Magnetic Mapping in the Inner Magnetosphere using Kamodo

Lutz Rastaetter[1], Rebecca Ringuette[1], Darren De Zeeuw[1], and Oliver Gerland[1]

[1]Affiliation not available

December 27, 2022

SH42E-2338: *Magnetic Mapping in the Inner Magnetosphere using Kamodo*

Lutz Rastaetter[1], Rebecca Ringuette[2,1], Darren DeZeeuw[3,1], and Oliver Gerland[4].

[1]*Community Coordinated Modeling Center, NASA Goddard Space Flight Center, Greenbelt, MD, USA*
[2]*ADNET Systems Inc., 6720B Rockledge Dr., Suite 504, Bethesda, MD, USA*
[3]*Catholic University of America, 620 Michigan Ave., N.E., Washington, DC, USA*
[4]*Ensemble Government Services LLC, Hyattsville, MD, USA*

New Abstract:

Many models require specialized access and interpolation schemes to effectively extract and interpolate their outputs. In particular, the Block-Adaptive Tree Solarwind Roe Upwind Scheme (BATSRUS) component of the Space Weather Modeling Framework (SWMF) requires Kamodo to take advantage of its block-based adaptive grid structure, and the Lyon-Fedder Mobarry magnetosphere model (or its successor GAMERA) needs a scheme that appreciates the distorted spherical arrangement of grid vertices on a non-orthogonal grid.

With the flythrough layer developed by Ringuette et al. (SH42E-2337), the underlying model readers have been adapted to use multiple time steps in a single Python session to perform 4-dimensional interpolations in time and space. Kamodo now utilizes lazy interpolation that loads data only when needed.

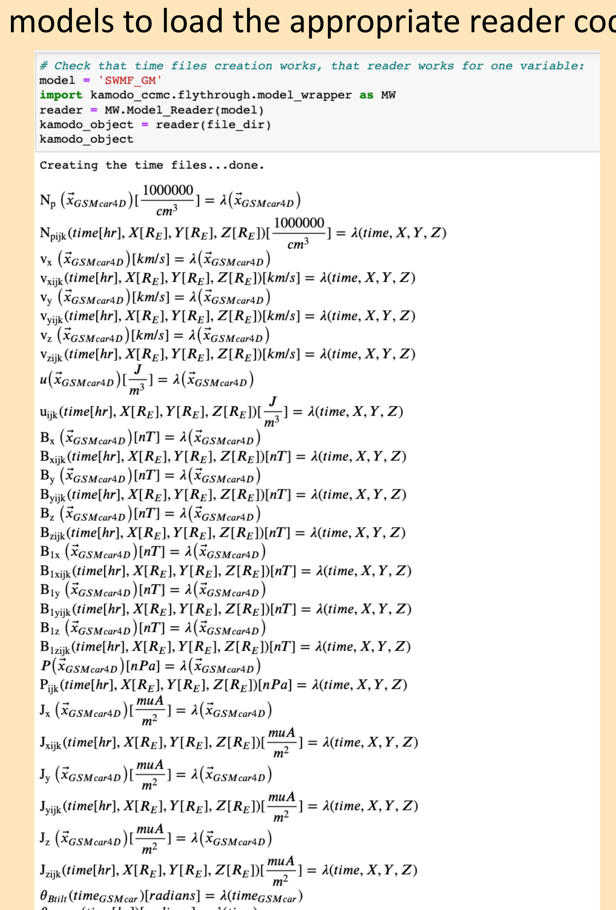
We present the successful integration of SWMF/BATSRUS magnetosphere access and interpolation into the new 4D Kamodo framework utilizing an external library of C code. Through function composition, Kamodo facilitates the calculation of derived quantities and the transformation of positions and vectors into different coordinate systems.

This work is a significant step towards performing field line tracing in Kamodo with SWMF magnetosphere outputs.

4D Model Access:

Functionalizing the Data

Kamodo has a generic Model object that takes the name of one of the supported models to load the appropriate reader code.



The Model object then functionalizes all available variables, 4D (time, space) or 1D (timeline) contained in a model output directory. At this time nothing beyond some metadata has been read.

