SpacePy: Space Science Tools for Python

API Documentation

May 17, 2010

Contents

Contents 1

1 Package spacepy 2
- 1.1 Modules . . . 2

2 Module spacepy.constants 4
- 2.1 Functions . . . 4
- 2.2 Variables . . . 4

3 Module spacepy.empiricals 5
- 3.1 Functions . . . 5
- 3.2 Variables . . . 5

4 Module spacepy.omni 6
- 4.1 Functions . . . 6
- 4.2 Variables . . . 7

5 Module spacepy.onerapy 8
- 5.1 Functions . . . 8
- 5.2 Variables . . . 17

6 Module spacepy.onerapylib 18
- 6.1 Variables . . . 18

7 Module spacepy.poppy 19
- 7.1 Functions . . . 20
- 7.2 Variables . . . 20
- 7.3 Class PPro . . . 20
  - 7.3.1 Methods . . . 21
  - 7.3.2 Properties . . . 22

8 Module spacepy.radbelt 23
- 8.1 Functions . . . 23
- 8.2 Variables . . . 23
- 8.3 Class RBmodel . . . 23
  - 8.3.1 Methods . . . 24
  - 8.3.2 Properties . . . 26

9 Module spacepy.satdata 27
9.1 Functions . . . 27
9.2 Variables . . . 27
9.3 Class PSDdataClass . . . 27
9.3.1 Methods . . . 28
9.3.2 Properties . . . 28

10 Module spacepy.seapy 29
10.1 Functions . . . 29
10.2 Variables . . . 30
10.3 Class Sea . . . 30
10.3.1 Methods . . . 31
10.3.2 Properties . . . 32
10.4 Class Sea2d . . . 33
10.4.1 Methods . . . 33
10.4.2 Properties . . . 34

11 Module spacepy.setup 36
11.1 Functions . . . 36
11.2 Variables . . . 36

12 Module spacepy.spacetime 37
12.1 Functions . . . 37
12.2 Variables . . . 38
12.3 Class SpaCo . . . 39
12.3.1 Methods . . . 40
12.3.2 Properties . . . 46
12.4 Class TickDelta . . . 46
12.4.1 Methods . . . 47
12.4.2 Properties . . . 49
12.5 Class TickTock . . . 49
12.5.1 Methods . . . 50
12.5.2 Properties . . . 73

13 Module spacepy.testing 74
13.1 Functions . . . 74
13.2 Variables . . . 75

14 Module spacepy.toolbox 77
14.1 Functions . . . 77
14.2 Variables . . . 96

15 Module spacepy.utils 97
15.1 Functions . . . 97
15.2 Variables . . . 100

Index 101

1 Package spacepy

SpacePy: Space Science Tools for Python

SpacePy is a package of tools primarily aimed at the space science community. This __init__.py file sets the parameters for import statements.

If running the ipython shell, simply type '?' after any command for help. ipython also offers tab completion, so hitting tab after '<module name>.' will list all available functions, classes and variables.

Detailed HTML documentation is available locally in the spacepy/doc directory and can be launched by typing: >>> spacepy.help()

Version: 0.1dev

Author: The SpacePy Team

License: SpacePy: Space Science Tools for Python

1.1 Modules

- **constants** *(Section 2, p. 4)*
- **empiricals**: empirical models
  (Section 3, p. 5)
- **omni**: tools to read and process omni data
  (Section 4, p. 6)
- **onerapy**: module wrapper for onera_desp_lib Reference: D.
  (Section 5, p. 8)
- **onerapylib**: This module 'onerapylib' is auto-generated with f2py (version:2).
  (Section 6, p. 18)
- **poppy**: PoPPy – Point Processes in Python.
  (Section 7, p. 19)
- **radbelt**: Functions supporting radiation belt diffusion codes
  (Section 8, p. 23)
- **satdata**: Spacecraft data functions
  (Section 9, p. 27)
- **seapy**: SeaPy – Superposed Epoch in Python.
  (Section 10, p. 29)
- **setup** *(Section 11, p. 36)*

- **spacetime**: Implementation of TickTock class and SpaCo class functions
  (Section 12, p. 37)
- **testing**: test suite for spacepy
  (Section 13, p. 74)
- **toolbox**: Toolbox of various functions and generic utilities.
  (Section 14, p. 77)
- **utils**: Set of generic utilities.
  (Section 15, p. 97)

2 Module spacepy.constants

2.1 Functions

vIon(*temp*, *units*=`'eV'`, *Z*=`1`, *verbose*=`False`)

Calculate ion velocity [m/s] from temperature

Inputs: temp - Temperature in eV or Kelvin units (default is 'eV') - 'eV' or 'K' Z (default is 1) - ratio of ion mass to proton

Returns: v1 - ion velocity in metres per second

eIon(*vel*, *units*=`'both'`, *Z*=`1`, *verbose*=`False`)

Calculate ion energy [eV or K] from velocity

Inputs: vel - ion velocity in m/s units (for output; default is 'both') - 'eV' or 'K' Z (default is 1) - ratio of ion mass to proton

