Heliophysics Data Environment Enhancements
Abstracts of Selected Proposals
(NNH19ZDA001N-HDEE)

Below are the abstracts of proposals selected for funding for the Heliophysics Data Environment Enhancements program. Principal Investigator (PI) name, institution, and proposal title are also included. 15 proposals were received in response to this opportunity. On October 19, 2019, 11 proposals were selected for funding.

Ricky Egeland/University Corporation For Atmospheric Research (UCAR)
SunPy support for multidimensional data and spectropolarimetric analysis

Solar spectropolarimetry provides the most detailed description of the solar atmosphere, in particular its dynamic and complex magnetic field. Both calibrated Stokes spectra, as well as atmospheric parameters from inversions are inherently multidimensional. Observational data include Stokes I, Q, U, & V for each surface pixel location (X, Y) at a given wavelength, a total of 7 dimensions. Inversion results consist of (e.g.) plasma temperature, line-of-sight velocity, and full vector magnetic field for each pixel location, in some cases as a function of optical depth. Such multidimensional datasets are presently beyond the capabilities of SunPy, which largely is geared towards handling Maps, 2D representations of image-like data. To support solar spectropolarimetry in Python, we propose to develop StokesPy, a new SunPy affiliate package to manipulate and visualize Stokes profiles and compare them to inversion results. We will develop this code using publicly-available Stokes parameters from SDO/HMI and Hinode SOT/SP and inversion results at the Joint Science Operations Center (JSOC) and Community Spectro-polarimetric Analysis Center (CSAC) initiative, respectively. We will aim for the package to be generally applicable to a wide range of spectropolarimeters and inversion codes. We will solicit feedback from the wider solar spectropolarimetry community and promote the package with the aim of improving Python and SunPy use within this realm of solar physics. This addition to the SunPy ecosystem will be particularly timely as new spectropolarimetric data products become available from DKIST in late 2019/early 2020, and others to follow throughout the decade. StokesPy has the potential to become a locus point of Python development in the spectropolarimetry community, seeding new developments and functionality for the broad benefit of heliophysics research.

Andrew Inglis/Catholic University Of America
The AFINO code as a complete, interdisciplinary Python analysis package

Waves, oscillations and pulsations are ubiquitous phenomena in many scientific disciplines. This is particularly true in the Heliophysics domain, where wave-like plasma processes occur in all layers of the solar atmosphere, from standing and travelling magnetoacoustic waves in plasma loops, to global EUV waves (or EIT waves), to pulsations and oscillations triggered by solar flare energy release. The commonality of all

these processes is that identifying them and gaining physical insight requires time-domain analysis of the observational data.

The AFINO (Automated Flare Inference of Oscillations) code was developed in the Python programming language in order to provide a robust, automatable means of analyzing solar flare data in search of oscillatory signals, while accounting for the background power-law power spectrum properties of flares (e.g. Vaughan 2010, Inglis et al. 2015, 2016). We emphasize however that AFINO is a generic code that can operate on any time series data; the analysis method has many uses in the wider fields of Astrophysics and Earth Science.
To date AFINO has been successfully used to carry out a large-scale of solar flare time series (Inglis et al. 2016). It has since been adapted to search for waves in solar wind data (Murphy et al. 2018). AFINO was also included in a comprehensive blind-test study carried out by Broomhall et al. 2019, who compared a variety of popular time series analysis methods. AFINO was found to perform well in these blind tests, and featured the lowest false alarm rate amongst all tested methods. In total, AFINO has been used as a primary analysis method in 6 separate scientific papers (Inglis et al. 2015, 2016, Murphy et al. 2018, 2019, Hayes et al. 2019, Broomhall et al. 2019).

Despite this success, two major factors have limited the use of AFINO to date. First, AFINO was written in Python 2.7 and depends on SunPy 0.6. However, in the time since AFINO s original development SunPy have decided drop Python 2.7 support, and have adopted Python 3.6 and higher, a trend reflected in the wider astrophysics community. In addition, the SunPy package has both added new features and refactored existing ones used by AFINO. As such, the AFINO codebase in its current form is out-of-date. Secondly, due to limited development resources AFINO does not exist as a clean, well-documented publicly available package, but only as an in-house code. This makes distribution and adoption of the code on a wider scale difficult.

