

hxform and Coordinate Transform Standards

Bob Weigel
George Mason University

DASH Meeting, Session 6
Interoperability between Python Packages
10/10/2023

Outline

1. [hxform](#)

- Thin wrapper for coordinate transforms with back-ends of SpacePy (Native Python and IRBEM Fortran versions), [Tsyganenko's Geopack-08 library](#) (Fortran), [cxform](#) (c); Spicepy and SunPy support may be added.

2. Space Time Coordinate Transform (STCT) Standards Working Group

- See [Draft Poster](#) and [Meeting Notes](#)
- Motivation: Experience with HAPI and hxform

3. Answers to Rebecca's Questions

1. hxform

- Thin wrapper for coordinate transforms with back-ends of SpacePy (Native Python and IRBEM Fortran versions), [Tsyganenko's Geopack-08 library](#) (Fortran), [cxform](#) (c); Spiceypy and SunPy support may be added.
- Motivation: Not one lib worked as-needed for all projects due to speed, compilation issues, up-to-date IGRF, needed transforms, bugs, etc.
- Has in-development code for intercomparing results and demos for non-wrapped packages such as SpiceyPy and SunPy.
- Not quite ready for sharing; developed for use by my students but not the general community (no pip install, docs need work, some poor implementation choices).

2. STCT Standards Working Group

NASA-sponsored project; started this year. Involves

- the development of a comprehensive standard for acronyms and definitions

and, at some level,

- the implementation of comprehensive software, services, and unit tests for coordinate transforms; and
- understanding the uncertainty of transforms due to implementation choices.

3. Answers

1. Why did your package choose the current method for coordinate representation and conversion?

Decision was based on 1. Ease to get working, 2. What worked for my given application.

2. What in Astropy coordinates is missing or incompatible with heliophysics coordinate systems & transformations in your work?

Does not support many common heliophysics transforms out-of-the box. Much time to learn their interface and conventions. Is it fast for many transforms? Would prefer to not need all of AstroPy when I only need transforms.

3. Answers

3. If AstroPy is enough for your application, what changes need to be made in your software to adopt Astropy?

I am not sure that AstroPy is the solution. SpiceyPy/SunPy/SPICE may be a better option. I could use SPICE kernels on my non-Python projects. I think small focused packages have many advantages.

4. If AstroPy is not enough, what capabilities are missing?

Don't know enough to answer.

3. Answers

5. What do we need and what will that look like?

I don't have an answer.

An important issue to work out is the fact that AstroPy (and SpiceyPy/SPICE) and Heliophysics developers work in fundamentally different ways.

AstroPy/SunSpice/SpiceyPy are more formal and account for more details. Heliophysics packages tend to be much easier to use, but omit options for possibly important details.

Current preference is for SpiceyPy/SunSpice/SPICE approach because it does one thing only, all details are stored in files that can be used in many languages, and it is what Heliophysics missions often use.