Returns: energy - tuple of energy in eV and K. Single values, in a tuple, are returned for non-default units

vAlfven(*B*, *n*, *Z*=`1`, *verbose*=`False`)

Calculate Alfven velocity [m/s] for input plasma paramters

Inputs: B - magnetic field strength [nT] n - plasma number density [cmˆ{-3}] Z (default is 1) - ratio of ion mass to proton

Returns: vA - Alfven velocity

2.2 Variables

Name	Description
c	**Value:** `299792458.0`
eVtoK	**Value:** `8.61771802827e-05`
k_b	**Value:** `1.38065812e-23`
mp	**Value:** `1.67362311e-27`
me	**Value:** `9.109389754e-31`
mu0	**Value:** `1.25663706144e-06`
q	**Value:** `1.6021773349e-19`
__package__	**Value:** `'spacepy'`

3 Module spacepy.empiricals

empirical models

Version: $Revision: 1.3 $, $Date: 2010/05/17 17:38:51 $

Authors: J. Koller, Los Alamos National Lab (jkoller@lanl.gov), Steve Morley (smorley@lanl.gov/morley_steve@hotmail.com

3.1 Functions

get_Lmax(*ticks*, *Lmax_model*)

calculate a simple empirical model for Lmax

get_plasma_pause(*ticks*, *Lpp_model*)

Plasmapause location model(s)

ShueMP(*P*, *Bz*)

Calculates the Shue et al. (1997) subsolar magnetopause radius

Ported from Drew Turner's (LASP) MatLab script

(section) Inputs:

SW ram pressure [nPa], IMF Bz (GSM) [nT]

(section) Output:

Magnetopause (sub-solar point) standoff distance [Re]

3.2 Variables

Name	Description
__log__	**Value:** `'\n$Log: empiricals.py,v $\nRevision 1.3 2010/05/17 17:3...`
__package__	**Value:** `'spacepy'`

4 Module spacepy.omni

tools to read and process omni data

Version: $Revision: 1.6 $, $Date: 2010/04/20 23:20:51 $

Author: Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

4.1 Functions

getomni(*ticktock*)

will load the pickled omni file, interpolate to the given ticktock time and return the omni values as dictionary with Kp, Dst, dens, velo, Pdyn, ByIMF, BzIMF, G1, G2, G3, etc. (see also http://www.dartmouth.edu/~rdenton/magpar/index.html and http://www.agu.org/pubs/crossref/2007/2006SW000296.shtml)

Note carefully: If the status variable is 2, the quantity you are using is fairly well determined. If it is 1, the value has some connection to measured values, but is not directly measured. These values are still better than just using an average value, but not as good as those with the status variable equal to 2. If the status variable is 0, the quantity is based on average quantities, and the values listed are no better than an average value. The lower the status variable, the less confident you should be in the value.

(section) Input:

- ticktock (TickTock class) : containing time information

(section) Returns:

- omnival (dictionary) : containing all omni values as a dictionary

(section) Example:

```
>>> tick = TickTock(['2002-02-02T12:00:00', '2002-02-02T12:10:00'], 'ISO')
>>> d = getomni(tick)
```

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 26-Jan-2010 (JK) V1.1: 11-Mar-2010: fixed bug in getomni; will now return the correct 6_status, 8_status (JK)

pickleomni(*fln*=`''`, *overwrite*=`True`, *data*=`None`)

read in fln='omni_intp.dat' file from Qin and Denton and save as pickle. The pickle will replace the current data file in the package, or provide data as strings

(section) Input:

- fln (string) : filename of the original ASCII file
- overwrite (optional boolean) : If true -> overwrite data file in package if false -> save in current working directory
- data (string) : container with data read-in from file

(section) Example:

```
>>> pickleomni('omnidata.dat')
```

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

getG123(*TAI*, *omnidata*)

get specific G1, G2, G3 for this TAI

4.2 Variables

Name	Description
__log__	**Value:** `'\n$Log: omni.py,v $\nRevision 1.6 2010/04/20 23:20:51 ...`
omnifln	**Value:** `'/home/smorley/projects/spacepy/data/omnidata.pbin'`
omnidata	**Value:** `{'6_status': array(['000000', '000000', '000000', ..., '0...`
__package__	**Value:** `'spacepy'`

5 Module spacepy.onerapy

module wrapper for onera_desp_lib Reference: D. Boscher1, S. Bourdarie1, P. O'Brien2, T. Guild2,(1 ONERA-DESP, Toulouse France; 2 Aerospace Corporation, Washington DC, USA), ONERA-DESP library V4.2, Toulouse-France, 2004-2008

Version: $Revision: 1.5 $, $Date: 2010/05/17 17:38:51 $

Author: Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

5.1 Functions

get_Bfield(*ticktock*, *spaco*, *extMag*='T01STORM', *options*=[1, 0, 0, 0, 0], *omnivals*=None)

call get_bfield in onera_desp lib and return a dictionary with the B-field vector and strenght.