We propose a modest 1-year development effort to solve both of these issues and release AFINO as a clean, fully documented analysis package to the Heliophysics community and beyond. AFINO will be updated to run on Python>3.6 systems, and dependencies updated to reflect the latest changes in the SunPy and AstroPy analysis packages. Complete documentation, including appropriate docstrings and examples will be provided. The codebase itself will be hosted on an open-access site such as Github. The end result will be a powerful tool for time series analysis that can be accessed by a wide scientific community across disciplines, without the need for intensive interaction with the original developers.

Jack Ireland/NASA Goddard Space Flight Center
Supporting and extending SunPy for the heliophysics community

The SunPy project provides basic infrastructure enabling solar and heliospheric data analysis in the scientific Python environment. This proposal will support existing SunPy functionality and extend it to provide a foundation for the Python in Heliophysics Community (PyHC).

We will create a new spectral data object in SunPy for the storage and manipulation of spectral data from multiple instrument sources. The new spectral data object will provide the same methods and properties for all spectra regardless of source, saving time and effort for the user, and will provide fundamental information such as observation time and location as object properties. This is important for multi-viewpoint studies, and provides support for space-based spectrometers across multiple heliophysical disciplines, such as Spectrometer Telescope for Imaging X-rays (STIX) onboard Solar Orbiter.

SunPy's existing coordinate infrastructure will be extended to include more celestial coordinate frames, in particular those described in Franz & Harper (2002). This will support understanding where observations were taken in space; including these coordinate systems will extend SunPy's and Astropy's coordinate transformation graph, enabling easy coordinate transformation between all available coordinate frames. Concomitant with this, we will create functionality to bring orbit information in to SunPy using capabilities already existing in heliopy, spiceypy and poliastro. This will enable science by providing easy access to the position and velocity of spacecraft, which is of vital importance in understanding in-situ observations and physics. For example, understanding the position and velocity of the Parker Solar Probe (PSP) is important in understanding and comparing observed and modeled in-situ solar wind conditions.

We will make it easier to use SunPy and PyHC functionality by copy-editing the existing documentation and updating/adding new examples to the SunPy gallery. Gallery examples will provide code snippets demonstrating common data analysis use cases implemented via SunPy and PyHC functionality. This will enable science by allowing users to minimize time wasted in re-inventing functionality that is already provided by SunPy and the PyHC. For example, re-projecting AIA image to other points of view (PSP, STEREO, Solar Orbiter) is important for understanding the solar surface as seen from those missions. To do this in Python requires knowledge of SunPy maps, AstroPy coordinates and spiceypy functionality; a simple example will demonstrate package functionalities, their interaction and explain the steps involved.

Finally, we will maintain the health and code quality of SunPy through the maintaining the continuous integration facility currently used by SunPy, and by conducting regular and frequent reviews of the codebase and test coverage. This will ensure code quality for all SunPy-related science applications and users, and strengthen SunPy's support for NASA's Heliophysics System Observatory into the future. Implementation of the work of this proposal will follow the PyHC Community Standards.

Nicholas Murphy/Smithsonian Institution/Smithsonian Astrophysical Observatory
The next development release of PlasmaPy

Plasma physics is of fundamental importance to heliophysics. The Python ecosystem for heliophysics therefore requires plasma physics functionality in order to support current and planned NASA missions to investigate the heliosphere. PlasmaPy is an open source Python package for plasma physics in the early stages of development. PlasmaPy strives

to provide the functionality needed to study heliospheric, laboratory, and astrophysical plasmas. The goals of this proposal are to add capabilities to PlasmaPy, improve intercompatibility between PlasmaPy and other packages in the emerging Python ecosystem for heliophysics, and officially release the next version of PlasmaPy. The first planned task is to create a standard to represent the initial conditions, boundary conditions, and computational domain for numerical simulations of plasmas. A Python implementation of this standard will be implemented into PlasmaPy. This task will provide the heliophysics software ecosystem with tools to enable researchers to straightforwardly switch between different codes with different physical models in order to perform benchmarks and verification studies. The second planned task will be to complete the implementation of non-equilibrium ionization (NEI) modeling capabilities into the heliophysics software ecosystem. NEI modeling capabilities will enable researchers to investigate the evolution of charge states of plasma in the solar wind and during solar eruptions, and thus be able to predict and interpret both remote and in situ observations by NASA missions. Throughout this project, we will coordinate with the Python in Heliophysics Community and make improvements to existing code and documentation within PlasmaPy for long-term maintainability.

