

pySPEDAS: Space Physics Environment Data Analysis Software in Python

Eric Grimes
egrimes@igpp.ucla.edu
IHDEA, October 21, 2020

pySPEDAS: Getting Started

Bleeding edge: <https://github.com/spedas/pyspedas>

The screenshot shows the GitHub repository page for `spedas / pyspedas`. The repository has 11 watchers, 33 stars, and 19 forks. The code tab is selected, showing the master branch with 1,133 commits. The README tab displays the contents of `README.md`, which includes information about pySPEDAS, its projects supported, and its dependencies. The repository also features a space-physics python package, a MIT License, and a single release of pySPEDAS 1.0.

Code

master · 1 branch · 1 tag

nickssl Merge pull request #64 from nickssl/master ... 1f43177 2 days ago 1,133 commits

.github/workflows Moved examples to new repository. 6 months ago

pyspedas Minor addition to tests. 2 days ago

.coveragerc Update .coveragerc 8 months ago

.gitattributes Revert "Revert "Merge branch 'master' of https://github.com/spedas/..." 2 years ago

.gitignore Added conda install. 13 months ago

CODE_OF_CONDUCT.md Update CODE_OF_CONDUCT.md 6 months ago

CONTRIBUTING.md Create CONTRIBUTING.md 6 months ago

LICENSE.txt Initial 2 years ago

MANIFEST.in updating MMS non-code files for pypi 12 months ago

README.md Updating required python version to 3.6+ 29 days ago

requirements.txt Adding astropy to the required dependencies to fix bug with time con... 25 days ago

setup.py version bump for pypi 7 days ago

README.md

pySPEDAS

[build](#) [passing](#) [coverage](#) [81%](#) [pypi](#) [v1.0.12](#) [license](#) [MIT](#) [status](#) [stable](#)

pySPEDAS is an implementation of the SPEDAS framework in python.

The Space Physics Environment Data Analysis Software (**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.

Projects Supported

- Advanced Composition Explorer (ACE)
- Arase (ERG)
- Cluster
- Colorado Student Space Weather Experiment (CSSWE)
- Deep Space Climate Observatory (DSCOVR)
- Equator-S
- Fast Auroral Snapshot Explorer (FAST)
- Geotail
- Geostationary Operational Environmental Satellite (GOES)
- Imager for Magnetopause-to-Aurora Global Exploration (IMAGE)
- Mars Atmosphere and Volatile Evolution (MAVEN)
- Magnetic Induction Coil Array (MICA)
- Magnetospheric Multiscale (MMS)

About

SPEDAS routines for Python

[space-physics](#) [python](#)

[Readme](#)

[MIT License](#)

Releases 1

[pySPEDAS 1.0](#) (Latest) on Jun 16

Packages

No packages published

Used by 7

Contributors 8

Languages

Python 100.0%

pySPEDAS: Getting Started

- Internally, pySPEDAS uses pyTplot to load the data into variables and create plots:

<https://github.com/MAVENSDC/PyTplot>

<https://pytplot.readthedocs.io>

Getting Started: Installing

pip install pyspedas

or

pip install pyspedas --upgrade

Getting Started: Data Directories

- By default, the data are placed in the directory where pySPEDAS is installed; you can configure this with the **SPEDAS_DATA_DIR** environment variable
- Most missions also have their own environment variable, which overrides **SPEDAS_DATA_DIR**, e.g., **MMS_DATA_DIR**
- These environment variables also work with our latest IDL versions; so if you load data in IDL, pySPEDAS should automatically find the data files locally

Getting Started: Data Sources

- These depend on the mission; e.g., for MMS, the default data source is the official Science Data Center at LASP, and data can be loaded from NASA SPDF using the **spdf** keyword to the load routine
- Most other missions (but not all) load data from NASA SPDF by default

Missions Supported

[Advanced Composition Explorer \(ACE\)](#)

[Arase \(ERG\)](#) - still experimental

[Cluster](#) - currently loads data from SPDF

[Colorado Student Space Weather Experiment \(CSSWE\)](#)

[Deep Space Climate Observatory \(DSCOVR\)](#)

[Equator-S](#)

[Fast Auroral Snapshot Explorer \(FAST\)](#)