4D Model Access (continued):

Functionalization for Specific Variables

```
kamodo_object = reader(file_dir, variables_requested=['B_x','B_y','B_z','theta_Btilt'])
kamodo_object
```

$\mathrm{B_x}\left(\vec{x}_{GSMcar4D}\right)[nT] = \lambda\left(\vec{x}_{GSMcar4D}\right)$
$\mathrm{B_{xijk}}(time[hr], X[R_E], Y[R_E], Z[R_E])[nT] = \lambda(time, X, Y, Z)$
$\mathrm{B_y}\left(\vec{x}_{GSMcar4D}\right)[nT] = \lambda\left(\vec{x}_{GSMcar4D}\right)$
$\mathrm{B_{yijk}}(time[hr], X[R_E], Y[R_E], Z[R_E])[nT] = \lambda(time, X, Y, Z)$
$\mathrm{B_z}\left(\vec{x}_{GSMcar4D}\right)[nT] = \lambda\left(\vec{x}_{GSMcar4D}\right)$
$\mathrm{B_{zijk}}(time[hr], X[R_E], Y[R_E], Z[R_E])[nT] = \lambda(time, X, Y, Z)$
$\theta_{Btilt}(time_{GSMcar})[radians] = \lambda(time_{GSMcar})$
$\theta_{Btiltijk}(time[hr])[radians] = \lambda(time)$

A list of requested output variables reduces the variables being functionalized

θ, the dipole axis tilt angle in GSM, is a single value for each time step. Kamodo creates a 1D time function.

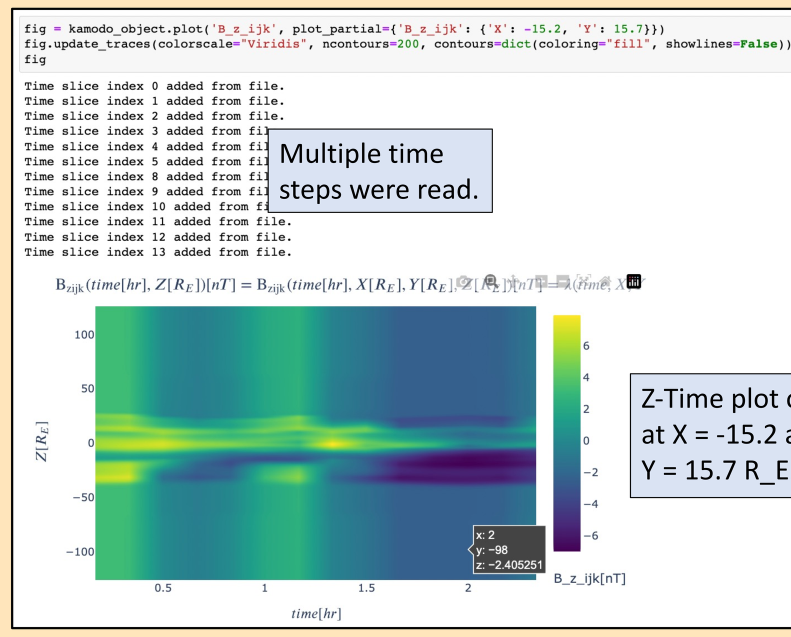
Test of Interpolation

Time = 1.2 hours (1:12 UT) is between 1:10 UT (index=6) and 1:20 UT (index=7), the nearest output times in the sample directory. The time is measured from midnight of the first day of the dataset. Data are read only as needed.
The gridded interpolator ($\mathrm{B_{zijk}}$) will return an array with default sets of positions in the unspecified ranks (here: formed by 448 unique positions in Y and 416 positions in Z throughout the BATSRUS grid).

```
# Confirm that interpolation works
print(kamodo_object.B_z([1.2, 10., 60., 50.]))
print(kamodo_object.B_z_ijk(time=1.2, X=10., Y=60., Z=50.))
print(kamodo_object.B_z_ijk(time=1.2, X=10).shape)
print(kamodo_object.B_z([[1.2,10., 60., 50.],[1.2,11., 61., 51.]]))

Time slice index 6 added from file.
Time slice index 7 added from file.
[0.2727453]
0.2727453038192415
(448, 416)
[0.2727453  0.27225559]

# Confirm that interpolation works (two differtnt time slices)
print(kamodo_object.B_z_ijk(time=0.8, X=10., Y=60., Z=50.))
print(kamodo_object.B_z([0.8, 10., 60., 50.]))
print(kamodo_object.B_z_ijk(time=0.8, X=10).shape)

Time slice index 3 added from file.
Time slice index 4 added from file.
1.318892373988329
[1.31889237]
(448, 416)
```