Dusan Odstrcil/George Mason University
Visualization of the ENLIL heliospheric model results with Python

The numerical heliospheric code ENLIL is the most frequently used code by the space weather community at the NASA Community Coordinated Modeling Center (CCMC). This code is also used for operational predictions at NOAA Space Weather Prediction Center (SWPC) and at NASA Space Weather Research Center (SWRC) to support NASA missions. Currently, all ENLIL visualizations are enabled via tailored IDL procedures. There are over 30 different visualization outputs to display boundary conditions, temporal profiles at planetary and spacecraft positions, values of magnetohydrodynamic parameters at ecliptic and meridional and radial slices, magnetic field vectors, topology of the magnetic field lines passing through the observer, shock alert plots, synthetic white-light images, etc. To enable high-quality fonts, firstly postscript files are generated by IDL procedures and then ImageMagick software is used to convert these files into images. This process is very a slow, producing of images takes often more time than numerical computation, and, only a subset of available IDL procedures is thus used at CCMC. We propose to explore using of Python to produce high-quality imaging as is currently produced by IDL. We anticipate significant acceleration of the visualization process and this would make more outputs available to users. Further, since the ENLIL is currently the only code that can simulate coronal mass ejections all around the Sun up to the Jupiter or Saturn orbit, our procedures would be useful for other, and more sophisticated, heliospheric codes in future. Finally, comparison between numerical simulations, in-situ measurements, and remote observations is a complex task and therefore a coordinated effort with the Python heliospheric community is very necessary.This project will be realized in cooperation and coordination with the CCMC, Python heliospheric community and other selected projects.

Kevin Reardon/Association Of Universities For Research In Astronomy, Inc.

An open-source toolkit for merging multi-observatory solar datasets

Solar physics relies heavily on probing the Sun using diagnostics available across the electromagnetic spectrum in order to sample multiple regions of the solar atmosphere. The formation mechanisms at different wavelengths offer insights into a variety of underlying physical processes. Spacecraft above the Earth's atmosphere are the only way to access the diagnostics in the UV and X-ray, or they can leverage the advantage of continuous, seeing-free coverage. Ground-based observatories instead can probe diagnostics in the visible, infrared, and radio with great detail and flexibility.

Precisely combining data from these different observatories has proved to be highly important to increasing our understanding of the structuring and dynamics of the solar atmosphere. Both space- and ground-based instruments continue to increase their spatial resolution to better resolve physical processes at their fundamental scales. This, however, increases the requirements and complexity of precisely remapping these different images onto a common spatio-temporal reference. Some of these issues are found across all instruments, such as optical distortions or time-varying jitter and roll. Other problems might be more specific to ground-based data including atmospheric refraction, turbulent image distortion, and variable transmission. Generally, the coordinate definitions given in the data headers are not sufficiently accurate for many scientific needs. Therefore, it is necessary to use the data themselves, or other calibrations, to achieve the correct mapping and generate a properly aligned dataset suitable for proper scientific exploitation. This currently requires expert knowledge, manual calibration, and the use of limited-availability codes written in a proprietary language.

We propose to produce an open-source, properly engineered toolkit in Python that can address some of these issues and make it more feasible for the broader community to seamlessly combine datasets from satellites and ground-based instrumentation. This would empower broader use of data from multiple sources in a robust fashion.
This development would use rely heavily on existing projects, such as SunPy, AstroPy, and others, to handle some of the key underlying tasks of data retrieval, coordinate definition, data-cube manipulation. This toolkit will provide tools to perform specific tasks such as non-rigid alignment between images, correction for plate-scale distortions, and compensation for atmospheric variations. We plan to work with the the Python in Heliophysics Community and ensure that the toolkit will be compliant with their agreed upon software standards.

This toolkit will also be leveraged in the ongoing development of an open-source visualization tool that will allow spacecraft and ground-based data to be easily viewed together, probing also the temporal and spectral dimensions of the datacubes generated by different instruments. This will be applied especially to DKIST observations, allowing users to jointly explore those complex data together with data from a variety of NASA assets (SDO, IRIS, Hinode, PSP, and others). This visualization tool will make the data more accessible while the precise co-alignment of the data from various sources will allow scientifically important features to be identified more efficiently. With accurate co-

alignment and user-friendly data manipulation, this will be a key way for users to identify, combine, and explore data from multiple observatories.

