Python Magnetic Vectors Documentation

Release 1.0.1

Russell Stoneback

Jan 04, 2022

CONTENTS

This documentation describes the Orthogonal Multi-pole Magnetic Basis Vectors (OMMBV) module, which contains routines to derive a vector basis from any magnetic field. These vectors may be used to organize electric field measurements as well as map those electric fields along magnetic field lines in a physically consistent manner.

CHAPTER

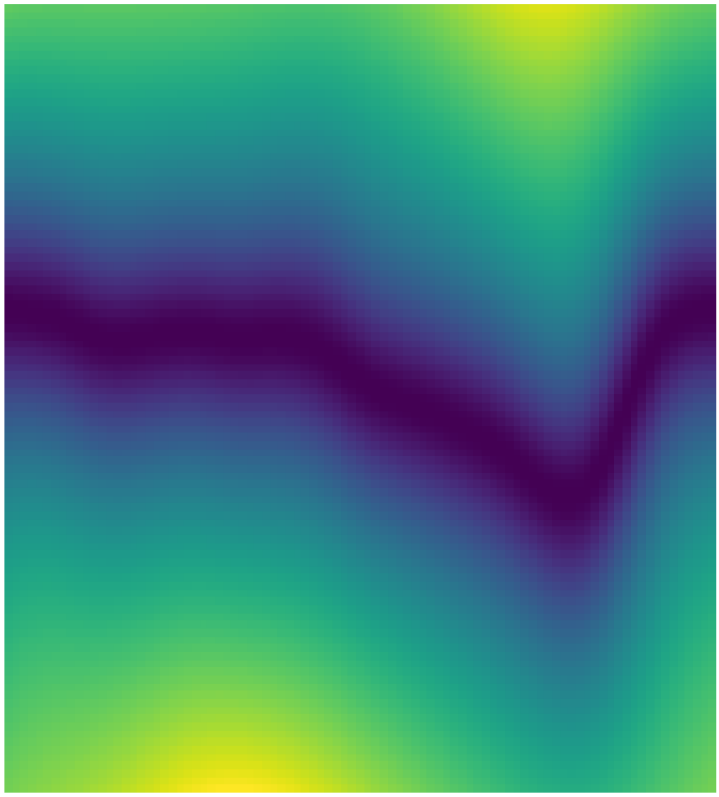
ONE

OVERVIEW

The derivation of an appropriate vector basis has been a long-term challenge in space science. The vector basis in current use is non-orthogonal and as such presents a number of challenges when attempting to apply it to space science studies. This package provides for the calculation of an orthogonal vector basis, consistent with the underlying physics, for general magnetic fields. Further, OMMBV calculates scaling factors suitable for translating an electric field at one location to any other location on the field line. OMMBV retains these properties for non-spherical planets, such as the Earth, as well as for magnetic fields that are more complicated than a pure dipole, such as the Earth's.

These features are fundamental for Space Science studies. Satellite measurements of plasma motions in-situ can now be accurately mapped along field lines for comparison to ground based equipment. Forcings from the neutral atmosphere (in say Earth's E-region Ionsophere) may now also be mapped to satellite altitudes to accurately characterize both the magnitude and direction of that forcing on plasma motion at satellite locations. Computer modelers may use OMMBV to map electric fields calculated within a single plane through the whole magnetosphere, significantly reducing computational resources.

OMMBV has been validated using a variety of test magnetic sources as well as via application to the Earth's magnetic field using the International Geomagnetic Reference Field (IGRF). Further, OMMBV includes two paths to determine the relevant vector basis. Both paths will only result in the same answers if the underlying system is truly orthogonal. The statistical performance of OMMBV between IGRF witha geodetic Earth and a spherical Earth with a pure dipole magnetic field is the same. OMMBV vectors are accurate up to 6-7 digits of precision at default calculation settings.



OMMBV

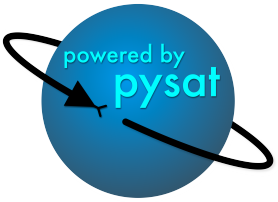
CHAPTER

TWO

INSTALLATION

The following instructions will allow you to install OMMBV.