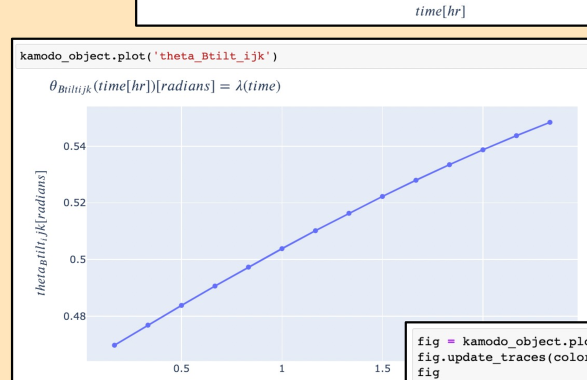
Visualizations

```
fig = kamodo_object.plot('B_z_ijk', plot_partial={'B_z_ijk': {'X': -15.2, 'Y': 15.7}})
fig.update_traces(colorscale="Viridis", ncontours=200, contours=dict(coloring="fill", showlines=False))
fig
```

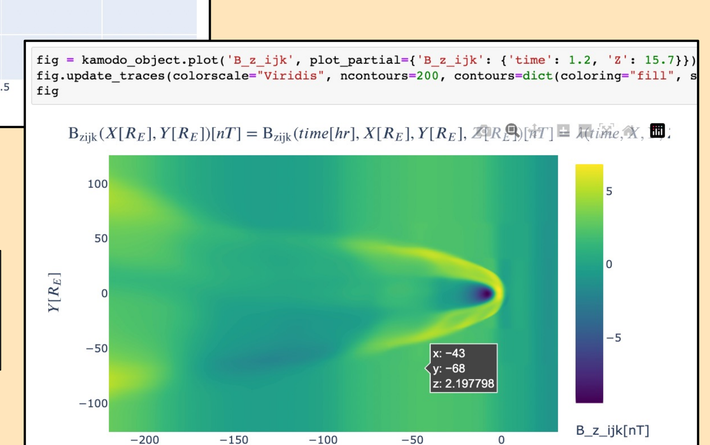


Multiple time steps were read.

Z-Time plot of $\mathrm{B_z}$ at X = -15.2 and Y = 15.7 R_E.



Time plot of θ



X-Y plot of $\mathrm{B_z}$ at Time = 1.2 hrs and Z = 15.7 R_E

Function Composition:

```
# Import satellite flythrough coordinate conversion code for more info on the coordinate systems.
from kamodo_ccmc.flythrough.utils import ConvertCoord
help(ConvertCoord)

    SpacePy
    Abbrev.   Full Name                        Lon. range     vertical variable
    --------------------------------------------------------------------------
    GDZ    Geodetic (WGS 84)                   (-180, 180)    Altitude (km)
    GEO    Geographic                          (-180, 180)    Radius (R_E)
    GSM    Geocentric Solar Magnetospheric     (-180, 180)    Radius (R_E)
    GSE    Geocentric Solar Ecliptic           (-180, 180)    Radius (R_E)
    SM     Solar Magnetic                      (-180, 180)    Radius (R_E)
    GEI    Geocentric Equatorial Inertial      (-180, 180)    Radius (R_E)
          (also ECI = Earth-Centered Inertial)
    MAG    Geomagnetic                         (-180, 180)    Radius (R_E)
    SPH    Spherical                            (0, 360)      Radius (R_E)
    RLL    Radius, Latitude, Longitude         (-180, 180)    Radius (R_E)
```

Coordinate systems available to *Kamodo* via *SpacePy* (above) and *AstroPy* (not shown).
Coordinate type 'car' indicates cartesian and 'sph' spherical coordinates (valid ranges for type 'sph' are shown above).

```
def Bvec(self,xvec_GSM_car4D):
    from numpy import vstack
    bx = self.B_x(xvec_GSM_car4D)
    by = self.B_y(xvec_GSM_car4D)
    bz = self.B_z(xvec_GSM_car4D)
    return((vstack([bx, by, bz])).T)

kamodo_object['Bvec[nT]']=Bvec

kamodo_object['B_mag'] = '(B_x^2 + B_y^2 + B_z^2)**0.5'

kamodo_object
```