[Geotail](#)

[Geostationary Operational Environmental Satellite \(GOES\)](#)

[Imager for Magnetopause-to-Aurora Global Exploration \(IMAGE\)](#)

[Mars Atmosphere and Volatile Evolution \(MAVEN\)](#)

[Magnetic Induction Coil Array \(MICA\)](#)

[Magnetospheric Multiscale \(MMS\)](#)

[OMNI](#)

[Polar Orbiting Environmental Satellites \(POES\)](#)

[Polar](#)

[Parker Solar Probe \(PSP\)](#)

[Van Allen Probes \(RBSP\)](#)

[Solar Terrestrial Relations Observatory \(STEREO\)](#)

[Time History of Events and Macroscale Interactions during Substorms \(THEMIS\)](#)

[Two Wide-Angle Imaging Neutral-Atom Spectrometers \(TWINS\)](#)

[Ulysses](#)

[Wind](#)

Note: while we can load CDFs from most instruments on each of these missions, post-processing of the data products for most missions are not at the same fidelity as our IDL tools

Load Routines

- Data are loaded by functions that follow the general form:

`pyspedas.mission.instrument()`

e.g., for MMS FPI data:

`pyspedas.mms.fpi()`

or RBSP EMFISIS data:

`pyspedas.rbsp.emfisis()`

Keywords

- Just as with our IDL load routines, the options are set with some standard keywords, e.g.,
 - `trange` - time range; e.g, `['2015-12-15', '2015-12-16']`
 - `level` - level of the data; e.g., `'l2'`, `'l3'`, etc
 - + many more
- To see all of the keywords supported by a load routine, use the Python help function, e.g.,
`help(pyspedas.mms.fpi)`

Keywords

```
Help on function mms_load_fpi in module pypespedas.mms:

mms_load_fpi(*args, **kwargs)                                     [source code]
    This function loads FPI data into tplot variables

    Parameters:
        trange : list of str
            time range of interest [starttime, endtime] with the format
            'YYYY-MM-DD', 'YYYY-MM-DD' or to specify more or less than a day
            ['YYYY-MM-DD/hh:mm:ss', 'YYYY-MM-DD/hh:mm:ss']

        probe : str or list of str
            list of probes, valid values for MMS probes are ['1','2','3','4'].

        data_rate : str or list of str
            instrument data rates for FPI include 'brst', 'fast'. The
            default is 'srvy'.

        level : str
            indicates level of data processing. the default if no level is specified is 'L2'

        datatype : str or list of str
            Valid datatypes for FPI are:
            'des-moms', 'dis-moms' (default)
            'des-dist', 'dis-dist'

        get_support_data: bool
            Data with an attribute "VAR_TYPE" with a value of "support_data"
            will be loaded into tplot. By default, only loads in data with a
            "VAR_TYPE" attribute of "data".

        time_clip: bool
            Data will be clipped to the exact trange specified by the trange keyword.

        varformat: str
            The file variable formats to load into tplot. Wildcard character
            "*" is accepted. By default, all variables are loaded in.

        suffix: str
            The tplot variable names will be given this suffix. By default,
            no suffix is added.

        center_measurement: bool
            If True, the CDF epoch variables are time-shifted to the middle
            of the accumulation interval by their DELTA_PLUS_VAR and
            DELTA_MINUS_VAR variable attributes

        notplot: bool
            If True, then data are returned in a hash table instead of
            being stored in tplot variables (useful for debugging, and
            access to multi-dimensional data products)

        available: bool
            If True, simply return the available data files (without downloading)
```