2.1 Prerequisites



OMMBV uses common Python modules, as well as modules developed by and for the Space Physics community. pysat is an optional module, used by OMMBV to make it easy for satellite missions to add OMMBV results. This module officially supports Python 3.6+.

Common modules	Community modules
numpy	pysat
scipy	

2.2 Installation Options

1. Clone the git repository

```
git clone https://github.com/rstoneback/OMMBV.git
```

2. Install OMMBV: Change directories into the repository folder and run the setup.py file. There are a few ways you can do this:

   A. Install on the system (root privileges required):

   ```
   sudo python3 setup.py install
   ```

   B. Install at the user level:

   ```
   python3 setup.py install --user
   ```

   C. Install with the intent to develop locally:

```
python3 setup.py develop --user
```

CHAPTER

THREE

CITATION GUIDELINES

When publishing work that uses OMMBV, please cite the package and any package it depends on that plays an important role in your analysis. Specifying which version of OMMBV used will also improve the reproducibility of your presented results.

3.1 OMMBV

- Stoneback, R. A., J. H. Klenzing, G. Iyer. (2021). rstoneback/OMMBV:

v0.5.5 (v0.5.5). Zenodo. https://doi.org/10.5281/zenodo.4960425

```
@Misc{pysatModels,
      author = {Stoneback, R. A. and Klenzing, J. H., and Iyer, G..},
      title  = {rstoneback/OMMBV: v0.5.5 (v0.5.5)},
      year   = {2021},
      date   = {2021-06-15},
      url    = {https://github.com/rstoneback/OMMBV},
      }
```

CHAPTER

FOUR

API

Primary functions for calculating magnetic basis vectors.

OMMBV._core.**calculate_geomagnetic_basis**(*latitude*, *longitude*, *altitude*, *datetimes*)

Calculate local geomagnetic basis vectors and mapping scalars.

Parameters

- **latitude** (*array-like of floats (degrees) [-90., 90]*) – Latitude of location, degrees, WGS84
- **longitude** (*array-like of floats (degrees) [-180., 360.]*) – Longitude of location, degrees, WGS84
- **altitude** (*array-like of floats (km)*) – Altitude of location, height above surface, WGS84
- **datetimes** (*array-like of datetimes*) – Time to calculate vectors

Returns

zon_x (y,z): zonal unit vector along ECEF X, Y, and Z directions fa_x (y,z): field-aligned unit vector along ECEF X, Y, and Z directions mer_x (y,z): meridional unit vector along ECEF X, Y, and Z directions

d_zon_mag: D zonal vector magnitude d_fa_mag: D field-aligned vector magnitude d_mer_mag: D meridional vector magnitude

d_zon_x (y,z) : D zonal vector components. ECEF X, Y, and Z directions d_mer_x (y,z) : D meridional vector components. ECEF X, Y, and Z. d_fa_x (y,z) : D field aligned vector components. ECEF X, Y, and Z.

e_zon_mag: E zonal vector magnitude e_fa_mag: E field-aligned vector magnitude e_mer_mag: E meridional vector magnitude

e_zon_x (y,z) : E zonal vector components. ECEF X, Y, and Z directions. e_mer_x (y,z) : E meridional vector components. ECEF X, Y, and Z. e_fa_x (y,z) : E field aligned vector components. ECEF X, Y, and Z.

Return type dict

Note: Thin wrapper around *calculate_mag_drift_unit_vectors_ecef* set to default parameters and with more organization of the outputs.

OMMBV._core.**calculate_mag_drift_unit_vectors_ecef**(*latitude*, *longitude*, *altitude*, *datetimes*, *step_size=0.5*, *tol=0.0001*, *tol_zonal_apex=0.0001*, *max_loops=15*, *ecef_input=False*, *centered_diff=True*, *full_output=False*, *include_debug=False*, *scalar=None*, *edge_steps=None*, *dstep_size=0.5*, *max_steps=None*, *ref_height=None*, *steps=None*, *pole_tol=1e-05*, *location_info=<function apex_location_info>*, *mag_fcn=None*, *step_fcn=None*, *min_loops=3*, *apex_kwargs=None*)

Calculate local geomagnetic basis vectors and mapping scalars.