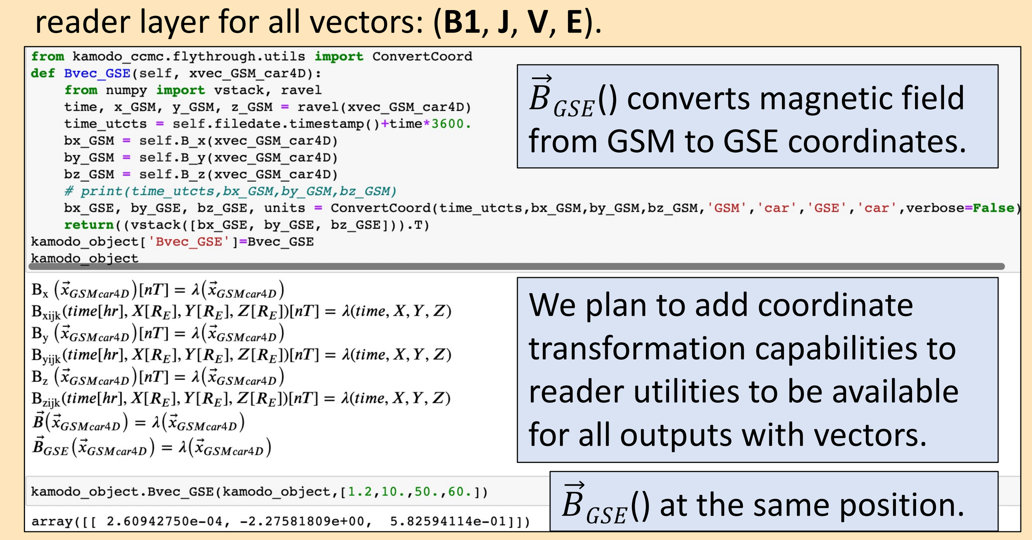
Define vector-valued function $\vec{B}$ and B magnitude variable and add to object.

$\mathrm{B_x}\left(\vec{x}_{GSMcar4D}\right)[nT] = \lambda\left(\vec{x}_{GSMcar4D}\right)$
$\mathrm{B_{xijk}}(time[hr], X[R_E], Y[R_E], Z[R_E])[nT] = \lambda(time, X, Y, Z)$
$\mathrm{B_y}\left(\vec{x}_{GSMcar4D}\right)[nT] = \lambda\left(\vec{x}_{GSMcar4D}\right)$
$\mathrm{B_{yijk}}(time[hr], X[R_E], Y[R_E], Z[R_E])[nT] = \lambda(time, X, Y, Z)$
$\mathrm{B_z}\left(\vec{x}_{GSMcar4D}\right)[nT] = \lambda\left(\vec{x}_{GSMcar4D}\right)$
$\mathrm{B_{zijk}}(time[hr], X[R_E], Y[R_E], Z[R_E])[nT] = \lambda(time, X, Y, Z)$
$\theta_{Btilt}(time_{GSMcar})[radians] = \lambda(time_{GSMcar})$
$\theta_{Btiltijk}(time[hr])[radians] = \lambda(time)$
$\vec{B}\left(\vec{x}_{GSMcar4D}\right)[nT] = \lambda\left(\vec{x}_{GSMcar4D}\right)$
$\mathrm{B_{mag}}\left(\vec{x}_{GSMcar4D}\right)[nT^{1.0}] = \sqrt{\mathrm{B_x}^2\left(\vec{x}_{GSMcar4D}\right) + \mathrm{B_y}^2\left(\vec{x}_{GSMcar4D}\right) + \mathrm{B_z}^2\left(\vec{x}_{GSMcar4D}\right)}$

```
kamodo_object.Bvec(kamodo_object,[1.2,10.,50.,60.])

Time slice index 6 added from file.
Time slice index 7 added from file.
Time slice index 6 added from file.
Time slice index 7 added from file.
Time slice index 6 added from file.
Time slice index 7 added from file.

array([[ 2.60942750e-04, -2.33178882e+00,  2.85525484e-01]])
```

Single interpolation triggers reading of data for 2 time steps for each of the 3 components.

Examples of computed variables:
Vector-valued magnetic field **B** , B magnitude (above) and **B** converted to GSE (below). These functions are to be added to reader layer for all vectors: **(B1, J, V, E).**



$\vec{B}_{GSE}()$ converts magnetic field from GSM to GSE coordinates.