Simple Example

```
[>>> import pyspedas
[>>>
[>>> pyspedas.mms.fpi(trange=['2015-10-16/13:06', '2015-10-16/13:07'], center_measurement=True, data_rate='brst', datatype='dis-moms')
13-Oct-20 13:59:50: Loading /Volumes/data/data/mms/mms1/fpi/brst/l2/dis-moms/2015/10/16/mms1_fpi_brst_l2_dis-moms_20151016130524_v3.3.0.cdf
[>>> 
Loaded variables:
mms1_dis_errorflags_brst
mms1_dis_compressionloss_brst
mms1_dis_startdelphi_count_brst
mms1_dis_startdelphi_angle_brst
mms1_dis_sector_despinp_brst
mms1_dis_energyspectr_px_brst
mms1_dis_energyspectr_mx_brst
mms1_dis_energyspectr_py_brst
mms1_dis_energyspectr_my_brst
mms1_dis_energyspectr_pz_brst
mms1_dis_energyspectr_mz_brst
mms1_dis_energyspectr_omni_brst
mms1_dis_spectr_bg_brst
mms1_dis_numberdensity_bg_brst
mms1_dis_numberdensity_brst
mms1_dis_densityextrapolation_low_brst
mms1_dis_densityextrapolation_high_brst
mms1_dis_bulkv_dbcs_brst
mms1_dis_bulkv_spintone_dbcs_brst
mms1_dis_bulkv_gse_brst
mms1_dis_bulkv_spintone_gse_brst
mms1_dis_prestensor_dbcs_brst
mms1_dis_prestensor_gse_brst
mms1_dis_pres_bg_brst
mms1_dis temptensor_dbcs_brst
mms1_dis temptensor_gse_brst
mms1_dis_heatq_dbcs_brst
mms1_dis_heatq_gse_brst
mms1_dis_temppara_brst
mms1_dis_tempperp_brst
['mms1_dis_errorflags_brst', 'mms1_dis_compressionloss_brst', 'mms1_dis_startdelphi_count_brst', 'mms1_dis_startdelphi_angle_brst', 'mms1_dis_sector_despinp_brst', 'mms1_dis_energyspectr_px_brst', 'mms1_dis_energyspectr_mx_brst', 'mms1_dis_energyspectr_py_brst', 'mms1_dis_energyspectr_my_brst', 'mms1_dis_energyspectr_pz_brst', 'mms1_dis_energyspectr_mz_brst', 'mms1_dis_energyspectr_omni_brst', 'mms1_dis_spectr_bg_brst', 'mms1_dis_numberdensity_bg_brst', 'mms1_dis_numberdensity_brst', 'mms1_dis_densityextrapolation_low_brst', 'mms1_dis_densityextrapolation_high_brst', 'mms1_dis_bulkv_dbcs_brst', 'mms1_dis_bulkv_spintone_dbcs_brst', 'mms1_dis_bulkv_gse_brst', 'mms1_dis_bulkv_spintone_gse_brst', 'mms1_dis_prestensor_dbcs_brst', 'mms1_dis_prestensor_gse_brst', 'mms1_dis_pres_bg_brst', 'mms1_dis temptensor_dbcs_brst', 'mms1_dis temptensor_gse_brst', 'mms1_dis_heatq_dbcs_brst', 'mms1_dis_heatq_gse_brst', 'mms1_dis_temppara_brst', 'mms1_dis_tempperp_brst']
>>> 
```