Zonal - Generally Eastward (+East); surface of constant apex height Field Aligned - Generally Northward (+North); points along geomagnetic field Meridional - Generally Vertical (+Up); gradient in apex height

The apex height is the geodetic height of the field line at its highest point. Unit vectors are expressed in Earth Centered Earth Fixed (ECEF) coordinates.

Parameters

- **latitude** (*array-like of floats (degrees) [-90., 90]*) – Latitude of location, degrees, WGS84
- **longitude** (*array-like of floats (degrees) [-180., 360.]*) – Longitude of location, degrees, WGS84
- **altitude** (*array-like of floats (km)*) – Altitude of location, height above surface, WGS84
- **datetimes** (*array-like of datetimes*) – Time to calculate vectors
- **step_size** (*float*) – Step size (km) to use when calculating changes in apex height (default=0.5)
- **tol** (*float*) – Tolerance goal for the magnitude of the change in unit vectors per loop (default=1.E-4)
- **tol_zonal_apex** (*float*) – Maximum allowed change in apex height along zonal direction (default=1.E-4)
- **max_loops** (*int*) – Maximum number of iterations (default=100)
- **ecef_input** (*bool*) – If True, inputs latitude, longitude, altitude are interpreted as x, y, and z in ECEF coordinates (km). (default=False)
- **full_output** (*bool*) – If True, return an additional dictionary with the E and D mapping vectors. (default=False)
- **include_debug** (*bool*) – If True, include stats about iterative process in optional dictionary. Requires full_output=True. (default=False)
- **centered_diff** (*bool*) – If True, a symmetric centered difference is used when calculating the change in apex height along the zonal direction, used within the zonal unit vector calculation. (default=True)
- **scalar** (*NoneType*) – Deprecated.
- **edge_steps** (*NoneType*) – Deprecated.
- **dstep_size** (*float*) – Step size (km) used when calculating the expansion of field line surfaces. Generally, this should be the same as step_size. (default=0.5)

- **max_steps** (*NoneType*) – Deprecated
- **ref_height** (*NoneType*) – Deprecated
- **steps** (*NoneType*) – Deprecated
- **location_info** (*function*) – Function used to determine a consistent relative position along a field line. Should not generally be modified. (default=apex_location_info)
- **pole_tol** (*float*) – When upward component of magnetic is within *pole_tol* of 1, the system will treat location as a pole and will not attempt to calculate unit vectors and scalars. (default=1.E-5)
- **mag_fcn** (*function*) – Function used to get information on local magnetic field. If None, uses default functions for IGRF. (default=None).
- **step_fcn** (*function*) – Function used to step along magnetic field. If None, uses default functions for IGRF. (default=None).
- **min_loops** (*int*) – Minimum number of iterations to attempt when calculating basis. (default=3)
- **apex_kwargs** (*dict or NoneType*) – If dict supplied, passed to *apex_location_info* as keyword arguments. (default=None)

Returns

- *zon_x, zon_y, zon_z, fa_x, fa_y, fa_z, mer_x, mer_y, mer_z,*
- *(optional dictionary)*
- *Optional output dictionary*
- ———————
- *Full Output Parameters*
- **d_zon_x (y,z)** (*D zonal vector components along ECEF X, Y, and Z directions*)
- **d_mer_x (y,z)** (*D meridional vector components along ECEF X, Y, and Z*)
- **d_fa_x (y,z)** (*D field aligned vector components along ECEF X, Y, and Z*)
- **e_zon_x (y,z)** (*E zonal vector components along ECEF X, Y, and Z directions*)
- **e_mer_x (y,z)** (*E meridional vector components along ECEF X, Y, and Z*)
- **e_fa_x (y,z)** (*E field aligned vector components along ECEF X, Y, and Z*)

Debug Parameters

diff_mer_apex : rate of change in apex height (km) along meridional vector diff_mer_vec : magnitude of vector change for last loop diff_zonal_apex : rate of change in apex height (km) along zonal vector diff_zonal_vec : magnitude of vector change for last loop loops : Number of loops vector_seed_type : Initial vector used for starting calculation (deprecated)

Note: The zonal and meridional vectors are calculated by using the observed apex-height gradient to rotate a pair of vectors orthogonal to eachother and the geomagnetic field such that one points along no change in apex height (zonal), the other along the max (meridional). The rotation angle theta is given by

Tan(theta) = apex_height_diff_zonal/apex_height_diff_meridional

The method terminates when successive updates to both the zonal and meridional unit vectors differ (magnitude of difference) by less than *tol*, and the change in apex_height from input location is less than *tol_zonal_apex*.

