cavsiopy: a Python package to calculate and visualize spacecraft instrument orientation

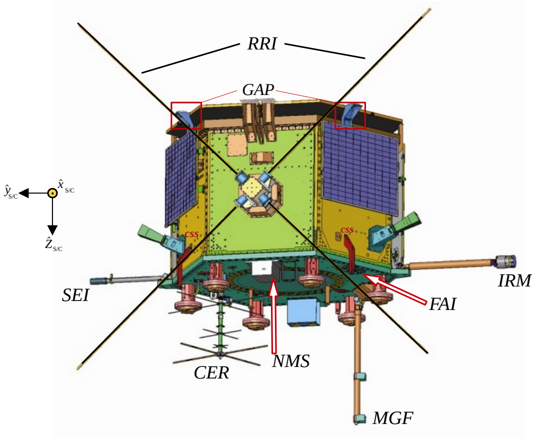
E.C.K. Eyiguler[1], W. Holley[2], A.D. Howarth[2], D.W. Danskin[1], C. J. Martin[1], K. Pandey[(1,3)], R.G. Gillies[2], A.W. Yau[2] and G.C. Hussey[1]

[1] Institute of Space and Atmospheric Studies (ISAS)
Department of Physics and Engineering Physics
University of Saskatchewan
Saskatoon, SK Canada
[2] Department of Physics and Astronomy, University of Calgary, Calgary, AB, Canada
[3] Center for Solar-Terrestrial Research, New Jersey Institute of Technology, Newark, NJ, USA

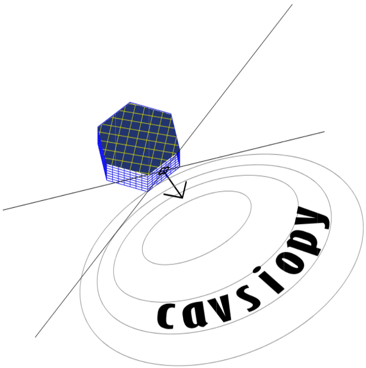
CEDAR 2024 Workshop - June 9-14, 2024
San Diego, CA, USA

The need for cavsiopy: e-POP on Swarm-Echo

- Swarm-E (2018)/ CASSIOPE
- perigee: 325km, apogee: 1500km
- 81° inclination
- elliptical, polar orbit
- RRI: 6m dipoles
- Purpose: Detection and investigation of radio waves
- Frequency: 10Hz - 18MHz
- Experiments with transmitters



cavsiopy



- RRI is 2D: may not always have optimum observation geometry
  - given instrument state vectors; computes attitude and orientation of any instrument onboard the spacecraft
  - supports transformations between reference frames(ORF, GEI/ICRF, ECEF, ITRF, NED, NEC)
  - calculates closeness to optimum observation direction
  - provides 2D, 3D visualizations of observation geometry

Installation and Modules

- pip install cavsiopy

  ## Dependencies

  h5py, geopy, astropy, pysofa2, numpy, matplotlib, and cartopy

- Downloadable from
  - Zenodo: `https://zenodo.org/records/8382041`
  - ICEBEAR Canada: `https://github.com/icebearcanada/cavsiopy`

Main Modules

- import data: ephemeris_importer
- compute attitude:attitude_analysis
  - via rotation matrices: use_rotation_matrices
  - via quaternions: use_quaternions
- visualization: attitude_plotter
- plot annotation tools: miscellaneous

Documentation:

