

Getting Started with PySPEDAS

Jim Lewis

jwl@ssl.Berkeley.edu

Tutorial agenda

- Motivation for developing and using PySPEDAS
- Setting up a Python environment
- Installing PySPEDAS
- Some PySPEDAS core concepts
- Some simple PySPEDAS examples
- Q & A
- Time permitting: live demo

PySPEDAS features and capabilities

- Load data from many different providers and formats into a common environment
 - Support for directly loading data from 30+ missions (including THEMIS, MMS, ERG/Arase, RBSP/Van Allen probes, Parker Solar Probe, FAST, WIND, many others)
 - Load via CDAWeb web service
 - Load via HAPI
- Analysis and Modeling tools
 - Interface to native Python geopack package
 - Field-aligned coordinates, minimum variance analysis, wave polarization, etc
- Plotting tools
 - Line plots
 - Spectrograms
 - Being worked on: interactive plots (zoom in on specific time ranges, see time/data values at cursor, etc)
 - Also coming soon: 3-D particle distribution interactive visualization tools (courtesy of ERG/Arase team, similar to IDL ISEE-3D tool)

Future development plans

- More missions and datasets
- Improved modeling tools (e.g. additional GEOPACK models, field line tracing)
- More interactivity with plots (e.g. mousing over to get times and data values)
- More wrappers for working with particle datasets for additional missions (e.g. THEMIS)

How to get help

- Documentation wiki: <http://spedas.org/wiki>
- Also: <https://pyspedas.readthedocs.io>
- Github issue tracker: <https://github.com/spedas/pyspedas/issues>
- Email questions and problem reports:
Themis_Science_Support@ssl.Berkeley.edu

- * [HTML Documentation](#) - Documentation of SPEDAS functions, with source code
- * [User's Guide](#) - Documentation for users
- * [Developer's Guide](#) - Documentation for plugin developers
- * [Crib Sheets](#) - Step-by-step usage scenarios
- * [Tips and Tricks](#) - SPEDAS tips and tricks
- * [Data Model](#) - Description of the data model used in SPEDAS
- * [Time Handling](#) - Description of how times are used in SPEDAS
- * [3D Data Structures](#) - Description of the data structures used in many SPEDAS particle data sets.
- * [3D Data Conversion Factors](#) - Description of the conversion factors used for different data quantities, for particle data.
- * [Autoplot Interface](#) - Description of the Autoplot interface
- * [Heliophysics Application Programmer's Interface \(HAPI\)](#) - Description of how to access HAPI data using SPEDAS

External Resources [\[edit\]](#)

- * [LICENSE.txt](#) - License for the SPEDAS framework
- * [SPEDAS Blog](#) - SPEDAS Developer Blog and Announcements
- * [SPEDAS ChangeLog](#) - Recent changes to the bleeding edge
- * [Overview Plots](#) - Summary plots generated with SPEDAS
- * [IDL Documentation](#) - IDL Documentation by Harris Geospatial Solutions
- * [Wiki User's Guide](#) - Information on using the wiki software.
- * Information on the [coordinate transformation library Rocotlib](#) and on a [method for computing wave polarization](#)

Python [\[edit\]](#)

- * [Getting Started with pySPEDAS](#) - SPEDAS software in the Python programming language
- * [PySPEDAS bleeding edge](#) - PySPEDAS bleeding edge, hosted at GitHub
- * [PySPEDAS documentation](#) - PySPEDAS documentation, hosted at ReadTheDocs

FIRST STEPS

Getting Started

FEATURES

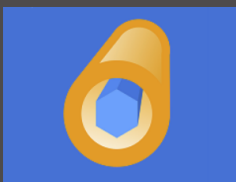
Load Routines

Analysis Tools

Magnetic Field Models

Coordinate Systems

Utilities



Webinoly Open Source Modern LEMP Stack that vastly simplifies all your WebServer Start Testing Now!

Ad by EthicalAds · Community Ad

PySPEDAS Documentation

build passing
coverage 66%
pypi v1.4.37
lgm grade no longer available
status stable
license MIT

PySPEDAS is an implementation of the Space Physics Environment Data Analysis Software (SPEDAS) framework in Python.

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.

PySPEDAS and [pyTplot](#) make creating multi-mission, multi-instrument figures simple, e.g., to create a figure showing magnetometer data from [Solar Orbiter](#), [Parker Solar Probe](#), [MMS](#), and [THEMIS](#),

```

import pyspedas
from pytplot import tplot

time_range = ['2020-04-20/06:00', '2020-04-20/08:00']

pyspedas.solo.mag(trange=time_range, time_clip=True)
pyspedas.psp.fields(trange=time_range, time_clip=True)
pyspedas.mms.fgm(trange=time_range, time_clip=True, probe=2)
pyspedas.themis.fgm(trange=time_range, time_clip=True, probe='d')

tplot(['B_RTN', 'psp_fld_l2_mag_RTN', 'mms2_fgm_b_gsm_srvy_l2_bvec', 'thd_fgs_gsm'])

```

Label issues and pull requests for new contributors Dismiss


Now, GitHub will help potential first-time contributors [discover issues](#) labeled with good first issue

Filters Labels 21 Milestones 0 New issue

103 Open ✓ 94 Closed Author Label Projects Milestones Assignee Sort

Bug with MMS FPI error flag bars when a timebar is set bug MMS
#520 opened yesterday by supervised

Improve flexibility of time string formats in timebar enhancement pytplot
#519 opened yesterday by supervised

timespan needs internal conversion to datetime64[ns] bug pytplot 
#518 opened 2 days ago by jameswilburlewis

Improve error handling in mms_feeps_pad bug
#517 opened 2 days ago by supervised

Update tdpwrspc to set plot metadata enhancement
#516 opened 2 days ago by supervised

Improve the error handling in mms_qcotrans bug MMS 
#515 opened 2 days ago by supervised

Getting Python

- Python 3.8 or later is required (current version is 3.11.4)
 - Some packages that PySPEDAS depends on may not be compatible or well-tested with 3.11; 3.10 might be a safer choice
- python.org is a great place to start: downloads & installation instructions, documentation, beginner's guide
- Another option (even if you already have a suitable Python installation!) is Anaconda (anaconda.com), which bundles the Python interpreter with additional scientific Python packages and other tools (including interactive development environments: highly recommended!)
- It is fine to have multiple Python versions installed on the same system.

