. She is now contracted as a scientific software developer through ADNET Systems, Inc, for the Community Coordinated Modeling Center at NASA GSFC, building Kamodo-based tools for easier interaction with space weather model data. Rebecca is also contributing to the improvement of the Heliophysics infrastructure to support open science development through publications and workshops. She orchestrated significant Kamodo development for this project, contributed the model flythrough portion of the presented workflow, originated the idea for this work, and guided the development of the written portion.

Shawn Polson

Shawn Polson graduated from the University of Colorado Boulder in 2020 with his Masters in Computer Science. His graduate school research focused on detecting anomalies in spacecraft with machine learning techniques. He is the Technical Lead of PyHC and he works as a software engineer at the Laboratory for Atmospheric and Space Physics (LASP). He wrote and revised this paper's text, provided technical leadership for this project, and helped enforce good Python programming practices.

Yihua Zheng

Yihua Zheng received her Ph.D. in 2001 from University of New Hampshire, on the field-aligned currents in the auroral zone using sounding rocket measurements. She is a space science and space weather scientist and also a forecaster who has been involved in providing space weather services to NASA robotic missions for over 10 years. She studies the myriad ways that space weather can manifest in near-Earth space. Yihua is the lead at the CCMC for inner magnetosphere models, and for coupling space environment models with engineering impacts. She provided science expertise for this project, and will further build upon this work in a future paper addressing the application of the workflow to a collection of science questions.

2 Appendix B. Deepnote Alternatives

There are well-established alternatives to Deepnote that may be more appropriate for other kinds of work. Leading examples are Binder (Ragan-Kelley and Willing 2018), Google Colaboratory (Google Research 2022b), and JupyterLab (JupyterLab 2022). We briefly describe these platforms here and link to their own documentation for deeper dives.

Binder

Binder turns a Git repository into a collection of interactive notebooks in an executable environment, allowing users to run them in their browser. Users point Binder to their repository, then Binder builds

a Docker image with dependencies that are specified in a dependency file (e.g., “requirements.txt”). A JupyterHub server hosts the repository's contents (Binder 2022).

Pros:

- Easy to set up
- Plays well with GitHub
- Straight-to-the-point

Cons:

- Notebooks must already be stored in a Git repository
- No notions of collaboration (real-time or otherwise)
- Minimal features
- Subject to the file size limits of the Git repository

Google Colaboratory

Google Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows users to write and execute arbitrary Python code through the browser, and is especially well suited to machine learning and data analysis. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access to computing resources including GPUs (Google Research 2022a).

Pros:

- Backed by Google
- Notebooks are stored and shared with Google Drive
- Has most of the same features as Deepnote
- Dark mode

Cons:

- Real-time collaboration was disabled in 2017
- A paid “pro” subscription is required for finer-grain control over the execution environment (e.g., terminal access)
- Only supports Python code
- Notebooks are stored and shared with Google Drive (this can be a pro or a con)

JupyterLab

JupyterLab is Jupyter's latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange countless workflows. A modular design invites extensions to expand and enrich functionality (JupyterLab 2022).

Pros:

- Jupyter is *the* notebook company
- Faster than Jupyter Notebook (especially when restarting a kernel)

- Self-managed servers (e.g., in Amazon Web Services) can allow complete environment control, including file storage space
- Dark mode

Cons:

- Installed and set up locally
- Not a notebook sharing platform
- Third-party platforms (e.g., Binder) are required to share notebooks

3 Appendix C. PyHC Package Descriptions

Kamodo

`kamodo-core` Version 21.10.0 / `kamodo-ccmc` Version 0.0.6

Kamodo is a CCMC tool for access, interpolation, and visualization of space weather models and data in Python (Kamodo 2022a). Its development is led by the CCMC in collaboration with Ensemble Consultancy, LLC. Kamodo allows model developers to represent simulation results as mathematical functions which may be manipulated directly by end users. It handles unit conversions transparently and supports interactive science discovery through Jupyter notebooks with minimal coding. We use the virtual flythrough capabilities from `kamodo-ccmc` to compare observations with the OpenGGCM model and the multiple-variable visualization available through `kamodo-core`. The virtual flythrough depends on SpacePy and Astropy for all coordinate conversions and Plotly for the specialized visualizations. The flythrough is available in the version of Kamodo with CCMC-added capabilities (https://github.com/nasa/Kamodo). This repository depends on the core capabilities available from separate repository (Kamodo 2022b), such as data functionalization, unit conversions, LaTeX representations, nearly automatic visualizations, and more.

PlasmaPy

Version 0.7.0

PlasmaPy's goal is to foster the creation of an open-source Python ecosystem for plasma research and education (PlasmaPy 2022b). The PlasmaPy package contains core functionality for this software ecosystem, while affiliated packages will contain more specialized functionality. PlasmaPy is a community-developed and community-driven free and open-source Python package that provides common functionality required for plasma physics in a single, reliable codebase. We use it to calculate plasma parameters that augment our detections of magnetopause crossings.

pySPEDAS

Version 1.3.3

pySPEDAS is an implementation of the Space Physics Environment Data Analysis Software (SPEDAS) framework for Python (pySPEDAS 2022a). The SPEDAS framework is written in IDL and contains data loading, data analysis and data plotting tools for various scientific missions

(NASA, NOAA, etc.) and ground magnetometers. SPEDAS is a grass-roots data analysis software for the Space Physics community. It supports multi-mission data ingestion, analysis and visualization. It standardizes the retrieval of data from distributed repositories, the scientific processing with a powerful set of legacy routines, the quick visualization with full output control and the graph creation for use in papers and presentations. We use it to get MMS data.

PyTplot

`pytplot-mpl-temp` Version 2.0.1

PyTplot is a python package which aims to mimic the functionality of the IDL “tplot” libraries (PyTplot 2022a). These plots have several user interaction tools built in, such as zooming and panning. They can be exported as standalone HTML files (to retain their interactivity) or as static PNG files. A recent rewrite of PyTplot's backend made it compatible with the popular Python library matplotlib, which we use because of its enhanced compatibility with Jupyter notebooks.

SpacePy

Version 0.2.3

SpacePy is a package targeted at the space sciences that aims to make basic data analysis, modeling and visualization easier (Niehof et al. 2022; Morley et al. 2011). It builds on the capabilities of the well-known NumPy and matplotlib packages. Publication quality output direct from analyses is emphasized among other goals. We use it to access the Shue et al. model, to do coordinate conversions, and to derive quantities like the magnetopause's distance from Earth and the spacecraft's distance from the magnetopause.