`https://cavsiopy.readthedocs.io/en/latest/index.html`

ephemeris_importer

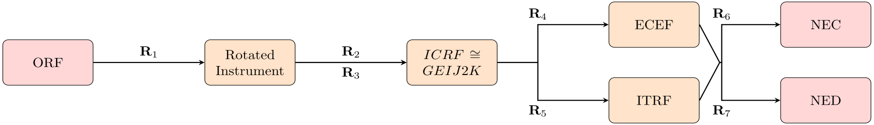
- cas_ephemeris: generic ephemeris files (*.txt)
- sp3_ephemeris: precise ephemeris (*.sp3)
- rri_ephemeris: RRI data file (*.h5)
- import_tle: TLE file from CelesTrak (*.txt)
- calculate_orbital_elements: from ICRF/GEIJ2K position and velocity
- compare_orbital: compared the calculated with TLE

```
dict_sp3 = ei.sp3_ephemeris(file_SP3, start_date, end_date)
```

All output is of type dictionary for simplicity

Transformations: use_rotation_matrices

I_{ORF}: instrument body vector in ORF –> $I_{NED} = R_7 \cdot R_5 \cdot R_3 \cdot R_1 \cdot I_{ORF}$

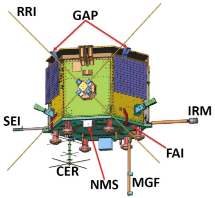


$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & sin\alpha \\ 0 & -sin\alpha & cos\alpha \end{bmatrix} \qquad R_y = \begin{bmatrix} cos\beta & 0 & -sin\beta \\ 0 & 1 & 0 \\ sin\beta & 0 & cos\beta \end{bmatrix} \qquad R_z = \begin{bmatrix} cos\gamma & sin\gamma & 0 \\ -sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R_{S/C \to ICRF} = \begin{bmatrix} \vec{x} \\ \vec{y} \\ \vec{z} \end{bmatrix} \qquad e = \begin{bmatrix} \vec{e}_{north} \\ \vec{e}_{east} \\ \vec{e}_{center} \end{bmatrix}$$

$$R_1 = R_z(\gamma) \cdot R_y(\beta) \cdot R_x(\alpha) \quad (1)$$
$$R_2 = R_z(\Omega) \cdot R_x(i) \cdot R_z(u) \cdot O \quad (2)$$
$$R_3 = R_{S/C \to ICRF}(\vec{r}_{icrf}, \vec{V}_{icrf}) \quad (3)$$
$$R_4 = R_z(\mu) \quad (4)$$
$$R_5 = R_M(t) \cdot R_s(t) \cdot N(t) \cdot P(t) \quad (5)$$
$$R_6 = [R_z(-\lambda) \cdot R_y(-\delta - \pi/2)]^T \quad (6)$$
$$R_7 = e \quad (7)$$

$$\vec{z} = (-\vec{r}_{icrf})/|\vec{r}_{icrf}| \quad (8)$$
$$\vec{y} = (-\vec{r}_{icrf} \times \vec{V}_{icrf})/|\vec{r}_{icrf} \times \vec{V}_{icrf}| = (\vec{z} \times \vec{V}_{icrf})/|\vec{z} \times \vec{V}_{icrf}| \quad (9)$$
$$\vec{x} = (\vec{y} \times \vec{z})/|\vec{y} \times \vec{z}| \quad (10)$$
$$\vec{e}_{center} = (-\vec{r}_{itrf})/|\vec{r}_{itrf}| \quad (11)$$
$$\vec{e}_{east} = (\vec{e}_{center} \times \begin{bmatrix} 0 & 0 & 1 \end{bmatrix})/|\vec{e}_{center} \times \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}| \quad (12)$$
$$\vec{e}_{north} = (\vec{e}_{east} \times \vec{e}_{center})/|\vec{e}_{east} \times \vec{e}_{center}| \quad (13)$$

