SciQLop

A Tool Suite for Multi-Mission High-Resolution In-Situ Data Analysis in the Heliophysics Community

Alexis Jeandet[1], Nicolas Aunai[1], Benjamin Renard[3], Vincent Génot[2], Patrick Boettcher[4], Myriam Bouchemit[2], Ambre Ghisalberti[1], Bayane Michotte de Welle[1], Nicolas André[2]

[1]Laboratory Of Plasma Physics, CNRS, Palaiseau CEDEX, France [2]Institut de Recherche en Astrophysique et Planétologie, CNRS, CNES, UPS, Toulouse, France [3]Akkodis, Toulouse, France
[4]YAISE, Villeconin, France

Sharing is encouraged

With the **SCI**entific **Q**t application for **L**earning from **O**bservations of **P**lasmas (**SciQLop**), analyzing space physics data is made easier. The project aims to solve the technical challenges involved in retrieving and interpreting data from remote servers, which can be daunting for students or newcomers. Even analyzing data from a single instrument on a given mission can raise some technical difficulties such as finding where to get them, how to get them and sometimes how to read them. These challenges can compound when building complex machine learning pipelines involving multiple instruments and even multiple spacecraft missions. The SciQLop project removes these technical difficulties while maintaining high performance, allowing scientists to focus on their data analysis.

Speasy : Making Space Physics Data Access Easy

Speasy is an open-source Python package that streamlines the discovery and retrieval of space physics data from remote servers. By providing a user-friendly API, Speasy removes the technical complexities involved in finding and downloading data from sources such as **CDAWeb**, **SSCWeb**, **CSA**, and **AMDA**, making it easier for researchers and students to focus on data analysis.

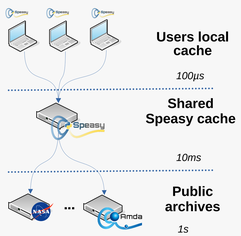
Key features:

- **Simple** API:

  With just one function, **get_data(what, [when])**, you can retrieve any combination of data products and time ranges. This makes it easy to access the specific data you need for your analysis, without having to navigate complex server structures or learn multiple APIs.

- **Efficient** caching mechanism

  Speasy delivers instant data access by reusing cached requests, no archive queries needed. Its cache auto-detects updates, removing outdated entries for fast, reliable results and lighter load on public archives.



- **Speasy** + **NumPy**/**SciPy**: Power Up Your Analysis:

  Run NumPy functions on Speasy variables and get Speasy variables back. Use SciPy to resample, filter, or interpolate—all with built-in wrappers.

- **No array conversion**: `np.sqrt(your_data)` keeps metadata intact.
- **Ready-to-use tools**: Smooth resampling, Butterworth filters, interpolation in one call.
- **Native workflows**: Results preserve time tags, units, and labels.

```python
def vect_to_mfa_delta(start, stop):
    B=spz.get_data(spz.inventories.tree.amda.Parameters.Cassini.MAG.orbit_saturn.cass_mag_krtphr.cass_b_krtphr,
                   start-timedelta(seconds=500),
                   stop+timedelta(seconds=500))
    B0=np.median(B, axis=0)
    delta_B=B-B0
    # REF = (ez, etheta, ephi)
    matrix_REF_to_MFA=np.empty((3,3))
    matrix_REF_to_MFA[0][:]=B0/np.linalg.norm(B0)
    matrix_REF_to_MFA[1]=np.cross(matrix_REF_to_MFA[0],[1,0,0])
    matrix_REF_to_MFA[2]=np.cross(matrix_REF_to_MFA[0],matrix_REF_to_MFA[1])

    B_mfa = np.einsum('kj,ij->ik', matrix_REF_to_MFA, delta_B)

    B_mfa = sosfiltfilt( sos=sos,var=B_mfa)

    return B_mfa


def mirror_mode_threshold(start_time: float, stop_time: float) -> SpeasyVariable or None:
    mms1_products = spz.inventories.data_tree.cda.MMS.MMS1
    products = [mms1_products.DIS.MMS1_FPI_FAST_L2_DIS_MOMS.mms1_dis_temppara_fast,
                mms1_products.DIS.MMS1_FPI_FAST_L2_DIS_MOMS.mms1_dis_tempperp_fast,
                mms1_products.FGM.MMS1_FGM_SRVY_L2.mms1_fgm_b_gse_srvy_l2,
                mms1_products.DIS.MMS1_FPI_FAST_L2_DIS_MOMS.mms1_dis_numberdensity_fast]

    tpara, tperp, b, n = spz.get_data(products, start_time, stop_time)

    anisotropy = tperp / tpara
    Pperp = tperp * n * 1e6
    b = interpolate(tperp, b)
    betaperp = Pperp * cst.mu_0 * cst.e * 2 / (b["Bt"] * 1e-9) ** 2
    mirror = betaperp * (anisotropy - 1)
    return mirror
```

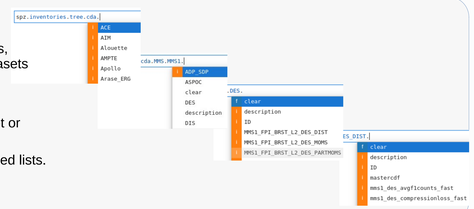
NumPy SciPy

- **Effortless Data Discovery**:

  Speasy dynamically catalogs all accessible remote servers and products, updating in real time. With Python runtime auto-completion, explore datasets instantly—no manual queries or digging through docs.

