# SunPy: Python for Solar Physicists

Stuart Mumford[‡*], David Pérez-Suárez[¶], Steven Christe[‖], Florian Mayer[§], Russell J. Hewett[**]

http://www.youtube.com/watch?v=bXPPTCkaVu8

**Abstract**—SunPy is a data analysis toolkit which provides the necessary software for analyzing solar and heliospheric datasets in Python. SunPy aims to provide a free and open-source alternative to the current standard, an IDL-based solar data analysis environment known as SolarSoft (SSW). We present the latest release of SunPy, version 0.3. Though still in active development, SunPy already provides important functionality for solar data analysis. SunPy provides data structures for representing the most common solar data types: images, lightcurves, and spectra. To enable the acquisition of scientific data, SunPy provides integration with the Virtual Solar Observatory (VSO), a single source for accessing most solar data sets, and integration with the Heliophysics Event Knowledgebase (HEK), a database of transient solar events such as solar flares or coronal mass ejections. SunPy utilizes many packages from the greater scientific Python community, including *NumPy* and *SciPy* for core data types and analysis routines, *PyFITS* for opening image files, in FITS format, from major solar missions (*e.g.*, SDO/AIA, SOHO/EIT, SOHO/LASCO, and STEREO) into WCS-aware map objects, and *pandas* for advanced time-series analysis tools for data from missions such as GOES, SDO/EVE, and Proba2/LYRA, as well as support for radio spectra (*e.g.*, e-Callisto). Future releases will build upon and integrate with current work in the Astropy project and the rest of the scientific python community, to bring greater functionality to SunPy users.

**Index Terms**—Python, Solar Physics, Scientific Python

## Introduction

Modern solar physics, similar to astrophysics, requires increasingly complex software tools both for the retrieval as well as the analysis of data. The Sun is the most well-observed star. As such, solar physics is unique in its ability to access large amounts of high resolution ground- and space-based observations of the Sun at many different wavelengths and spatial scales with high time cadence. Modern solar physics, similar to astrophysics, therefore requires increasingly complex software tools, both for the retrieval and the analysis of data. For example, NASA's Solar Dynamics Observatory (SDO) satellite records over 1 TB of data per day all of which is telemetered to the ground and available for analysis. As a result, scientists have to process large volumes of complex data products. In order to make meaningful advances in solar physics, it is important for the software tools to be standardized, easy to use, and transparent, so that the community can build upon a common foundation.

Currently, most solar physics analysis is performed with a library of routines called SolarSoft [SSW]. SolarSoft is a set of integrated software libraries, databases, and system utilities which provide a common programming and data analysis environment for solar physics. It is primarily an IDL (Interactive Data Language)-based system, although some instrument teams integrate executables written in other languages. While SSW is open-source and freely available, the IDL core is not. In addition, contributing to SolarSoft is not open to the public. One of SunPy's key aims is to provide a free and open source alternative to the SolarSoft library.

The scope of a solar physics library can be divided up into two main parts, data processing and data analysis. Data processing is the process of calibrating and aligning data, while data analysis is the scientific analysis of the processed data. SunPy's current scope is data analysis with minimal data processing.

SunPy currently depends upon the core scientific packages like *NumPy*, *SciPy* and *matplotlib*. As well as *Pandas*, *suds*, *PyFITS* / *astropy.io.fits* and *beautifulsoup4*. The latest release of SunPy is available in PyPI and can be installed in the usual manner.

## SunPy Data Types

At SunPy's core are interoperable data types that cover the wide range of observational data available. These core data types, *Lightcurve*, *Map*, and *Spectra*, cover multi-dimensional data and provide basic manipulation and visualization routines with a consistent API. In this section each of these key data types are described.

While these different data types have clear applications to different types of observations, there are also clear interlinks between them, for example a one pixel slice of a *MapCube* should result in a *Lightcurve* and a one pixel slice of a *Composite Map* should be a *Spectrum*. While these types of interoperability are not yet implemented in SunPy, it is a future goal.

The major change in version 0.3 of SunPy is a refactoring of the core data types. This process involved a change in the inheritance structure for *Map* and *Spectrum* away from inheriting the *numpy.ndarray* object to having a more flexible data attribute. This refactoring has also led to some related changes and the ground work being done to facilitate the integration of Astropy's *NDData* object.