Instrument Orientation-1: attitude_analysis




```
import cavsiopy.ephemeris_importer as ei
import cavsiopy.attitude_analysis as aa

# locate RRI data file
path_to_data = '/path/to/data/files/'
name_of_RRI_file= "RRI_20160418_222759_223156_lv1_12.0.0.h5"
file_RRI = path_to_data + name_of_RRI_file

# import RRI ephemeris
dict_rri = ei.rri_ephemeris(file_RRI)

# define RRI body vector
body_rri = [1, 0 , 0]

# rotate the RRI body vector in orbital frame
rbody = aa.rotate_inst(body_rri, dict_rri['roll'], dict_rri['pitch'],
                       dict_rri['yaw'])

# RRI body vector in North-East-Center
rri_nec = aa.find_instrument_attitude(rbody, dict_rri['GEIx'], dict_rri['GEIy'], dict_rri['GEIz'],
                                      dict_rri['GEIVx'], dict_rri['GEIVy'], dict_rri['GEIVz'],
                                      dict_rri['GEOx'], dict_rri['GEOy'], dict_rri['GEOz'],
                                      dict_rri['time_array'], dict_rri['time_array'][0],
                                      dict_rri['time_array'][-1], dict_rri['Lat'], dict_rri['Lon'],
                                      method1 ='ephemeris', frame2 = 'itrf', frame3 = 'nec')
```

Instrument Orientation-2: attitude_analysis

Figure: $IRM_{body} = [0, -1, 0]$

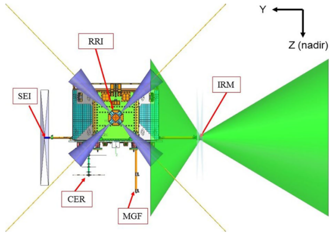
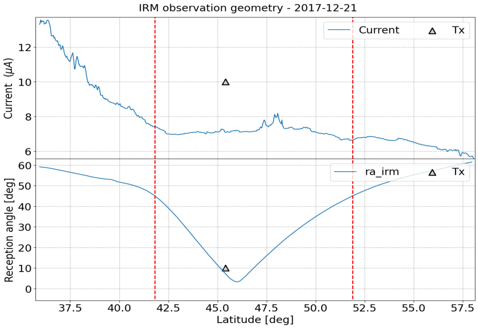


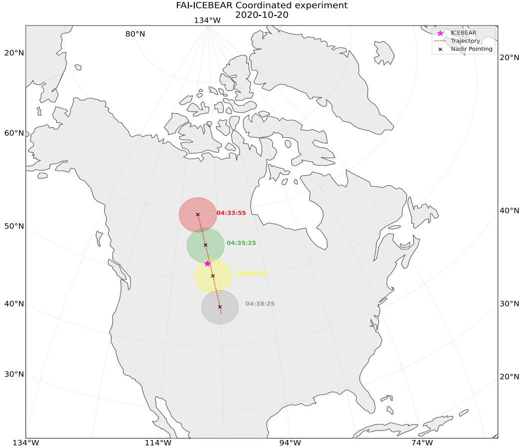
Fig. 1a Schematic view of IRM sensor placement, orientation, and field-of-view (FOV) in the spacecraft X–Z plane, showing the 360° FOV in azimuth (in the sensor aperture plane), ±2° elevation angle of acceptance (out of the aperture plane; between the *light blue triangles*), and the sampled range of clear FOV in elevation angle (−30° to +45°; between the green triangles). The +Z axis points toward nadir; the +X axis points toward ram. SEI = suprathermal electron imager, MGF = magnetic field instrument, RRI = radio receiver instrument, CER = coherent electromagnetic radio tomography instrument on e-POP



For finding the optimum observation intervals

1) find IRM attitude in GEIJ2K/ICRF
2) calculate the angle between IRM state vector and S/C ram direction (*Y-Z plane*).