Home

Environments

Learning

Community

Anaconda Notebooks New!

Cloud notebooks with hundreds of packages ready to code.

[Learn More](#)

[Documentation](#)


[Anaconda Blog](#)



All applications ▾ on

base (root) ▾


Channels

DataSpell

DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

[Install](#)




CMD.exe Prompt

0.1.1

Run a cmd.exe terminal with your current environment from Navigator activated

[Launch](#)




JupyterLab

3.5.3

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.


[Launch](#)



Jupyter Notebook

6.5.4


Web-based, interactive computing notebook environment. Edit and run human-readable



Powershell Prompt

0.0.1

Run a Powershell terminal with your current environment from Navigator activated



PyCharm Community

2021.3

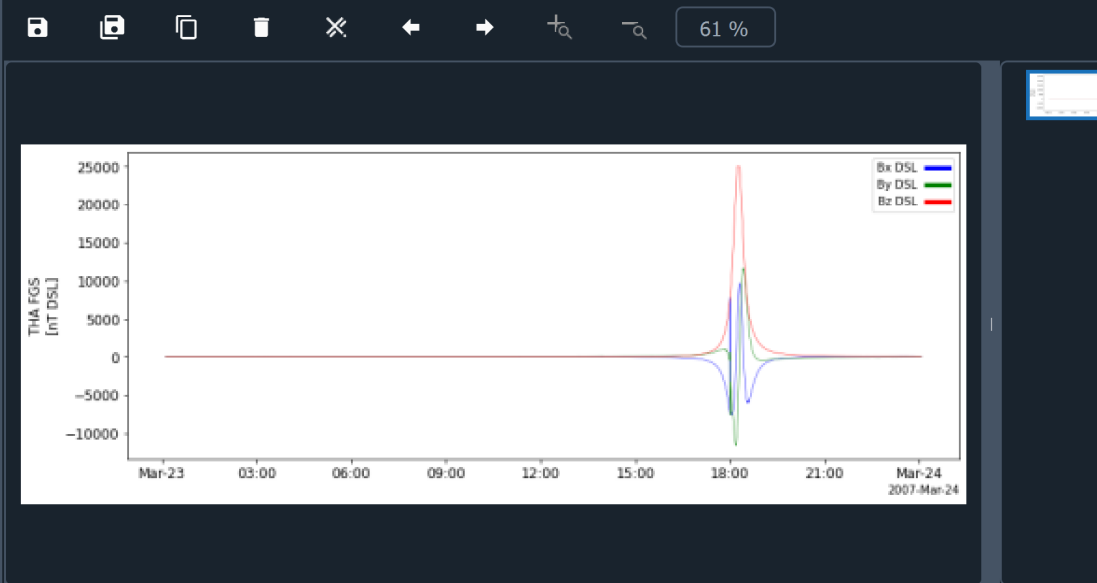
An IDE by JetBrains for pure Python development. Supports code completion

Python IDEs (interactive development environments)

- An IDE is not strictly needed – you could get by with just a text editor and a command line. But you'll probably want one.
 - Hover or click on function names to see documentation, parameter hints, etc
 - Jump from a function call point to the function definition
 - Integration with version control systems (git, svn, etc)
 - See potential problems while coding
 - Run scripts and tests easily and see results without leaving the IDE
 - Debugging: set breakpoints, step through code and inspect variable values
- The Spyder IDE is bundled with Anaconda (anaconda.org); PyCharm (www.jetbrains.com/pycharm) and Visual Studio Code (code.visualstudio.com) are some other popular free IDEs.

C:\Users\james\.spyder-py3\temp.py

```
temp.py x
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4 This is a temporary script file.
5 """
6
7
8 import pyspedas
9 from pyqtplot import tplot
10
11 pyspedas.themis.fgm(trange=['2007-03-23','2007-03-04'],probe='a')
12 tplot('tha_fgs_dsl')
```