### *Map*

The "Map" data type is designed for interpreting and processing the most common form of solar data, that of a two-dimensional

* *Corresponding author: stuart@mumford.me.uk*
‡ *The University of Sheffield*
¶ *Finnish Meteorological Institute*
‖ *NASA Goddard Space Flight Center*
§ *Vienna University of Technology*
** *Massachusetts Institute of Technology*

image most often taken by a CCD camera. A *Map* object consists of a data array endowed with a coordinate system and combined with meta data. Most often, these data are provided in the form of FITS files but image data can come from other file types, such as JPG2000, as well. The meta data in most solar FITS files conform to a historic standard to describe the image such as observation time, wavelength of the observation, exposure time, etc. In addition, standard header tags specify a coordinate system and provide the information necessary to transform the pixel coordinates to physical coordinates such as sky coordinates. Newer missions such as STEREO or AIA on SDO make use of a more precise standard defined by Thompson [WCS]. Thompson also defined standard coordinate transformations to convert from observer-based coordinates to coordinates on the Sun. Since the Sun is a gaseous body with no fixed points of reference and different parts of the Sun rotate at different rates, this is a particularly tricky problem. Through SunPy's WCS (World Coordinate System) library, which has implemented most of these coordinates systems, SunPy Map objects can transform data between them. SunPy maps also provide other core functionality such as routines to rescale, resample, rotate and visualize data and convenience functions for plotting using matplotlib.

There are many forms of image data that can be stored in a *Map*. SunPy maps can handle 2D image data as well as 3D image data for both wavelength-composite images and other series, such as time series data. All 2D map types have a common parent which has been designed with the possibility of integrating with the Astropy library's *NDData* object.

The other main functionality for SunPy's *Map* type, and other data types, is to provide transparent handling of instrument specific code. This code can take the form of translation of non-standard or specific meta data or more complex calibration routines. These functions are handled primarily by the implementation of "sources" which are subclasses of the 2D map object, which then hold this specific code. This leads to many different objects being in the map "family", this is why an automated factory class *Map* has been developed to provide the user with a transparent interface for the creation of Maps.

It is very simple to create and visualize a map in SunPy 0.3:

```
import sunpy
mymap = sunpy.Map(sunpy.AIA_171_IMAGE)
mymap.peek()
```

the output of this command is shown in Fig. 1.

SunPy's visualization routine are designed to interface as much as possible with matplotlib's pyplot package. It is therefore possible to create more complex plots using custom matplotlib commands.

```
import matplotlib.pyplot as plt
import sunpy

mymap = sunpy.Map(sunpy.AIA_171_IMAGE)

fig = plt.figure()
im = mymap.plot()
plt.title("The Sun!")
plt.colorbar()
plt.show()
```

This would produce the same image as Fig. 1 but with a custom title.

### *LightCurve*

Time series data are an important element in solar physics and many data sources are available. In recognition of this fact, SunPy