OMMBV._core.**cross_product**(*x1*, *y1*, *z1*, *x2*, *y2*, *z2*)
Cross product of two vectors, v1 x v2.

Deprecated since version 1.0.0: Function moved to *OMMBV.vector.cross_product*, this wrapper will be removed after v1.0.0.

Parameters

- **x1** (*float or array-like*) – X component of vector 1
- **y1** (*float or array-like*) – Y component of vector 1
- **z1** (*float or array-like*) – Z component of vector 1
- **x2** (*float or array-like*) – X component of vector 2
- **y2** (*float or array-like*) – Y component of vector 2
- **z2** (*float or array-like*) – Z component of vector 2

Returns Unit vector x,y,z components

Return type x, y, z

OMMBV._core.**ecef_to_enu_vector**(*x*, *y*, *z*, *glat*, *glong*)
Convert vector from ECEF X,Y,Z components to East, North, Up.

Deprecated since version 1.0.0: Function moved to *OMMBV.vector.ecef_to_enu*, this wrapper will be removed after v1.0.0.

Position of vector in geospace may be specified in either geocentric or geodetic coordinates, with corresponding expression of the vector using radial or ellipsoidal unit vectors.

Parameters

- **x** (*float or array-like*) – ECEF-X component of vector
- **y** (*float or array-like*) – ECEF-Y component of vector
- **z** (*float or array-like*) – ECEF-Z component of vector
- **glat** (*float or array_like*) – Geodetic or geocentric latitude (degrees)
- **glong** (*float or array_like*) – Geodetic or geocentric longitude (degrees)

Returns **east, north, up** – Vector components along east, north, and up directions

Return type np.array

OMMBV._core.**ecef_to_geocentric**(*x*, *y*, *z*, *ref_height=6371.0*)
Convert ECEF into geocentric coordinates.

Deprecated since version 1.0.0: Function moved to *OMMBV.trans.ecef_to_geocentric*, this wrapper will be removed after v1.0.0.

Parameters

- **x** (*float or array_like*) – ECEF-X in km
- **y** (*float or array_like*) – ECEF-Y in km
- **z** (*float or array_like*) – ECEF-Z in km
- **ref_height** (*float or array_like*) – Reference radius used for calculating height in km. (default=trans.earth_geo_radius)

Returns **latitude, longitude, altitude** – Locations in latitude (degrees), longitude (degrees), and altitude above *reference_height* in km.

Return type np.array

OMMBV._core.**ecef_to_geodetic**(**args*, ***kwargs*)
Convert ECEF into Geodetic WGS84 coordinates.

Deprecated since version 1.0.0: Function moved to *OMMBV.trans.geodetic_to_ecef*, this wrapper will be removed after v1.0.0.

OMMBV._core.**enu_to_ecef_vector**(*east*, *north*, *up*, *glat*, *glong*)
Convert vector from East, North, Up components to ECEF.

Deprecated since version 1.0.0: Function moved to *OMMBV.vector.enu_to_ecef*, this wrapper will be removed after v1.0.0.

Position of vector in geospace may be specified in either geocentric or geodetic coordinates, with corresponding expression of the vector using radial or ellipsoidal unit vectors.

Parameters

- **east** (*float or array-like*) – Eastward component of vector
- **north** (*float or array-like*) – Northward component of vector
- **up** (*float or array-like*) – Upward component of vector
- **glat** (*float or array_like*) – Geodetic or geocentric latitude (degrees)
- **glong** (*float or array_like*) – Geodetic or geocentric longitude (degrees)

Returns **x, y, z** – Vector components along ECEF x, y, and z directions

Return type np.array

OMMBV._core.**geocentric_to_ecef**(*latitude*, *longitude*, *altitude*)
Convert geocentric coordinates into ECEF.

Deprecated since version 1.0.0: Function moved to *OMMBV.trans.geocentric_to_ecef*, this wrapper will be removed after v1.0.0.