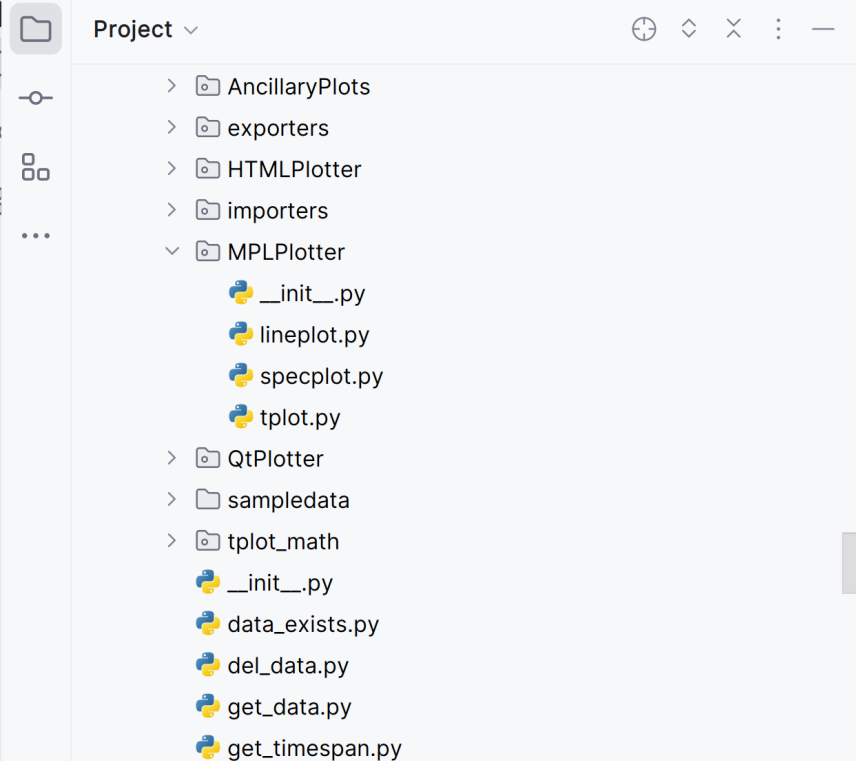


Help Variable Explorer Plots Files

```
Console 1/A x
In [4]: runfile('C:/Users/james/.spyder-py3/temp.py', wdir='C:/Users/james/.spyder-py3')
14-Jun-23 10:05:10: Downloading remote index: http://themis.ssl.berkeley.edu/data/themis/tha/12/fgm/2007/
14-Jun-23 10:05:11: File is current: themis_data/tha/12/fgm/2007/
tha_12_fgm_20070323_v01.cdf
```

Warning

Figures now render in the Plots pane by default. To make them also appear inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.



```
39 elif keyword == 'hours':
40     dt *= 3600
41 elif keyword == 'minutes':
42     dt *= 60
43 elif keyword == 'seconds':
44     dt *= 1
45 else:
46     logging.warning("Invalid 'keyword' option %s.\nEnum(None, 'hours', 'minutes', 'seconds', 'da
47
48 if not isinstance(t1, (int, float, complex)):
49     t1 = tplot_utilities.str_to_int(t1)
50 t2 = t1+dt
51 xlim(t1, t2)
52
53 return
```

timespan()

Run Python tests in misc_tests.py ×



✔ Tests passed: 3 of 3 tests - 123 ms

✔ Test Results		123 ms
▾	✔ misc_tests	123 ms
▾	✔ UtilTestCases	123 ms
✔	test_dailynames (Te	27 ms
✔	test_tcopy (Test tcc	28 ms
✔	test_tkm2re	68 ms

```
13-Jun-23 15:35:07: test copied to another-copy
13-Jun-23 15:35:07: tcopy error: No pyplot variables found.
13-Jun-23 15:35:07: tcopy error: List with the names_in does not match list          with the names out.
13-Jun-23 15:35:07: No tplot variables found: doesnt_exist
13-Jun-23 15:35:07: Number of output variable names (newname) should match the number of input variables.
```

Python virtual environments and package management

- Unlike IDL, the Python ecosystem is heavily dependent on open-source packages to provide functionality beyond the core language.
 - numpy for numerical algorithms, PyQt or matplotlib for graphics, CDF and NetCDF libraries, etc.
 - It is typical for a Python package (like PySPEDAS) to depend on several other packages, sometimes with specific versioning requirements.
 - The more packages installed in a single environment, the greater the potential for versioning conflicts
- Python provides for “virtual environments”, which allow different environments for different purposes to coexist on the same system, rather than installing all packages under the main Python installation.

Python virtual environments and package management

- We highly recommend installing PySPEDAS in a dedicated virtual environment.
 - Anaconda virtual environments can be managed via Anaconda Navigator, or using the command line: `conda create --name my_new_environment`
 - Your IDE may offer the option to create a virtual environment when creating a new project.
 - Python can create a new virtual environment for you, with the command line: `python -m venv /path/to/new/virtual/environment`

Python virtual environments and package management

- There are two styles of package management commonly used in the Python community:
 - conda (if using an Anaconda python distribution): `conda install package-name`
 - pip (Pip Installs Packages): `pip install package-name`
 - Each system has its own repository of Python packages and ways of tracking locally installed packages. (So conda is not aware of packages installed via pip, and vice versa).
 - Many packages are available via both methods, but PySPEDAS is only available via pip (for now).

How to get PySPEDAS

- Installing for the first time: `pip install pyspedas`
- Upgrading an existing installation:
`pip install --upgrade pyspedas`
- This installs (or updates) the latest released version of the PySPEDAS source code, along with all the other packages it depends on.
- Another method (for example, if you're interested in modifying or contributing to PySPEDAS) is to get the source code directly from our GitHub repository. Your IDE might be able to create a project directly from Github, or you can use the command:
`git clone https://github.com/spedas/pyspedas.git /path/to/pyspedas`

mms-examples Public

Examples of using PySPEDAS to access MMS data

📄 Jupyter Notebook 🏢 MIT 👤 2 ⭐ 7 🔄 0 🐞 0 Updated 2 days ago



pyspedas Public

Python-based Space Physics Environment Data Analysis Software

space space-physics science-research heliophysics python

📄 Python 🏢 MIT 👤 49 ⭐ 112 🔄 103 🐞 0 Updated last week



spedas-scripts Private

Scripts for the SPEDAS changelog, and mirroring the bleeding edge to GitHub

📄 HTML 👤 0 ⭐ 0 🔄 0 🐞 0 Updated 2 weeks ago



bleeding_edge Public

IDL-based Space Physics Environment Data Analysis Software (bleeding edge)