We plan to add coordinate transformation capabilities to reader utilities to be available for all outputs with vectors.

$\vec{B}_{GSE}()$ at the same position.

Links to our GitHub Repositories:

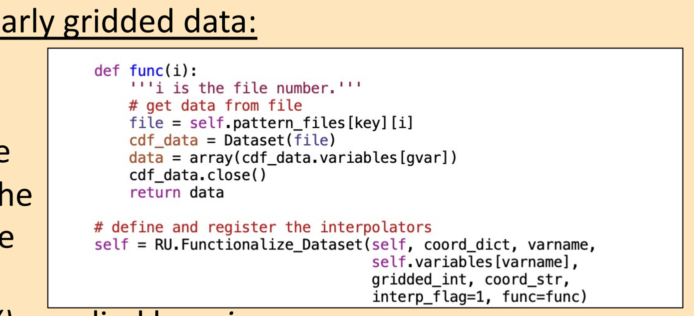
Get the CCMC's Kamodo!

Get the core Kamodo!

Interpolator Insertion:

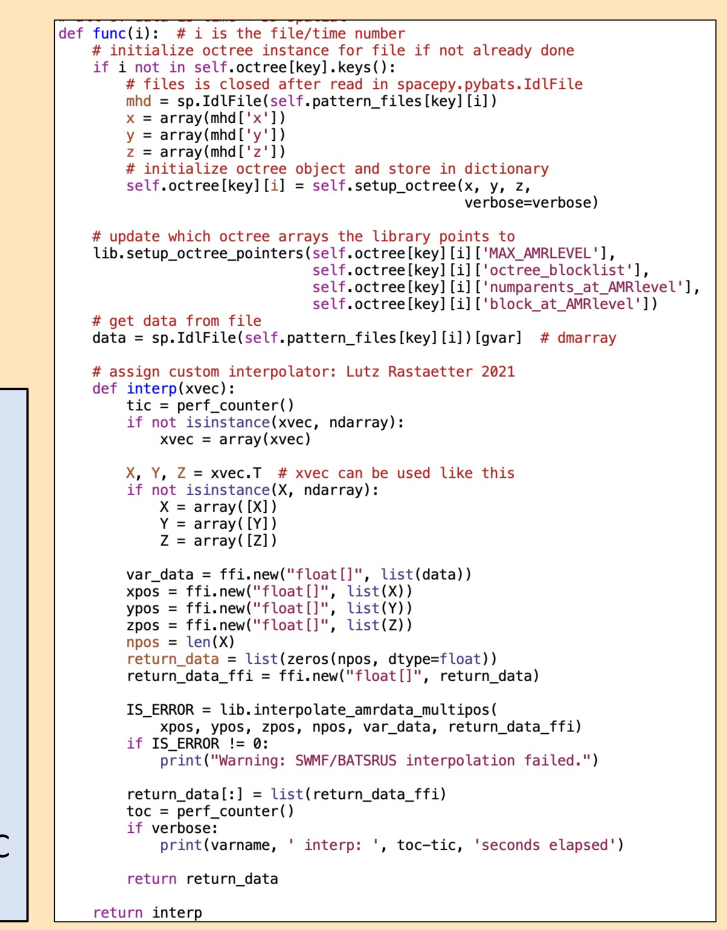
Standard case with regularly gridded data:

The function ***func()*** in ***register_variable()*** in the model reader retrieves the data for a variable for use by the *RegularGridInterpolator()* supplied by *scipy*.



Custom interpolator for BATSRUS:

func() now defines and returns an interpolator instance for each requested variable.
The entities *ffi* and *lib* are part of an external library, *_interpolate_amrdata* (imported earlier), that combine several functions compiled from C code:
setup_octree() in Kamodo (not shown) uses a function of the same name in the library *to* obtain the block size in X, Y, Z directions (≥ 4x4x4 cell, typically 6x6x6 or 8x8x8 at the CCMC) and returns an *octree* dictionary that is stored in the Kamodo object for each time step.
setup_octree_pointers() sets the proper positions in the octree dictionary for a selected time step.
interpolate_amrdata_multipos() performs the actual interpolation of the *data* using the octree grid.