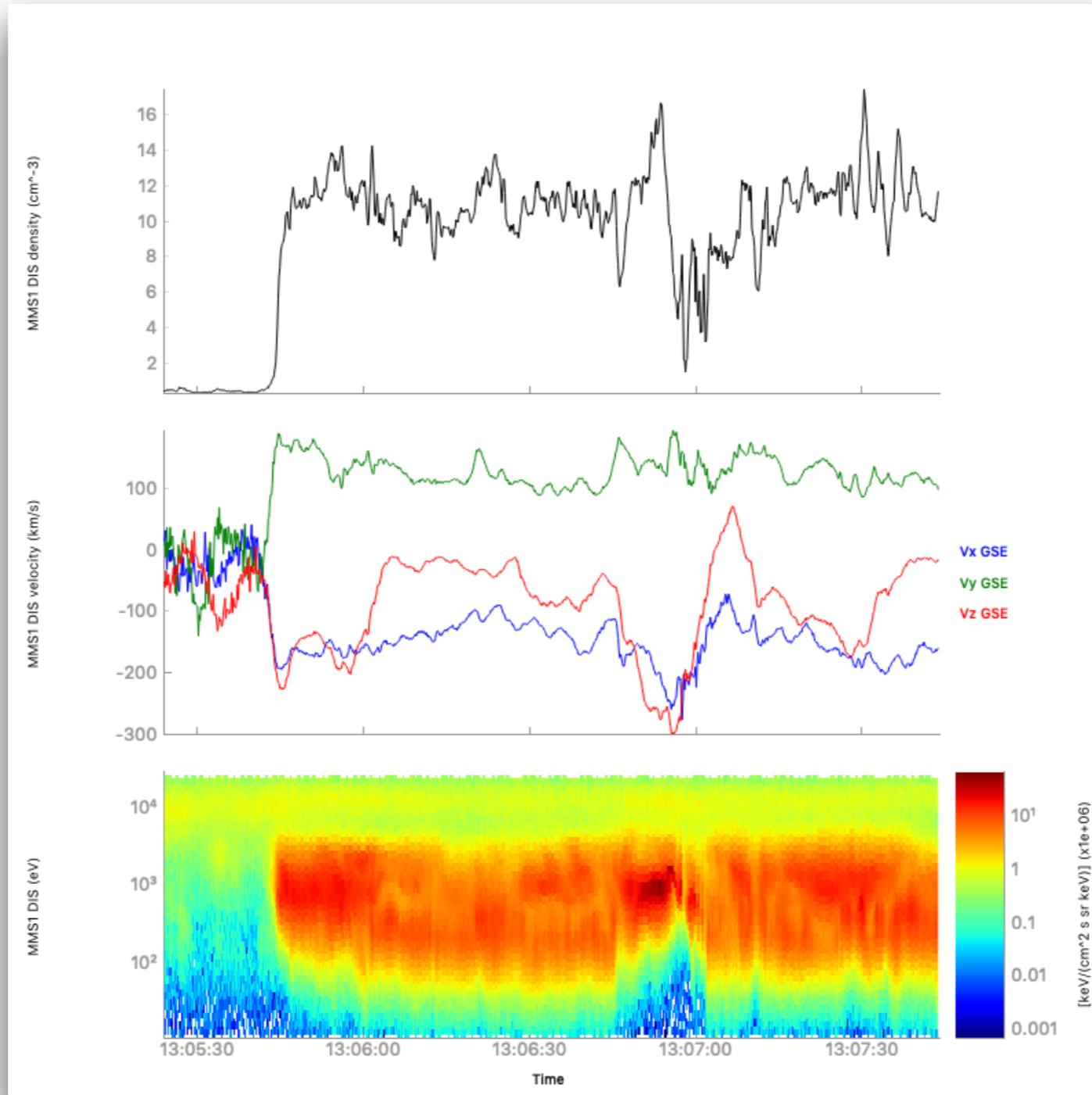
Play Keynote Live



Table Chart Text Shape Media Comment

Simple Example

```
>>> from pyplot import tplot  
>>>  
>>> tplot(['mms1_dis_numberdensity_brst', 'mms1_dis_bulky_gse_brst', 'mms1_dis_energyspectr_omni_brst'])
```



How we're testing

- We're using Github Actions to run unit tests that load the data; we currently have > 80% test coverage

The screenshot shows a GitHub Actions build log for a pull request. The repository is `spedas / pypspedas`. The pull request is titled "Merge pull request #64 from nickssl/master". The build is for the branch `master` and the commit `1f43177`. The build is labeled "build (macos-latest, 3.8)" and was triggered by a push. It succeeded 2 days ago in 56m 33s. The log details the steps taken:

- Set up job (23s)
- Run actions/checkout@v2 (28s)
- Set up Python 3.8 (1s)
- Install dependencies (Windows) (0s)
- Install dependencies (macOS) (2m 54s)
- Lint with flake8 (17s)
- Test with unittest (52m 29s)

The "Test with unittest" step shows the command `Run coverage run -a -m pypspedas.mms.tests.load_routine_tests` running. The log output includes numerous error messages related to failed OpenGL renderer attempts and dependency downloads from <https://spdf.gsfc.nasa.gov/pub/data/mms1/aspac/srvy/l2/2015/10/>.