spacecraft space-physics

📄 IDL 👤 0 ⭐ 8 🔄 38 (1 issue needs help) 🐞 0 Updated 2 weeks ago



pyspedas_examples Public

Examples of using pySPEDAS

📄 Python 🏢 MIT 👤 1 ⭐ 5 🔄 0 🐞 0 Updated on May 4



pyspedas-validation Public

Tools for validating that the data products in pySPEDAS match those in IDL

📄 IDL 🏢 MIT 👤 0 ⭐ 1 🔄 0 🐞 0 Updated on Apr 8, 2022



PySPEDAS Concepts

- Load routines: Download data from mission-specific or multi-mission archives (e.g. THEMIS, MMS, SPDF), maintain local cache of downloaded data files, convert CDF or NetCDF data to tplot variables.
- Tplot variables: Basic data structure on which SPEDAS and PySPEDAS are built. Maps strings (tplot variable names) to time series data arrays and metadata
- Metadata: Information that describes important data attributes needed by plotting and analysis tools (e.g. units, coordinate systems, plot labels, energy channel values for spectra, etc.)
- Plotting tools: “tplot” routine for creating line plots or spectrograms of tplot variables, helper routines for setting per-variable or per-plot options. Based on matplotlib package.
- Analysis tools: Manipulate tplot variables in various ways. PySPEDAS includes many tools for performing arithmetic operations, interpolation, and coordinate transformations. Also includes more complicated tasks such as generation of plasma moments from 3-D particle distributions, or wave polarization analysis. Many tools are implemented using the numpy library.

Using Load Routines

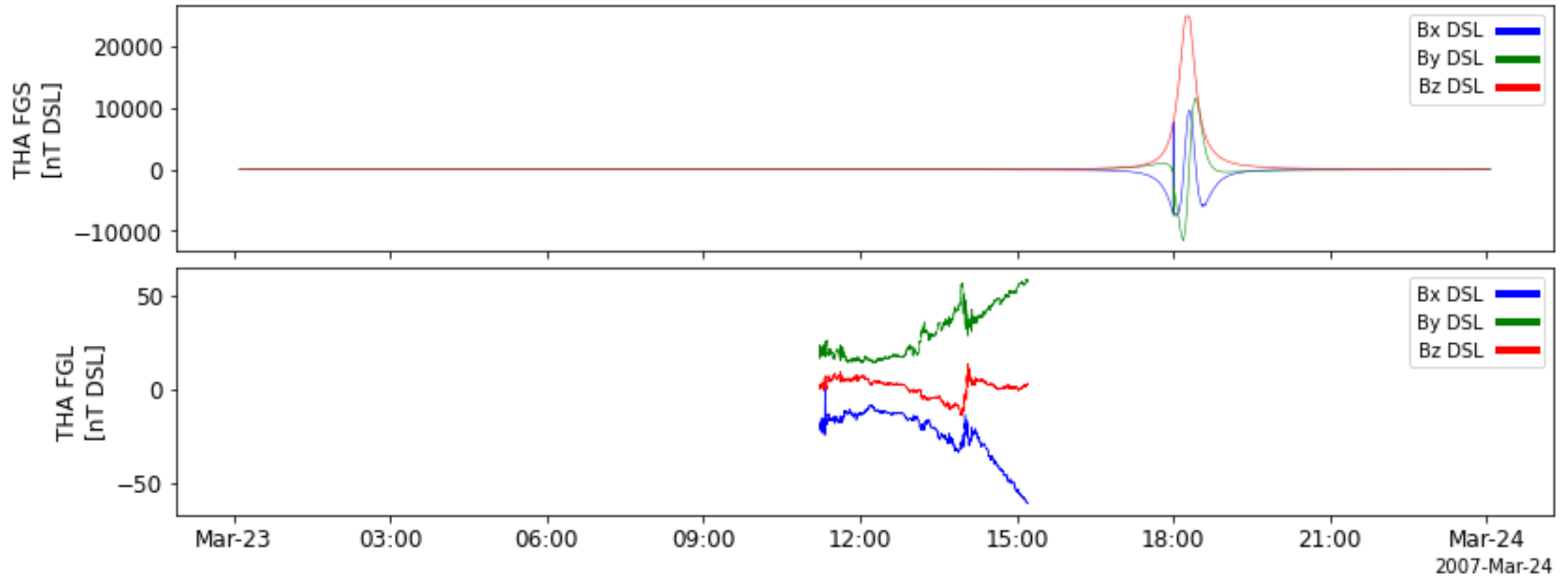
- Use python 'import' statements to make load routines available in your program. "import pyspedas" makes all the supported mission namespaces available. Many missions follow a naming convention like `pyspedas.MISSION.INSTRUMENT()` to access their load routines.
- Commonly supported load routine parameters:
 - `trange` : Specify a time range for data to be loaded (array of start and end times, represented as string-valued timestamps or double precision Unix times)
 - `probe`: For missions that support multiple probes, specify one or more probes to load data for
 - `level`: Missions typically publish data at various calibration levels, e.g. `level='l1'` (for raw, uncalibrated data), `level='l2'` (calibrated data in engineering units and geophysical coordinates), to l3 and above (highly processed/key parameter data)
 - `datatype`: Specify a subset of available variables in the requested dataset to be loaded
 - `get_support_data`: A flag (True/False) commonly used to request loading ancillary data that may be needed for downstream analysis tools

Example 1: Load and plot THEMIS FGM data

Basic PySPEDAS example: Import necessary modules, load THEMIS FGM data for probe THEMIS-A, for time range 2007-03-23 to 2007-03-24, print a list of tplot variables, and plot two of them.

```
1 import pyspedas
2 from pytplot import tplot_names, get_data, tplot
3
4 pyspedas.themis.fgm(trange=['2007-03-23', '2007-03-24'], probe='a')
5 tplot_names()
6 tplot(['tha_fgs_ds1', 'tha_fgl_ds1'])
7
```

Example 1, PySPEDAS plot output



Working with tplot variables

- The pytplot package contains several routines for listing, creating, or extracting information from tplot variables. To use specific routines, import them from pytplot, e.g:

```
from pytplot import get_data, store_data, tplot_names
```
- `tplot_names(pattern)` prints tplot names currently loaded and matching the given wildcard pattern
- Many missions follow a naming convention like `PROBE_MISSION_INSTRUMENT_DATATYPE` for their variable names. There may also be suffixes to indicate data levels, units, or coordinate systems
- `get_data('tplot_variable_name')` is used to retrieve the timestamps, data values, or metadata for a given variable as normal Python data structures.