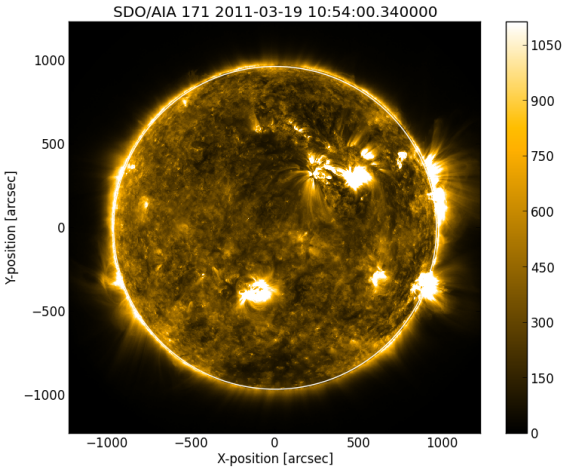


***Fig. 1:*** *Default visualization of a AIAMap.*

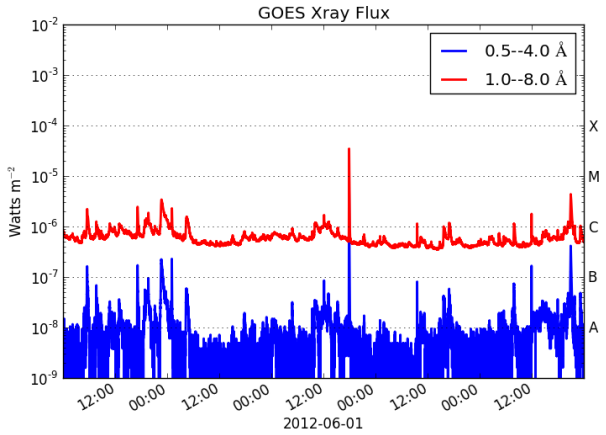


***Fig. 2:*** *Default visualization of a GOESLightCurve.*

provides a *Lightcurve* object which recognizes a number of data sources. The main engine behind the *Lightcurve* object is the *pandas* data analysis library. Each *Lightcurve* holds its data inside a *pandas* object. The *Lightcurve* object, as all other SunPy objects, is wrapper around a data object. Since *pandas* already provides many capabilities, the SunPy *Lightcurve* object does not need to. The *Lightcurve* object recognizes the following data sources; GOES X-ray Sensor (XRS) , SDO EUV Variability Experiment (EVE), and PROBA2/LYRA. Since time series data is generally relatively small and there is no established standard as to how it should be stored and distributed, each SunPy *Lightcurve* object provides the ability to download it's own data in its constructor. The example below retrieves the data, creates a *Lightcurve* object and plots the data in the default manner (show in 2):

```
import sunpy
goes = sunpy.lightcurve.GOESLightCurve.create(
            '2012/06/01', '2012/06/05')
goes.peek()
```

### *Spectra*

SunPy offers a *Spectrogram* object, currently with a specialization for e-Callisto (an international network of solar radio

spectrometers) spectrograms. It allows the user to seamlessly join different observations; download data through an interface that only requires location and time-range to be specified; linearize the frequency axis and automatically downsample large observations to allow them to be rendered on a normal computer screen and much more to help analyze spectrograms. The data can currently be read from Callisto FITS files, but the system is designed in a way that makes it easy to include new data-sources with potentially different data formats (such as LOFAR).

*Spectra* is designed to have a consistent interface along with the other data types. This means the plotting and manipulation methods, where there is shared functionality share the same names and the general structure of the objects are standardized.

### Solar Data Retrieval and Access

Most solar observations provided by NASA or ESA follow an open data policy[1] which means that all data is available publicly, as soon the data is telemetered to the ground. However, these data are normally archived by the institution in charge of the instrument that made the observations. This fact makes browsing data and data retrieval a difficult and tedious task for the scientist. In recognition of this fact, the Virtual Solar Observatory (VSO) [VSO] was developed. The VSO strives to provides a one-stop shop for solar data, by building a centralized database with access to multiple archives. The VSO allows the user to search using parameters such as instrument name or type, time, physical observable and/or spectral range. VSO's main interface is web-based, but an API based on a WSDL webservice is also available. SunPy provides a Python front-end to this API.

A new problem arose with the launch of the SDO mission. The large size of the images (4 times larger than the previous missions), together with the fast cadence of their cameras (~10 images per minute) makes it challenging to use of the data in the same manner as from previous observations. Previously the standard workflow was to download long time series of data and to view animations to identify features of interest to the scientist. For SDO this would involve downloading prohibitively large amounts of data. The Heliophysics Event Knowledgebase [HEK] was created to solve this overload of data. The principle behind the HEK is to run a number of automated detection algorithms on the data that is obtained by SDO to populate a database with information about the features and events observed in each image. Thus allowing searches for event types or properties, enabling scientists to selectively download only the portion or slices of the images needed for further analysis. SunPy provides a programmatic way to search and retrieve the information related to the events, but currently does not have facilities for downloading the observational data. This allows, for example, over plotting of the feature contours on an image, to study their properties and evolution, etc. The HEK interface in SunPy was developed in concert with SunPy's VSO tool, so they have a consistent interface.