Daniel Ryan/Catholic University Of America

Supporting Heliophysics Data Analysis and Software Development with the Development of NDCube, a Generalized N-dimensional Coordinate-aware Data Class

This work enhances the ndcube package to meet the needs of the heliophysics community.
ndcube is an open-source, community-developed Python package developed within the Python in Heliophysics Community framework.
It provides unit- and coordinate-aware data classes for generalized multi-dimensional astronomical and heliospheric data analysis.
These classes inherit from astropy NDData and combine data, uncertainties, data quality flags, real world coordinates coordinates, units and other general metadata into single objects.
They provide comprehensive slicing, visualization, and coordinate transformation functionalities beyond what's provided by astropy NDData that facilitate fast and accurate manipulation and analysis of heliospheric datasets.
This contrasts with the traditional data analysis paradigm where each component (data, coordinates, etc.) were stored in separate arrays or objects.
Manipulating them was tedious and error-prone.
ndcube therefore helps to increase the quality and quantity of users' scientific output.

ndcube is distinct from packages like xarray in that is supports functional coordinate transformations.
This makes it ideal to meeting the needs of fields within heliophysics, like solar physics which uses functional coordinate systems extensively.
However, further work is need to maximize ndcube's potential to the heliophysics community.

We will add new features to ndcube relevant to analyzing NASA heliospheric observations.
These include: arithmetic operations; resampling; support for generalized World Coordinate System (gWCS); removing ndcube's dependency on sunpy; and save and read functionalities.
These features will be developed by leveraging currently available functionalities in stable, open-source, scientific Python packages including numpy, scipy, astropy, etc.
ndcube does not only support heliophysicists, but also other heliophysics Python packages.
It is a dependency of IRISpy, an open-source package for analyzing UV spectroscopic imaging observations of the solar chromosphere from NASA's IRIS satellite.
ndcube is also planned to become a dependency of other heliophysics Python packages including sunpy and PySat.

The proposed new features will therefore facilitate planned future development of these packages as well as providing more powerful analysis tools directly to users.
This will directly and indirectly help heliophysicists pursue NASA's heliophysics strategic goals and objectives including: understanding the Sun, Earth, Solar System and Universe; exploring the physical processes in the space environment from the Sun to the Earth and throughout the solar system, and; advancing our understanding of the connections that link the Sun, the Earth, planetary space environments.

This work will be developed, tested and documented in accordance with PyHC standards and made publicly available on GitHub and in ndcube s next major release, 2.0.

Joshua Semeter/Boston University
PyGemini: a community 3D local scale ionosphere dynamics model

GEMINI: the Geospace Environment Model for Ion-Neutral Interactions: https://github.com/gemini3d provides comprehensive modeling capabilities for the ionosphere-thermosphere (IT) system across spatial scales ranging from ~200 m to thousands of km at sub-second temporal resolution. GEMINI s local scale physics models fill a crucial modeling scale gap, connecting kinetic scale models to global and assimilative models.

GEMINI has been applied to a wide variety of ionospheric physics problems including studies of:
* IT responses following earthquakes: https://doi.org/10.1029/2018GL081569
* fluid turbulence (Deshpande and Zettergren, 2019, in press)
* tropospheric weather-related IT responses: https://ui.adsabs.harvard.edu/abs/2016AGUFMSA32A..06S
* plasma density cavity formation: https://doi.org/10.1029/2012JA017637
* auroral arc electrodynamics: https://doi.org/10.1002/2014JA020860

GEMINI s modular design includes existing interfaces to models of:
* neutral dynamics (MAGIC, op. cit.)
* kinetic electron transport (GLOW): https://github.com/NCAR/GLOW
* ionospheric radio propagation (SIGMA, op. cit.)

GEMINI is open source software with auto-generated HTML documentation and a community wiki. GEMINI s parallel numerical core is written in Fortran 2008 (OpenMPI). GEMINI model runs are scalable from laptops to thousands of CPU cores on HPC systems. Model setup, plotting and analysis of simulation results is presently done via Matlab scripts. The proposed PyGEMINI upgrades will bring the richness of physics that GEMINI encapsulates to the Python Heliophysics (HP) community via improved HP standard interfaces and data formats.

Upgrading to PyGEMINI addresses NASA HP Decadal Survey priorities by modeling IT coupling across scales, particularly local and small-scale phenomena that bear a significant share of the IT energy flux budget. GEMINI is one of the few local-scale, general purpose IT models and is adaptable to a wide variety of observational missions and coupled model use cases. GEMINI has a continuous-integration test framework on Travis-CI https://travis-ci.com/gemini3d/GEMINI and is designed to make plugging-in to external program modules as easy as possible. Numerous missions are already using GEMINI, including ISINGLASS, VISIONS2 and SEED.