Example 1A, tplot variables

Examine the data and metadata for one of the tplot variables just loaded

```
data_arrays = get_data('tha_fgs_dsl')
print("Structure returned by get_data()", data_arrays)
data_times=data_arrays.times
data_vals=data_arrays.y
print("Data times",data_times)
print("Data values",data_vals)
```

Executed at 2023.10.04 23:58:17 in 3ms

Example 1A, console output

```
Structure returned by get_data() variable(times=array([1.17460840e+09, 1
.17460840e+09, 1.17460840e+09, ... ,
    1.17469467e+09, 1.17469467e+09, 1.17469467e+09])), y=array([[ 16.50905  ,
-8.597386  ,  14.1494465],
[ 16.497149  , -8.618044  ,  14.173288  ],
[ 16.53812   , -8.73796   ,  14.173288  ],
... ,
```

Example 1A, getting tplot variable metadata

Extract the metadata from the same tplot variable

```
md=get_data('tha_fgs_dsl',metadata=True)
print("Metadata structure:")
print(md)
print("")
print("Metadata CDF.LABELS entries:", md["CDF"]["LABELS"])
print("")
print("Metadata data_att entries:", md["data_att"])
print("")
print("Metadata plot_options entries:",md["plot_options"])
```

Example 1A, metadata console output

```
Metadata CDF.LABELS entries: ['Bx FGS-A', 'By FGS-A', 'Bz FGS-A']
```

```
Metadata data_att entries: {'coord_sys': 'DSL', 'units': 'nT DSL',  
'depend_1_units': 'nT DSL', 'depend_2_units': None, 'depend_3_units': None}
```

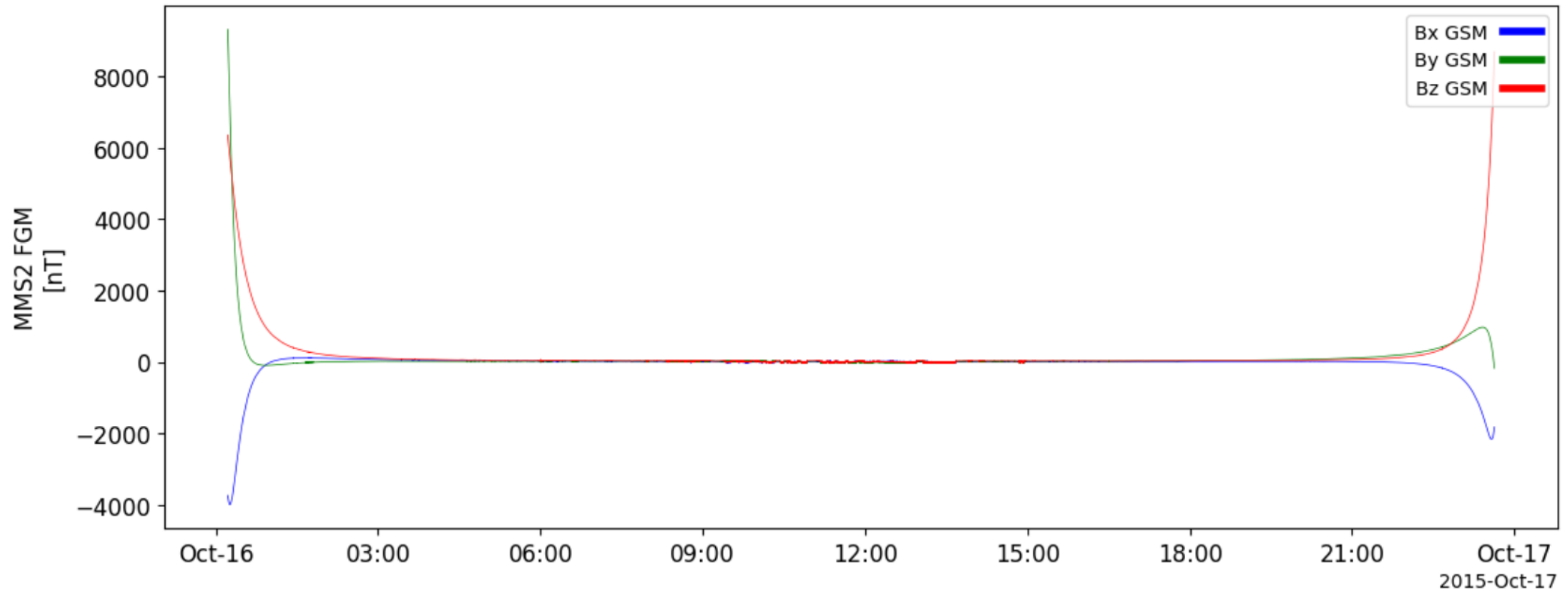
```
Metadata plot_options entries: {'xaxis_opt': {'axis_label': 'Time', 'crosshair':  
'X', 'x_axis_type': 'linear'}, 'yaxis_opt': {'axis_label': 'THA FGS', 'crosshair':  
'Y', 'y_axis_type': 'linear', 'y_range': [-11531.753, 24993.445], 'axis_subtitle':  
'[nT DSL]', 'legend_names': ['Bx DSL', 'By DSL', 'Bz DSL']}, 'zaxis_opt':  
{'axis_label': 'tha_fgs_dsl', 'crosshair': 'Z', 'z_axis_type': 'linear'},
```