- **Live inventory**: Always see the latest available products.
- **Code-friendly discovery**: Tab-complete datasets directly in your script or notebook.
- **Zero setup**: Start browsing data sources immediately—no preconfigured lists.

Spend less time searching, more time analyzing.



SciQLop: A Fast and User-Friendly GUI for Space Physics Data Analysis

The SciQLop GUI app is a fast and extensible data analysis tool written in Python using PySide. Built with the aim of simplifying the analysis of in-situ space physics measurements, this app utilizes Speasy to access data from remote servers. In addition to its speed and simplicity, the app is also designed to be highly extensible, with an embedded IPython kernel and JupyterLab for easy integration with other Python tools and libraries.

Key features:

- **Effortless Exploration for Everyone**:

  SciQLop's intuitive interface balances power and simplicity, letting researchers at all coding levels focus on science—not software.

- **Drag-and-drop simplicity**: Build workflows, layer data, and customize plots with clicks, not code.
- **Lightning-fast interaction**: Seamless, lag-free zoom/pan keeps exploration fluid, even for gigabyte-scale datasets.

- **Build Virtual Products, Visualize Instantly**:

  Create custom **virtual products**—Python functions that take a time range and return time series or NumPy arrays. SciQLop automatically recomputes them on-the-fly as you zoom or pan, with live visualization and analysis in-app.

- **Define in minutes**: Combine data sources or algorithms into reusable functions with minimal code.
- **Dynamic updates**: Results refresh seamlessly as users interact with plots.

- **SciQLop** + **JupyterLab**: Code, Customize, Visualize:

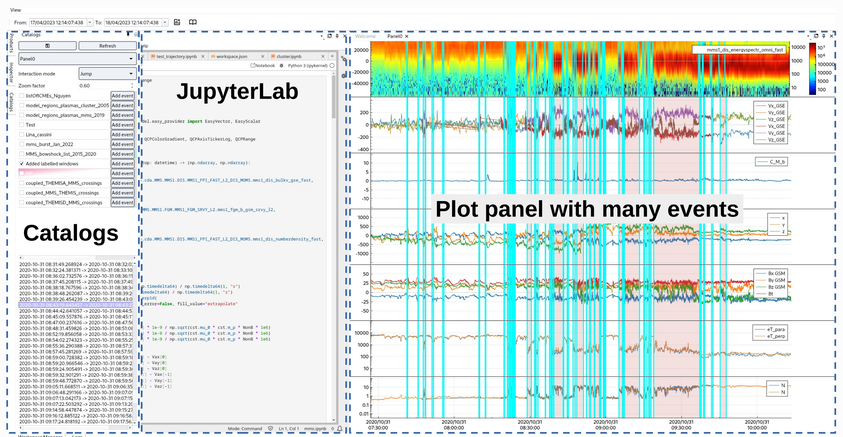
  Bridge interactive science and Python's power: Craft custom plot panels and products in Jupyter Notebooks using SciQLop's API.

- **Hybrid workflows**: Combine SciQLop's responsive UI (pan/zoom, real-time updates) with Python's rich libraries (NumPy, Pandas).
- **Rapid prototyping**: Build and test custom data pipelines or visualizations without leaving Jupyter.
- **Best of both worlds**: Deploy polished, interactive panels in SciQLop or export results to Python for deeper analysis.

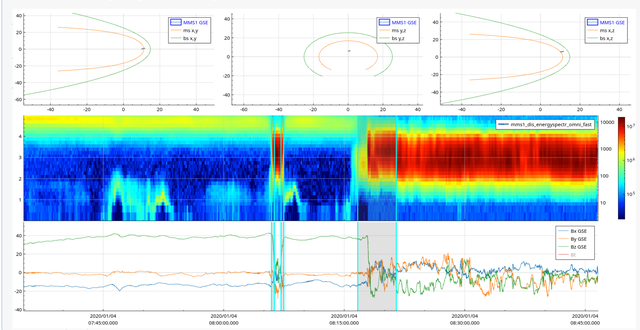
Planed features:

- **Catalogs coedition**: SciQLop will include catalogs online sharing and live co-edition.

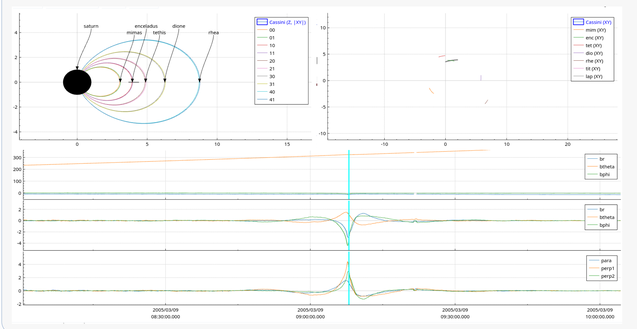
- "**Marketplace**": SciQLop will allow to browse, discover and fetch Notebooks or extensions from online community repositories.



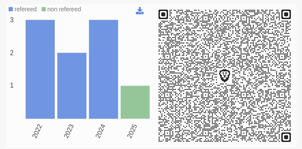
MMS Custom plot panel example



Cassini Custom plot panel example



Scientific publications citing Speasy



- **Speasy in Research**

  Speasy has been used by scientists and students for years to access and analyze space physics data. Publications citing Speasy showcase its practical role in simplifying workflows for studies and research projects.

  Reliable. Straightforward. Community-tested.