The proposed PyGEMINI open source development work will:
1. replace Matlab model setup and analysis scripts with HP standard Python modules that interoperate with external HP Python modules, ingesting object tracks and model parameters; with model output in HP standard API and data formats, along with producing web and publication-ready plots

2. complete build system modernization so all prerequisite libraries compile with Python 3 and a C / Fortran compiler on any computing environment (viz. MacOS, Linux, Windows).
3. develop a new Python user interface for setting up model runs for analysis of spacecraft, rocket and other observational data.
4. provide model outputs and accept model inputs in community standard formats and protocols including HAPI and NetCDF / HDF5.
Community accessibility is enhanced by removing reliance on Matlab, and by developing user-friendly setup and execution interfaces that interoperate with data from packages such as PySat.

One of the biggest hurdles for adoption of modeling software is successfully performing the initial build and test. We are transitioning the GEMINI build system to CMake, which will allow building PyGEMINI on any computing environment suitable for IT system modeling. Python and HP community standards will be adhered to, including PEP8 style and MyPy type hint checks. Funding from the HDEE program will allow us to greatly accelerate the growth of GEMINI to PyGEMINI, smoothing the pathway for the HP community to model IT physics vis-a-vis observed data across IT system scales. The PyGEMINI Fortran-Python integration architecture will be an example for other IT models to better connect with the Python HP community.

Jonathon Smith/Catholic University Of America
Scintillation and High-resolution Low-level Ionospheric Data Instrument for Pysat

We propose to produce a python tool utilizing low-level data products from distributed global positioning system (GPS) networks. The product of this tool will be the user's choice of ionospheric indices such as S4, Rate of total electron content (TEC) Index (ROTI), or the disturbance ionosphere index (DIX). As part of an effort to meet community standards for the quality of the code and its availability, this package will not be standalone but developed for integration into the larger ionospheric research package, Pysat.

The currently available data products from distributed GPS receivers are simultaneously extensive and limited. Although parameters like TEC and ROTI are available globally from many networks of GPS the low-level intensity information and the indices that can be derived from this data such as S4 are not widely available outside of the COSMIC data analysis and archive center (CDAAC) system for utilization of signals from the Constellation Observing System for Meteorology, Ionosphere, and Climate (COSMIC). However, it is these ground-based receivers are not only communicating with one satellite. Given the number of flying GPS satellites, it stands to reason that this leaves an immense reservoir of untapped data.

This kind of data effort has no doubt been undertaken before at the very least with individual receivers or local networks of several receivers for focused science topics. Providing access to a global network of low-level data products from GPS receivers would not only allow for wider use of existing indices for global monitoring of GPS

scintillation but also lower the barrier of entry for the analysis of more rigorous measures of ionospheric irregularity based on GPS signal.

The work plan for this undertaking begins with a small scale application of the techniques to be expanded to available GPS networks. Starting with the Canadian High Arctic Ionospheric Network (CHAIN) a new instrument type will be added to the Pysat package that will be capable of downloading and parsing the data from the receivers. The functionality of this instrument will be extended to a broader range of GPS networks and individual receivers to include all publicly available data sources for GPS data. Ready-made functions for calculating the S4, ROTI, and DIX indices will be included in this instrument. Additionally, the Pysat style functionality customization approach will be implemented by nature of being included in the larger package. This will allow for this project to produce the intended science as well as provide a straightforward method for the community to manipulate the data however they see fit.

The tools are only as good as the science they facilitate, and the experiments made immediately feasible by this work include validation of TEC modeling through WACCM and SAMI3 as well as validation of DIX index prediction for ionospheric disturbances. Specifically comparison between plasma measurements and scintillation to explore the relationship between irregularity growth rate and scintillation. It has been established and widely accepted that the small scale irregular plasma in the bottomside ionosphere and inside of larger plumes is responsible for the type of wave modulation that devastates GPS communications. The irregularity growth rate serves as an indication of the likelihood an irregularity will grow into a larger plume of depleted plasma in the presence of perturbations in the bottomside ionosphere. By simultaneously calculating the local growth rate, observing scintillation, and in situ plasma density a cross-examination of the type of irregularities producing scintillation can be undertaken.

Jon Vandegriff/Johns Hopkins University
Enhancing and Standardizing Python Readers for HAPI Data

The Heliophysics Application Programmer s Interface (HAPI) is an emerging standard that data providers can use to make data available through a computer-to-computer service. HAPI offers a common streaming format for time series data products such as the in-situ measurements that are typical on NASA s Heliophysics missions. Python readers for HAPI compliant servers have been written and are available on GitHub, but we propose here to 1) bolster these readers to add features and make them more user-friendly, 2) develop a larger reader framework in the emerging HelioPython ecosystem of which HAPI readers are one part 3) update the HAPI reader documentation and create a user guide and 3) develop one sample application using the HAPI readers.