Example 2, MMS FGM data

Load and plot MMS FGM data

```
mms_tplot_vars = pyspedas.mms.fgm(probe=[1,2], trange=['2015-10-16', '2015-10-17'])  
print(mms_tplot_vars)  
tplot('mms2_fgm_b_gsm_srvy_l2_bvec')
```

Example 2, PySPEDAS plot output



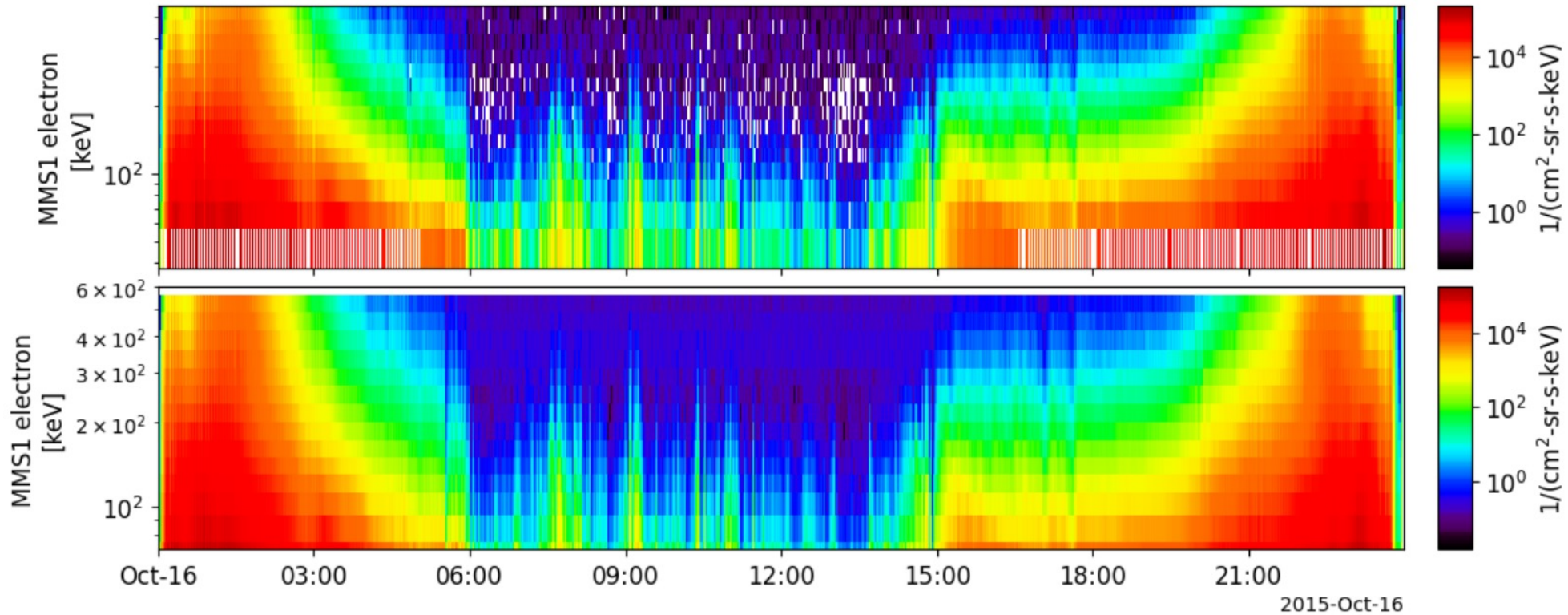
Example 3, MMS FEEPS data and spectrogram

Load and plot MMS FEEPS spectrogram data

```
pyspedas.mms.feeps(probe='1', trange=['2015-10-16', '2015-10-17'], datatype='electron')
tplot(['mms1_epd_feeps_srvy_l2_electron_intensity_omni',
       'mms1_epd_feeps_srvy_l2_electron_intensity_omni_spin'])
```