Parameters

- **latitude** (*float or array_like*) – Geocentric latitude (degrees)
- **longitude** (*float or array_like*) – Geocentric longitude (degrees)
- **altitude** (*float or array_like*) – Height (km) above presumed spherical Earth with radius 6371 km.

Returns **x, y, z** – x, y, z ECEF locations in km

Return type np.array

OMMBV._core.**geodetic_to_ecef**(*latitude*, *longitude*, *altitude*)
Convert WGS84 geodetic coordinates into ECEF.

Deprecated since version 1.0.0: Function moved to *OMMBV.trans.geodetic_to_ecef*, this wrapper will be removed after v1.0.0.

Parameters

- **latitude** (*float or array_like*) – Geodetic latitude (degrees)
- **longitude** (*float or array_like*) – Geodetic longitude (degrees)
- **altitude** (*float or array_like*) – Geodetic Height (km) above WGS84 reference ellipsoid.

Returns **x, y, z** – x, y, z ECEF locations in km

Return type np.array

OMMBV._core.**normalize_vector**(*x*, *y*, *z*)

Normalize vector to produce a unit vector.

Deprecated since version 1.0.0: Function moved to *OMMBV.vector.normalize*, this wrapper will be removed after v1.0.0.

Parameters

- **x** (*float or array-like*) – X component of vector
- **y** (*float or array-like*) – Y component of vector
- **z** (*float or array-like*) – Z component of vector

Returns Unit vector x,y,z components

Return type x, y, z

OMMBV._core.**project_ECEF_vector_onto_basis**(*x*, *y*, *z*, *xx*, *xy*, *xz*, *yx*, *yy*, *yz*, *zx*, *zy*, *zz*)

Project vector onto different basis.

Deprecated since version 1.0.0: Function moved to *OMMBV.vector.project_onto_basis*, this wrapper will be removed after v1.0.0.

Parameters

- **x** (*float or array-like*) – X component of vector
- **y** (*float or array-like*) – Y component of vector
- **z** (*float or array-like*) – Z component of vector
- **xx** (*float or array-like*) – X component of the x, y, z unit vector of new basis in original basis
- **yx** (*float or array-like*) – X component of the x, y, z unit vector of new basis in original basis
- **zx** (*float or array-like*) – X component of the x, y, z unit vector of new basis in original basis
- **xy** (*float or array-like*) – Y component of the x, y, z unit vector of new basis in original basis
- **yy** (*float or array-like*) – Y component of the x, y, z unit vector of new basis in original basis
- **zy** (*float or array-like*) – Y component of the x, y, z unit vector of new basis in original basis
- **xz** (*float or array-like*) – Z component of the x, y, z unit vector of new basis in original basis
- **yz** (*float or array-like*) – Z component of the x, y, z unit vector of new basis in original basis
- **zz** (*float or array-like*) – Z component of the x, y, z unit vector of new basis in original basis

Returns Vector projected onto new basis

Return type x, y, z

OMMBV._core.**python_ecef_to_geodetic**(*x*, *y*, *z*, *method='closed'*)
Convert ECEF into Geodetic WGS84 coordinates using Python code.

Deprecated since version 1.0.0: Function moved to *OMMBV.trans.geodetic_to_ecef*, this wrapper will be removed after v1.0.0.

Parameters

- **x** (*float or array_like*) – ECEF-X in km
- **y** (*float or array_like*) – ECEF-Y in km
- **z** (*float or array_like*) – ECEF-Z in km
- **method** (*str*) – Supports 'iterative' or 'closed' to select method of conversion. 'closed' for mathematical solution (page 96 section 2.2.1, http://www.epsg.org/Portals/0/373-07-2.pdf) or 'iterative' (http://www.oc.nps.edu/oc2902w/coord/coordcvt.pdf). (default = 'closed')

Returns **latitude, longitude, altitude** – Locations in latitude (degrees), longitude (degrees), and altitude above WGS84 (km)

Return type np.array

OMMBV._core.**scalars_for_mapping_ion_drifts**(*glats*, *glons*, *alts*, *dates*, *max_steps=None*, *e_field_scaling_only=None*, *edge_length=None*, *edge_steps=None*, ***kwargs*)
Translate ion drifts and electric fields to equator and footpoints.