Camera Field-of-View: attitude_plotter



– Works strictly for nadir experiments –

```
# instrument properties
instrument_name = 'FAI'
fov_deg = 26 # degrees

# target location and name
pLat = 58 # degrees
pLon = -106 # degrees
pAlt = 0.494 # km
target_name = 'ICEBEAR'

# call the fov_plotter and plot
extent = [Lonmin, Lonmax, Latmin, Latmax]
fig_fov, ax_fov = ap.fov_plotter(extent,
                                 time_array,
                                 Lon, Lat, Alt,
                                 fov_deg,
                                 pLon, pLat, step = 90,
                                 instrument_name,
                                 target_name)
```

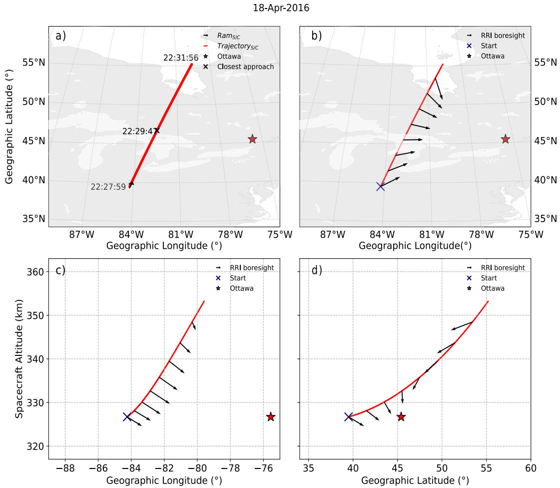
Visualization-2D: attitude_plotter

a) trajectory_plotter_2d_map

b) attitude_2d_latlon

c) attitude_altitude_plots

d) attitude_altitude_plots



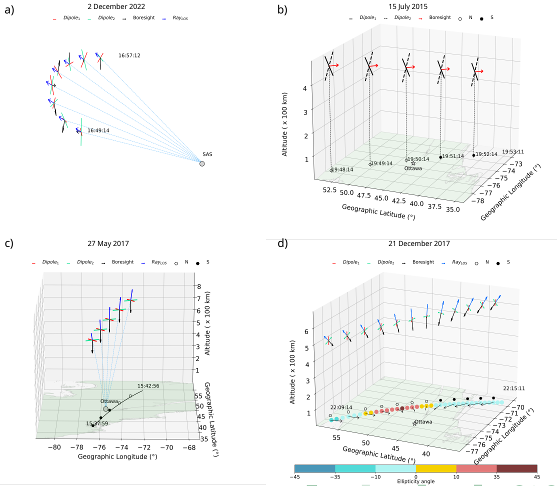
Visualization-3D: attitude_plotter

a) for geometry: display_observation_geometry

b) RRI-ram: attitude_3d_connect_to_subpoint

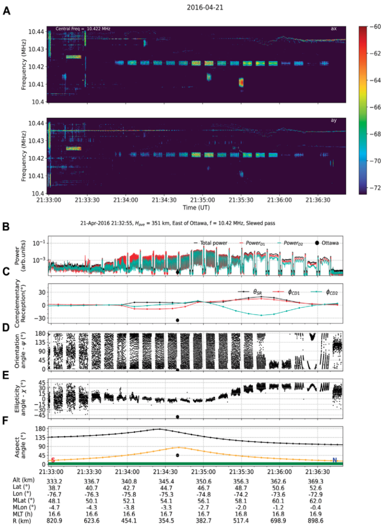
c) RRI-nadir: attitude_3d_connect_to_target

d) slew: attitude_3d_ground_quiver



Extra: Auxiliary

- Additional dependencies on pyIGRF and spacepy.
- Plot spectrogram
- Inspect and plot data validity
- Attitude analysis based on quaternions
- Radio wave polarization computation
- Calculation of magnetic aspect angle using straight line propagation assumption



Thank you!

- Developed cavsiopy for CASSIOPE/Swarm-E attitude analysis
- Data import, attitude estimation, straight-line propagation, distance calculations and visualization tools
- Strive to make the package generic and easy-to-use for other satellite missions

Details in GitHub, readthedocs, and Eyiguler et al. (2023) *Frontiers in Astronomy and Space Sciences*.

University of Saskatchewan
University of Calgary
Canadian Space Agency Agence spatiale canadienne
NSERC CRSNG
esa