Executed at 2022-10-05 00:10:40 in 50.255ms

Example 3, PySPEDAS spectrogram plot



Analysis tools: coordinate transformations

Example 4: Coordinate transforms

Perform some coordinate transforms on previously loaded THEMIS FGM data

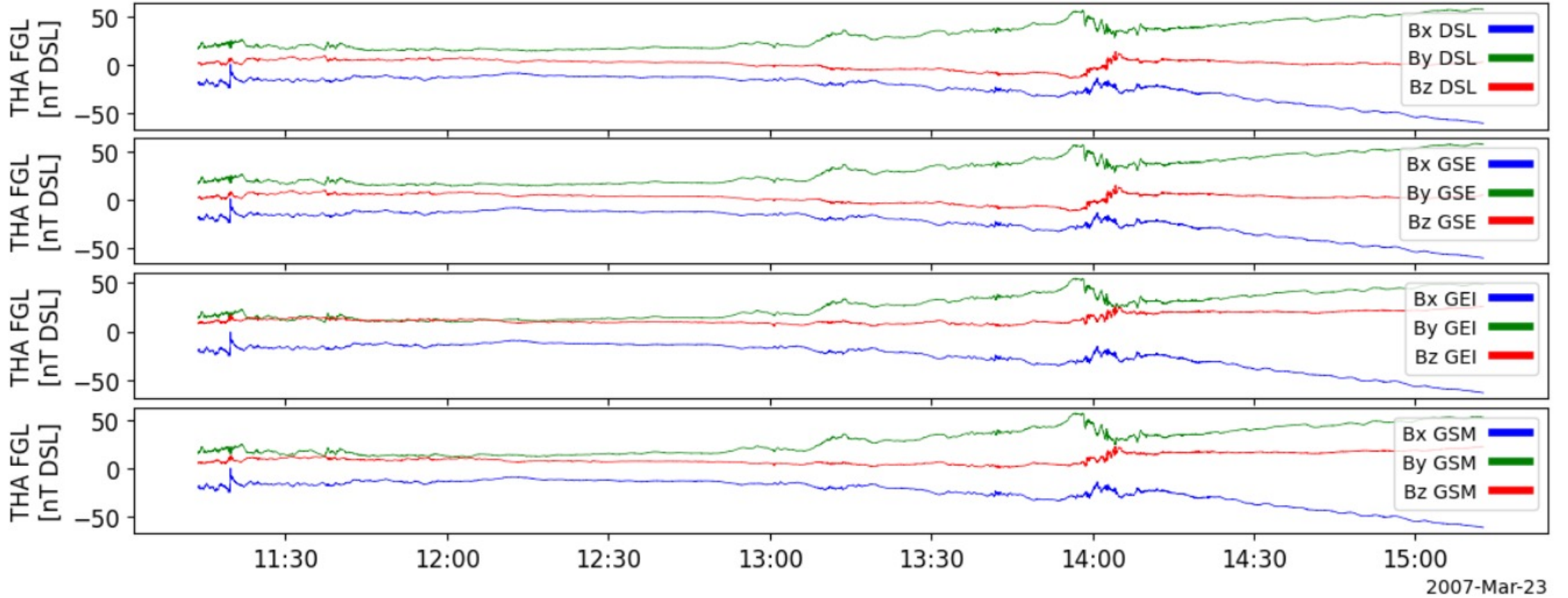
```
from pyspedas.themis.cotrans import dsl2gse

dsl2gse('tha_fgl_dsl', name_out='tha_fgl_gse')
pyspedas.cotrans('tha_fgl_gse', coord_out='GSI', name_out='tha_fgl_gei')
pyspedas.cotrans('tha_fgl_gse', coord_out='GSM', name_out='tha_fgl_gsm')
tplot(['tha_fgl_dsl', 'tha_fgl_gse', 'tha_fgl_gei', 'tha_fgl_gsm'])
```

Example 4: console output

```
05-Oct-23 00:12:23: tinterpol (linear) was applied to: tha_spinras_corrected_hires
05-Oct-23 00:12:23: tinterpol (linear) was applied to: tha_spindex_corrected_hires
05-Oct-23 00:12:24: Running transformation: subgei2gse
05-Oct-23 00:12:24: ['gse', 'gei']
05-Oct-23 00:12:24: Running transformation: subgse2gei
05-Oct-23 00:12:24: Output variable: tha_fgl_gei
05-Oct-23 00:12:24: ['gse', 'gsm']
05-Oct-23 00:12:24: Running transformation: subgse2gsm
05-Oct-23 00:12:24: Output variable: tha_fgl_gsm
```

Example 4: PySPEDAS plots



Other analysis tools

- Wavelet transforms
- Wave polarization analysis
- 2-D slices of 3-D particle distributions

Jupyter notebooks

- For PySPEDAS, we mainly use Jupyter notebooks for tutorial examples.
- Jupyter notebooks run in an interactive Python process, with code and results displayed in a browser window.
- Each code fragment lives in its own “cell” (but they all run in a common environment).
- There are also annotation cells, which can contain rich content – embedded images, styled text, live web links, etc. So they are much more versatile than inline code comments.

Running Jupyter notebooks

- You will probably need to install the jupyter package in the virtual environment you're using: "pip install jupyter"
- The command "jupyter notebook" will open a page in your browser. From there, you should be able to navigate to the notebook you're interested in, open it and start running.
- Another option, which doesn't require a Python installation at all, is to use "Google Colab". A notebook can be uploaded to a Google Drive page (with a few extra commands in the notebook to install the required packages). Opening the notebook will spin up a Python session in the cloud and let you run the notebook there, and display the results in your browser.

Jupyter notebook

localhost:8888/notebooks/advanced/MMS_basic_tail_science.ipynb#

UPDATE Read [the migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

Don't show anymore

jupyter MMS_basic_tail_science (unsaved changes)



Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3 (ipykernel)

Run Cell Code

MMS Basic Tail Science

This notebook is for basic tail science figures (from EVA, first plot); based on the IDL crib sheet: mms_basic_tail.pro

Generates a figure containing:

1. AE
2. FGM, srvy, GSM
3. FGM Bz
4. FGM magnitude
5. EDP, -log(scspot)
6. FPI N (ion/electron density)
7. FPI Vi (ion velocity, 3 components)
8. (ExB) -> x, y, z
9. J total
10. EIS protons
11. FPI ion spectra
12. HPCA O+
13. FEEPS intensity
14. FPI electron spectra
15. EDP X, Y, Z
16. EDP HFESP
17. DSP BPSD

```
In [ ]: ▶ import pyspedas
import numpy as np
from pytploit import tplot, options, get_data, store_data
```

16. EDP TPLST
17. DSP BPSD

```
In [ ]: ▶ import pyspedas
import numpy as np
from pytplo import tplot, options, get_data, store_data
```

Set some data options

```
In [ ]: ▶ trange = ['2015-10-16', '2015-10-17']
probe = '1'
```