One key reason to include HAPI data access in the emerging low-level Python library is to allow people to start creating other functionality on top of HAPI that enables a different and interoperable way of thinking about data. If access to any data set in any data center is now trivial, other kinds of functionality now become possible, such as data fusing libraries (combining data from different sources, with options for averaging or

interpolating datasets to match time cadences), coincident measurement locators (finding times when two instruments measured similar regions), and generic data mining or event finding routines (find all magnetic depolarization events from any magnetometer data set at any data center).

Therefore, in addition to just making the HAPI Python library available, we will also include an example of one higher level service that uses HAPI. We will create a data fusion application that can merge different time series data sets from any HAPI server. This example will serve as a template for others who want to create additional HAPI-based services in the future, and it will ensure that these services follow a coherent approach that is consistent with the principles and growth strategy of the HelioPython library as a whole.

Bifford Williams/G & A Technical Software, Inc.
SkyWinder: a PyHC package to investigate dynamics, clouds, and turbulence in the middle and upper atmosphere

We propose a Value Added Enhancement Python package in support of NASA Heliophysics science goals including:
1. Dynamics and coupling between the mesosphere and thermosphere on small (<10m) to suborbital scales
2. Physical processes of the sun-earth system such as polar mesospheric clouds, turbulence and neutral atmosphere waves and instabilities.

We propose to create a python software package broadly applicable to suborbital mobile aeronomy experiments such as balloon-borne payloads, airplanes, sounding rockets, and suborbital reusable launch vehicles (sRLVs) funded by NASA Heliophysics H-FORT/Low Cost Access to Space (LCAS). We will standardize, package, document, and expand on the successful NASA PMC-Turbo balloon mission software which used python for flight control, telemetry, analysis of imager and lidar data, and data visualization. The package developed in this effort will have broad uses for mid- and upper-atmosphere science and instrumentation including mesospheric airglow imagers, optical cameras, and other lidars, especially those hosted on moving platforms. The current PyHC codebase includes more than 50 projects, but there is a need for a package dealing with mobile, suborbital missions and dynamics, coupling, and clouds in the mesosphere and lower thermosphere.

PMC-Turbo was a balloon-borne experiment designed to observe gravity waves, instabilities, and turbulence using Polar Mesospheric Clouds (PMCs) as tracers. PMCs form a thin layer at ~83 km altitude above the summer polar regions that act as sensitive tracers for small to medium scale dynamics in a very active region of the atmosphere. The PMC-Turbo balloon-borne platform was built to record these turbulent events using seven high-resolution imagers to measure PMC brightness variations and a Rayleigh lidar which measured the PMC altitudes. PMC-Turbo flew at 38 km altitude for nearly 6 days from Esrange Space Center in Sweden to northern Canada. In flight, our modular Python flight software ran on a distributed array of 7 CPU s to receive and execute commands

sent from the ground and downlink compressed images and housekeeping data over three independent telecom channels in a resilient manner.

Many mobile platforms (balloon, aircraft, sRLV) require star cameras to keep track of their pointing at float. This technology has not been standardized, but the software written for PMC-Turbo provides a robust, tested, and resilient base for other groups needing pointing and flight control. Most of these platforms communicate via TDRSS and/or Iridium and the PMC-Turbo telemetry software already gracefully handles communication over a variety of channels.

After the successful flight, the PMC-Turbo team developed python analysis software to flat-field the captured images, find the pointing of the images using the star field, and combining and projecting the images to accurate spatial scales on the PMC layer. The team also developed data visualization tools to use the combined images and lidar profiles to observe dynamics at a variety of resolutions and time scales.

We propose to upgrade, clean, standardize, document, and package the PMC-Turbo python code into a GitHub public package, accessible to other aeronomy groups. The PI (B. Williams) has participated in many past LCAS missions and will provide scientific supervision. The postdoc (C. Kjellstrand) wrote much of the existing python code and will upgrade the code. This package will be part of the developing PyHC project and will conform to the standards of stability and documentation that the larger PyHC effort is currently developing. Our team has already participated in PyHC telecons and we will attend future PyHC meetings in order to coordinate with existing PyHC code for data transformation and visualization.