Parameters

- **glats** (*list-like of floats (degrees)*) – Geodetic (WGS84) latitude
- **glons** (*list-like of floats (degrees)*) – Geodetic (WGS84) longitude
- **alts** (*list-like of floats (km)*) – Geodetic (WGS84) altitude, height above surface
- **dates** (*list-like of datetimes*) – Date and time for determination of scalars
- **e_field_scaling_only** (*Deprecated*) –
- **max_steps** (*Deprecated*) –
- **edge_length** (*Deprecated*) –
- **edge_steps** (*Deprecated*) –
- ****kwargs** (*Additional keywords*) – Passed to *calculate_mag_drift_unit_vectors_ecef*. *step_fcn*, if present, is also passed to *footpoint_location_info*.

Returns array-like of scalars for translating ion drifts. Keys are, 'north_zonal_drifts_scalar', 'north_mer_drifts_scalar', and similarly for southern locations. 'equator_mer_drifts_scalar' and 'equator_zonal_drifts_scalar' cover the mappings to the equator.

Return type dict

OMMBV._core.**step_along_mag_unit_vector**(*x*, *y*, *z*, *date*, *direction*, *num_steps=1*, *step_size=25.0*, *scalar=1*, ***kwargs*)
Move by following specified magnetic unit vector direction.

Moving along the field is effectively the same as a field line trace though extended movement along a field should use the specific *field_line_trace* method.

Parameters

- **x** (*ECEF-x (km)*) – Location to step from in ECEF (km).
- **y** (*ECEF-y (km)*) – Location to step from in ECEF (km).

- **z** (*ECEF-z (km)*) – Location to step from in ECEF (km).
- **date** (*list-like of datetimes*) – Date and time for magnetic field
- **direction** (*str*) – String identifier for which unit vector direction to move along. Supported inputs, 'meridional', 'zonal', 'aligned'
- **num_steps** (*int*) – Number of steps to take along unit vector direction (default=1)
- **step_size** (*float*) – Distance taken for each step (km) (default=25.)
- **scalar** (*int*) – Scalar modifier for step size distance. Input a -1 to move along negative unit vector direction. (default=1)
- ****kwargs** (*Additional keywords*) – Passed to *calculate_mag_drift_unit_vectors_ecef*.

Returns [x, y, z] of ECEF location after taking num_steps along direction, each step_size long.

Return type np.array

Note: *centered_diff=True* is passed along to *calculate_mag_drift_unit_vectors_ecef* when direction='meridional', while *centered_diff=False* is used for the 'zonal' direction. This ensures that when moving along the zonal direction there is a minimal change in apex height.

Provide support for adding OMMBV to NASA Ionospheric Connections Explorer.

OMMBV.satellite.**add_footpoint_and_equatorial_drifts**(*inst*, *equ_mer_scalar='equ_mer_drifts_scalar'*, *equ_zonal_scalar='equ_zon_drifts_scalar'*, *north_mer_scalar='north_footpoint_mer_drifts_scalar'*, *north_zon_scalar='north_footpoint_zon_drifts_scalar'*, *south_mer_scalar='south_footpoint_mer_drifts_scalar'*, *south_zon_scalar='south_footpoint_zon_drifts_scalar'*, *mer_drift='iv_mer'*, *zon_drift='iv_zon'*)

Translate ion velocities to those at footpoints and magnetic equator.

Note: Presumes scalar values for mapping ion velocities are already in the *inst*, labeled by 'north_footpoint_zon_drifts_scalar', 'north_footpoint_mer_drifts_scalar', 'equ_mer_drifts_scalar', 'equ_zon_drifts_scalar'.

Also presumes that ion motions in the geomagnetic system are present and labeled as 'iv_mer' and 'iv_zon' for meridional and zonal ion motions.

This naming scheme is used by the other pysat oriented routines in this package.