Validating the data products from IDL

- We also have IDL unit tests that load the data in IDL and Python and check that the data products match

```
function mms_python_validation_ut::test_aspoc_default
    mms_load_aspoc, probe=1, trange=['2015-10-16', '2015-10-17']
    py_script = ["from pyspedas import mms_load_aspoc", "mms_load_aspoc(probe=1, trange=['2015-10-16', '2015-10-17'])"]
    vars = ['mms1_aspoc_ionc_l2']
    return, spd_run_py_validation(py_script, vars)
end

function mms_python_validation_ut::test_fgm_default
    mms_load_fgm, probe=1, trange=['2015-10-16', '2015-10-17']
    py_script = ["from pyspedas import mms_load_fgm", "mms_load_fgm(probe=1, trange=['2015-10-16', '2015-10-17'])"]
    vars = ['mms1_fgm_b_gse_srvy_l2', 'mms1_fgm_b_gsm_srvy_l2']
    return, spd_run_py_validation(py_script, vars)
end

function mms_python_validation_ut::test_fgm_brst
    mms_load_fgm, data_rate='brst', probe=4, trange=['2015-10-16/13:06', '2015-10-16/13:07']
    py_script = ["from pyspedas import mms_load_fgm", "mms_load_fgm(data_rate='brst', probe=4, trange=['2015-10-16/13:06', '2015-10-16/13:07'])"]
    vars = ['mms4_fgm_b_gse_brst_l2', 'mms4_fgm_b_gsm_brst_l2']
    return, spd_run_py_validation(py_script, vars)
end

function mms_python_validation_ut::test_fgm_curlometer
    mms_load_fgm, data_rate='brst', probe=[1, 2, 3, 4], trange=['2015-10-30/05:15:45', '2015-10-30/05:15:48'], /get_fgm_ephem, /time_clip
    fields = 'mms'+['1', '2', '3', '4']+ '_fgm_b_gse_brst_l2'
    positions = 'mms'+['1', '2', '3', '4']+ '_fgm_r_gse_brst_l2'

    mms(curl, trange=['2015-10-30/05:15:45', '2015-10-30/05:15:48'], fields=fields, positions=positions)
    py_script = ["from pyspedas import mms_load_fgm", $
                 "from pyspedas.mms import curlometer", $
                 "mms_load_fgm(time_clip=True, data_rate='brst', probe=[1, 2, 3, 4], trange=['2015-10-30/05:15:45', '2015-10-30/05:15:48']), $"
                 "positions = ['mms1_fgm_r_gse_brst_l2', 'mms2_fgm_r_gse_brst_l2', 'mms3_fgm_r_gse_brst_l2', 'mms4_fgm_r_gse_brst_l2'], $"
                 "fields = ['mms1_fgm_b_gse_brst_l2', 'mms2_fgm_b_gse_brst_l2', 'mms3_fgm_b_gse_brst_l2', 'mms4_fgm_b_gse_brst_l2'], $"
                 "curlometer(fields=fields, positions=positions)"]
    vars = ['baryb', 'curlB', 'divB', 'jtotal', 'jpar', 'jperp', 'alpha', 'alphaparallel']
    return, spd_run_py_validation(py_script, vars, tol=1e-4)
end
```

Finding More Examples

- We're using Jupyter notebooks to create examples; to find these, visit: <https://github.com/spedas>

The screenshot shows the GitHub repository page for the SPEDAS organization. The top navigation bar includes links for Repositories (4), Packages, People (3), Teams, Projects, and Settings. Below the header are search and filter fields for 'Find a repository...', 'Type: All', 'Language: All', 'Customize pins', and a green 'New' button. The main content area displays four repository cards:

- pyspedas**: SPEDAS routines for Python. Language: Python. License: MIT. Stars: 19. Updated: 2 days ago.
- pyspedas_examples**: Examples of using pySPEDAS. Language: Python. License: MIT. Stars: 1. Updated: Sep 13.
- mms-examples**: Examples of using pySPEDAS to access MMS data. Language: Jupyter Notebook. License: MIT. Stars: 3. Updated: Aug 26.
- bleeding_edge**: SPEDAS bleeding edge (IDL). Language: IDL. Stars: 3. Updated: Aug 3.

Two blue arrows point from the bottom left towards the first two repository cards: **pyspedas** and **pyspedas_examples**.

Top languages
Python (blue dot), IDL (brown dot), Jupyter Notebook (orange dot)

People
3 >
Profile picture of a person with a beard, three GitHub user icons.

Invite someone