Events on the Sun also affect the rest of the solar system. Very high energy radiation produced during solar flares has effects on our ionosphere almost instantaneously. High-energy particles arriving few minutes later can permanently damage spacecraft. Similarly large volumes of plasma traveling at high velocities (~1000 km/s), produced as an effect of a coronal mass ejection, can have multiple negative effects on our technological dependent society. These effects can be measured everywhere in the solar system, and the HELiophysics Integrated Observatory [HELIO] has built a set of tools that helps to find where these events have been measured, taking into account the speed of the different events and the movement of planets and spacecraft within that time range. HELIO includes 'Features' and 'Event' catalogs similar to what is offered by HEK. It also offers access to solar observations, similar to the VSO, but enhanced with access to observations at other planets through a propagation model to link any event with its origin or its effects. Each of these tools has an independent webservice, therefore they could be easily implemented as a set of independent tools. However, SunPy offers the opportunity to create a better implementation where the data retrieved could interact with the rest of SunPy's features. HELIO implementation on SunPy is in its early development stages.

### Community

One of SunPy's major advantages over its predecessors is that it is being developed as an open source community inside the wide and diverse general scientific python community. While the SolarSoft library is "open source" in terms of the code being freely available, most of the development takes place internally and there is no clear process for contribution from outsiders. In addition to transitioning the solar physics community to Python, SunPy also aims to instill the principals of open source development in the community.

The scientific python community is much more established in other disciplines than it is in solar physics. SunPy is making use of existing scientific python projects, with deeper integration with projects like Astropy and scikit-image possible in the future. This collaboration is another strength that sets the scientific python community apart from other similar solutions.

SunPy has benefited greatly from summer of code schemes. During its first two years (2011, 2012), SunPy participated on the ESA Summer of Code In Space (SOCIS). This program is inspired by Google Summer Of Code (GSOC) and aims to raise the awareness of open source projects related to space, promote the European Space Agency and to improve the existing space-related open-source software. The VSO implementation, and the first graphical user interface (GUI) were developed during these two summer programs. In 2013 SunPy is also taking part in GSOC under the umbrella of the Python Software Foundation (PSF). We are looking forward to the advances this will bring to the capabilities and reach of the project through the work of our two students.

SunPy has also benefited from fledgling input from the solar physics community, for example the implementation of the e-Callisto spectrograph support was enabled by the Astrophysics Research Group at Trinity College Dublin. It is hoped that this kind of contribution from the solar physics community will become the driving force for the project once a core library is in place.

### Future

SunPy 0.3 provides an excellent, flexible base for future expansion of the project. This work has provided the footing for future integration with Astropy. The capabilities of Astropy combined with the overlapping requirements of SunPy and Astropy mean that there is much scope for these two projects to work closely together. SunPy plans to investigate making use of the NDData type of Astropy which is built upon ndarray and combines metadata with

arrays of data, as well as integration of Astropy's WCS and unit implementations.

The goal for SunPy is to develop the project into a flexible package for data analysis and scientific application. While in the long term SunPy aims to become the defacto package for all solar physics data processing and analysis, to achieve this goal it is required that SunPy gains more traction within the solar physics community. This is both to increase the user base and to attract new missions and instruments to adopt Python/SunPy for their data processing pipeline.

The SunPy team would like to thank the organizers of SciPy for the opportunity to present on the SunPy project.

## REFERENCES

[VSO] F. Hill, et al. *The Virtual Solar Observatory A Resource for International Heliophysics Research*, Earth Moon and Planets, 104:315-330, April 2009. DOI: 10.1007/s11038-008-9274-7

[HEK] N. Hurlburt, et al. *Heliophysics Event Knowledgebase for the Solar Dynamics Observatory (SDO) and Beyond*, Solar Physics, 275:67-78, January 2012. DOI: 10.1007/s11207-010-9624-2 arXiv:1008.1291

[HELIO] D. Pérez-Suárez et al. *Studying Sun–Planet Connections Using the Heliophysics Integrated Observatory (HELIO)* Solar Physics, 280:603-621, October 2012. DOI: 10.1007/s11207-012-0110-x

[WCS] W. T. Thompson, *Coordinate systems for solar image data*, A&A 449, 791–803 (2006)

[SSW] S. L. Freeland, B. N. Handy, *Data Analysis with the SolarSoft System*, Solar Physics, v. 182, Issue 2, p. 497-500 (1998)

1. All use of data coming from NASA mission from the Heliophysics Division follows a explicit copyright and Rules of the Road.