Parameters

- **inst** (*pysat.Instrument*) –
- **equ_mer_scalar** (*str*) – Label used to identify equatorial scalar for meridional ion drift. (default='equ_mer_drifts_scalar')
- **equ_zonal_scalar** (*str*) – Label used to identify equatorial scalar for zonal ion drift. (default='equ_zon_drifts_scalar')
- **north_mer_scalar** (*str*) – Label used to identify northern footpoint scalar for meridional ion drift. (default='north_footpoint_mer_drifts_scalar')
- **north_zon_scalar** (*str*) – Label used to identify northern footpoint scalar for zonal ion drift. (default='north_footpoint_zon_drifts_scalar')

- **south_mer_scalar** (*str*) – Label used to identify northern footpoint scalar for meridional ion drift. (default='south_footpoint_mer_drifts_scalar')
- **south_zon_scalar** (*str*) – Label used to identify southern footpoint scalar for zonal ion drift. (default='south_footpoint_zon_drifts_scalar')
- **mer_drift** (*str*) – Label used to identify meridional ion drifts within *inst*. (default='iv_mer')
- **zon_drift** (*str*) – Label used to identify zonal ion drifts within *inst*. (default='iv_zon')

Returns Modifies pysat.Instrument object in place. Drifts mapped to the magnetic equator are labeled 'equ_mer_drift' and 'equ_zon_drift'. Mappings to the northern and southern footpoints are labeled 'south_footpoint_mer_drift' and 'south_footpoint_zon_drift'. Similarly for the northern hemisphere.

Return type None

OMMBV.satellite.**add_mag_drift_unit_vectors**(*inst*, *lat_label='latitude'*, *long_label='longitude'*, *alt_label='altitude'*, ***kwargs*)

Add geomagnetic basis vectors expressed in S/C coordinates.

Internally, routine calls *add_mag_drift_unit_vectors_ecef*, which requires the orientation of the S/C basis vectors in ECEF using the following naming, 'sc_*hat_*' where *hat (*=x,y,z) is the S/C basis vector and* _ (*=x,y,z) is the ECEF direction.

Parameters

- **inst** (*pysat.Instrument object*) – Instrument object to be modified
- **lat_label** (*str*) – Label used within *inst* to identify latitude information. (default='latitude')
- **long_label** (*str*) – Label used within *inst* to identify longitude information. (default='longitude')
- **alt_label** (*str*) – Label used within *inst* to identify altitude information. (default='altitude')
- ****kwargs** – Passed along to *calculate_mag_drift_unit_vectors_ecef*

Returns Modifies instrument object in place. Adds 'unit_zon_*' where * = x,y,z 'unit_fa_*' and 'unit_mer_*' for zonal, field aligned, and meridional directions. Note that vector components are expressed in the S/C basis.

Return type None

OMMBV.satellite.**add_mag_drift_unit_vectors_ecef**(*inst*, *lat_label='latitude'*, *long_label='longitude'*, *alt_label='altitude'*, ***kwargs*)

Add geomagnetic basis vectors expressed in ECEF coordinates.

Parameters

- **inst** (*pysat.Instrument*) – Instrument object that will get unit vectors
- **lat_label** (*str*) – Label used within *inst* to identify latitude information. (default='latitude')
- **long_label** (*str*) – Label used within *inst* to identify longitude information. (default='longitude')
- **alt_label** (*str*) – Label used within *inst* to identify altitude information. (default='altitude')

- ****kwargs** – Passed along to calculate_mag_drift_unit_vectors_ecef

Returns

unit vectors are added to the passed Instrument object with a naming scheme:

'unit_zon_ecef_*' : unit zonal vector, component along ECEF-(X,Y,Z) 'unit_fa_ecef_*' : unit field-aligned vector 'unit_mer_ecef_*' : unit meridional vector

Return type None

OMMBV.satellite.**add_mag_drifts**(*inst*)

Add ion drifts expressed along geomagnetic basis vectors.

Note: Requires ion drifts under labels 'iv_*' where * = (x,y,z) along with unit vectors labels 'unit_zonal_*', 'unit_fa_*', and 'unit_mer_*', where the unit vectors are expressed in S/C coordinates. These vectors are calculated by *add_mag_drift_unit_vectors*.

Parameters **inst** (*pysat.Instrument*) – Instrument object will be modified to include new ion drift magnitudes

Returns Instrument object modified in place

Return type None

CHAPTER

FIVE

GUIDE FOR DEVELOPERS

5.1 Code of Conduct

5.2 Contributing

PYTHON MODULE INDEX

O

INDEX