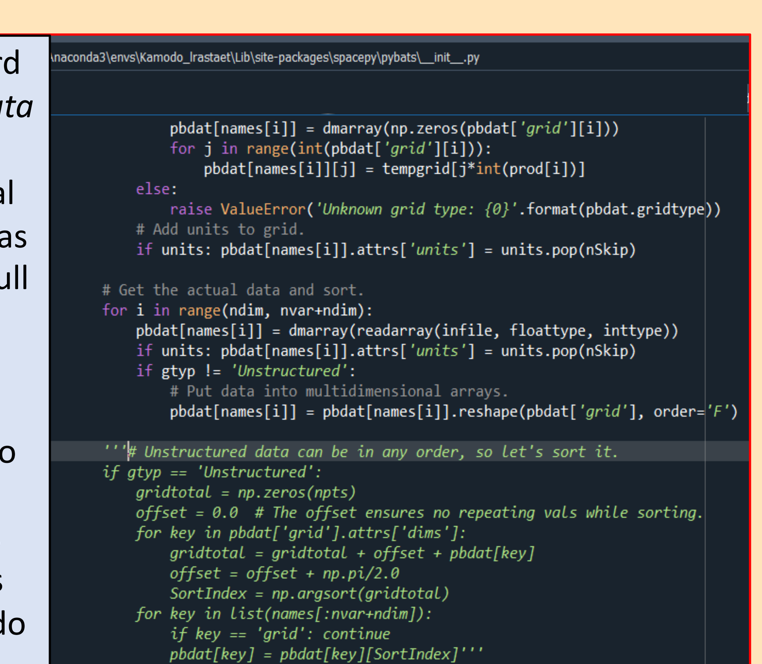


We plan to apply the same custom interpolator approach to support other models with complex grids, such as *GAMERA, LFM, GUMICS* and *SAMI3*. C code performing such interpolations already exists at CCMC and may be ported.

Kamodo uses *SpacePy* to read the SWMF model data.
The *_read_idl_bin()* method in *spacepy/pybats/__init__.py always* sorts "unstructured" data. To support Kamodo the responsible block of code was commented out (below in green starting at the highlighted line) in a modified local installation of *SpacePy* version 0.4.1.

Code to add a keyword *sort_unstructured_data* to *IdlFile()* while preserving the original behavior by default has been submitted via Pull Request.

Setting the keyword to *False* then turns off sorting and preserves block structure that is needed by the Kamodo reader.



Our Team:

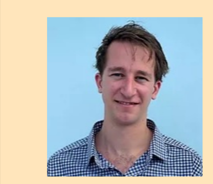
CCMC Staff:
https://ccmc.gsfc.nasa.gov/staff/

Lutz Rastaetter: Internal cross-language interfaces, specialized interpolators, CCMC-Vis, team management.

Rebecca Ringuette: Model interfaces, metadata, flythrough and other CCMC capabilities.

Darren De Zeeuw: GitHub management, visualization, metadata.

Ensemble Government Services partners:
https://www.ensembleconsultancy.com/government-services



Oliver Gerland and company: Core Kamodo capabilities, expert bug squashers.

Related Materials:

Posters and Papers:

- Enhanced Visualization using Kamodo for CCMC ITM Instant Runs SA32D-1694
- SH42E-2337: Science Workflows using Kamodo
- SM25C-2002: Kamodo's Satellite Constellation Mission Planning Tool
- Developing an Executable Paper With the Python in Heliophysics Community.
  DOI: 10.1002/essoar.10510006.1
  Accepted in *Frontiers in Astronomy and Space: Space Physics*.

References (DOI):

- Kamodo (core): 10.21105/joss.04053
- CCMC's Kamodo Flythrough: 10.3389/fspas.2022.1005977
- CCMC's Kamodo Model Readers: under review by *Advances in Space Research*.
- GITM: 10.1029/2016SW001465
- SWMF: 10.1029/2005JA011126
- GAMERA: 10.3847/1538-4365/ab3a4c
- LFM: 10.1016/j.jastp.2004.03.020
- GUMICS: 10.1016/j.jastp.2012.03.006
- SAMI3: 10.1029/2000JA000035
- SpacePy: https://spacepy.github.io/
- AstroPy: https://www.astropy.org/

Note:
If you find an issue with the Kamodo software, please report it on GitHub. For collaboration, please email **Rebecca.ringuette@nasa.gov**.
Geospace Model Support:
Lutz. Rastaetter@nasa.gov