Load the data

```
In [ ]: ▶ pyspedas.omni.data(trange=trange)
pyspedas.mms.fgm(trange=trange, probe=[1, 2, 3, 4], data_rate='srvy', level='l2', time_clip=True, get_fgm_ephemeris=True)
pyspedas.mms.mec(trange=trange, probe=probe, data_rate='srvy', level='l2', time_clip=True)
pyspedas.mms.fpi(trange=trange, probe=probe, data_rate='fast', level='l2', datatype=['des-moms', 'dis-moms'], time_clip=True)
pyspedas.mms.edp(trange=trange, probe=probe, datatype='scpot', level='l2', time_clip=True)
pyspedas.mms.edp(trange=trange, probe=probe, datatype='dce', data_rate='fast', time_clip=True)
pyspedas.mms.edp(trange=trange, probe=probe, datatype=['dce', 'hfesp'], data_rate='srvy', level='l2', time_clip=True)
pyspedas.mms.dsp(trange=trange, probe=probe, datatype='bpsd', data_rate='fast', level='l2', time_clip=True)
pyspedas.mms.hpca(trange=trange, probe=probe, datatype='ion', data_rate='srvy', level='l2', time_clip=True)
pyspedas.mms.eis(trange=trange, probe=probe, datatype='extof', time_clip=True)
pyspedas.mms.feeps(trange=trange, probe=probe, datatype='electron', time_clip=True)
```

Sum the HPCA spectra over the full field of view

Additional PySPEDAS example notebooks


















- On Github, we have several repositories containing quite a few Jupyter notebooks:
 - PyTplot and other tools: <https://github.com/spedas/pyspedas-examples>
 - MMS examples: <https://github.com/spedas/mms-examples>
 - THEMIS examples: <https://github.com/spedas/themis-examples>
- You can navigate to individual notebooks in the Github pages and download them – you don't need to use Git or install the whole repository.



master

pyspedas_examples / pyspedas_examples / notebooks /

↑ Top

 Cluster_CIS_data_from_CSA.ipynb	Adding example of loading/plotting Cluster CIS data from CSA	5 months ago
 Coordinate_transformations_with_OMNI_data.ipynb	adding cotrans notebook with OMNI data	last year
 Exploring_the_Heliosphere_with_Python.ipynb	several updates	5 months ago
 Introduction_to_PyTplot.ipynb	adding pytplot basics notebook	5 months ago
 PySPEDAS_PyTplot_for_Solar_Physicists.ipynb	Adding 2 new notebooks	last year
 PySPEDAS_PyTplot_timeseries_data.ipynb	minor update	5 months ago
 PySPEDAS_loading_data_from_HAPI_servers.ipynb	Adding notebooks created at the PyHC spring hackathon	last year
 PyTplot_annotations.ipynb	Adding examples showing greek letters, superscripts and subscripts	last month
 PyTplot_error_bars.ipynb	Adding notebooks created at the PyHC spring hackathon	last year
 PyTplot_highlight_intervals_and_vertical_bars.ipynb	Adding notebooks created at the PyHC spring hackathon	last year
 PyTplot_legend_options.ipynb	Adding notebooks created at the PyHC spring hackathon	last year
 PyTplot_markers_and_symbols.ipynb	fixing a minor issue	last year
 PyTplot_pseudo_variables.ipynb	Adding two new PyTplot notebooks	8 months ago
 PyTplot_range_options.ipynb	Adding two new PyTplot notebooks	8 months ago
 PyTplot_spectrogram_options.ipynb	Adding notebooks created at the PyHC spring hackathon	last year
 RBSP_RBSPICE_examples.ipynb	Adding RBSP RBSPICE examples	last month
 Swarm_data_in_PySPEDAS.ipynb	Updated with the latest version of PySPEDAS/PyTplot	last year

supervised Adding initial SST notebook

72ff295 · 2 years ago [History](#)

Name	Last commit message	Last commit date
..		
All Sky Imager data.ipynb	Adding All Sky Keograms	2 years ago
Electric_Field_Instrument_(EFI)_data.ipynb	Adding initial EFI notebook	2 years ago
Electrostatic_Analyzer_(ESA)_data.ipynb	Updating the ESA notebook	2 years ago
Fluxgate magnetometer (FGM) data.ipynb	adding some initial notebooks	2 years ago
Ground magnetometer data.ipynb	adding some initial notebooks	2 years ago
Search-coil magnetometer (SCM) data.ipynb	Update Search-coil magnetometer (SCM) data.ipynb	2 years ago
Solid_State_Telescope_(SST)_data.ipynb	Adding initial SST notebook	2 years ago
State data.ipynb	Create State data.ipynb	2 years ago



master

mms-examples / advanced /

Go to file

Add file



supervised Adding minimum variance analysis notebook

079694f · 3 months ago [History](#)

Name	Last commit message	Last commit date
..		
Combining_HPCA_and_EIS_energy_spectra.ipynb	Adding notebook showing how to combine EIS and HPCA energy spectra	last year
FPI_curlometer.ipynb	Adding new FPI curlometer notebook	last year
Generate spectrograms and moments with mms_part_getspec.ipynb	updated using pyspedas 1.3	2 years ago
Generate_2D_slices_of_FPI_and_HPCA_data.ipynb	Adding notebook showing 2D slices of FPI and HPCA distribution functi...	last year
MMS_LMN_coordinate_transformation.ipynb	updated using pyspedas 1.3	2 years ago
MMS_basic_dayside_science.ipynb	updated using pyspedas 1.3	2 years ago
MMS_basic_tail_science.ipynb	adding basic tail science notebook	last year
MMS_minimum_variance_analysis.ipynb	Adding minimum variance analysis notebook	3 months ago
MMS_neutral_sheet_models.ipynb	Adding notebook showing how to calculate distance to the neutral sheet	7 months ago
Plasma calculations with PlasmaPy.ipynb	updated using pyspedas 1.3	2 years ago
Poynting_flux_with_MMS_data.ipynb	Updates	last year
Wave_polarization_using_SCM_data.ipynb	updated using pyspedas 1.3	2 years ago

How to get help

- Documentation wiki: <http://spedas.org/wiki>
- Also: <https://pyspedas.readthedocs.io>
- Github issue tracker: <https://github.com/spedas/pyspedas/issues>
- Email questions and problem reports:
Themis_Science_Support@ssl.Berkeley.edu

Q & A, live demo(?)