(section) Input:

- ticktock (TickTock class) : containing time information
- spaco (Spaco class) : containing spatial information
- extMag (string) : optional; will choose the external magnetic field model possible values ['0', 'MEAD', 'T87SHORT', 'T87LONG', 'T89', 'OPQUIET', 'OPDYN', 'T96', 'OSTA', 'T01QUIET', 'T01STORM', 'T05', 'ALEX']
- options (optional list or array of integers length=5) : explained in Lstar
- omni values as dictionary (optional) : if not provided, will use lookup table
- (see Lstar documentation for further explanation)

(section) Returns:

- results (dictionary) : containing keys: Bvec, and Blocal

(section) Example:

```
>>> t = TickTock(['2002-02-02T12:00:00', '2002-02-02T12:10:00'], 'ISO')
>>> y = SpaCo([[3,0,0],[2,0,0]], 'GEO', 'car')
>>> op.get_Bfield(t,y)
{'Blocal': array([  945.99989101,  3381.71633205]),
       'Bvec': array([[  8.05001055e-01,  -1.54645026e+02,   9.33273841e+02],
       [  3.36352963e+02,  -5.33658140e+02,   3.32236076e+03]])}
```

(section) See Also:

Lstar, find_Bmirror, find_magequator

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 05-Mar-2010 (JK)

find_Bmirror(*ticktock*, *spaco*, *alpha*, *extMag*=`'T01STORM'`, *options*=`[1, 0, 0, 0, 0]`, *omnivals*=`None`)

call find_mirror_point from onera_desp library and return a dictionary with values for Blocal, Bmirr and the GEO (cartesian) coordinates of the mirror point

(section) Input:

- ticktock (TickTock class) : containing time information
- spaco (Spaco class) : containing spatial information
- alpha (list or ndarray) : containing the pitch angles
- extMag (string) : optional; will choose the external magnetic field model possible values ['0', 'MEAD', 'T87SHORT', 'T87LONG', 'T89', 'OPQUIET', 'OPDYN', 'T96', 'OSTA', 'T01QUIET', 'T01STORM', 'T05', 'ALEX']
- options (optional list or array of integers length=5) : explained in Lstar
- omni values as dictionary (optional) : if not provided, will use lookup table
- (see Lstar documentation for further explanation)

(section) Returns:

- results (dictionary) : containing keys: Blocal, Bmirr, GEOcar

(section) Example:

```
>>> t = TickTock(['2002-02-02T12:00:00', '2002-02-02T12:10:00'], 'ISO')
>>> y = SpaCo([[3,0,0],[2,0,0]], 'GEO', 'car')
>>> op.find_Bmirror(t,y,[90,80,60,10])
{'Blocal': array([ 0.,  0.]),
 'Bmirr': array([ 0.,  0.]),
 'GEOcar': SpaCo( [[ NaN  NaN  NaN]
 [ NaN  NaN  NaN]] ), dtype=GEO,car, units=['Re', 'Re', 'Re']}
```

(section) See Also:

Lstar, get_Bfield, find_magequator

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 05-Mar-2010 (JK)

find_magequator(*ticktock*, *spaco*, *extMag*=`'T01STORM'`, *options*=`[1, 0, 0, 0, 0]`, *omnivals*=`None`)

call find_magequator from onera_desp library and return a dictionary with values for Bmin and the GEO (cartesian) coordinates of the magnetic equator

(section) Input:

- ticktock (TickTock class) : containing time information
- spaco (Spaco class) : containing spatial information
- extMag (string) : optional; will choose the external magnetic field model possible values ['0', 'MEAD', 'T87SHORT', 'T87LONG', 'T89', 'OPQUIET', 'OPDYN', 'T96', 'OSTA', 'T01QUIET', 'T01STORM', 'T05', 'ALEX']
- options (optional list or array of integers length=5) : explained in Lstar
- omni values as dictionary (optional) : if not provided, will use lookup table
- (see Lstar documentation for further explanation)

(section) Returns:

- results (dictionary) : containing keys: Bmin, SpaCo instance with GEO coordinates of the magnetic equator

(section) Example:

```
>>> t = TickTock(['2002-02-02T12:00:00', '2002-02-02T12:10:00'], 'ISO')
>>> y = SpaCo([[3,0,0],[2,0,0]], 'GEO', 'car')
>>> op.find_magequator(t,y)
{'Bmin': array([  945.63652413,  3373.64496167]),
 'GEOcar': SpaCo( [[ 2.99938371  0.00534151 -0.03213603]
 [ 2.00298822 -0.0073077   0.04584859]] ), dtype=GEO,car, units=['Re', 'Re', 'Re']}
```

(section) See Also:

Lstar, get_Bfield, find_Bmirr

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 05-Mar-2010 (JK)

coord_trans(*spaco*, *returntype*, *returncarsph*)

thin layer to call coor_trans1 from onera_desp lib this will convert between systems GDZ, GEO, GSM, GSE, SM, GEI, MAG, SPH, RLL

(section) Input:

- spaco (SpaCo instance) : containing coordinate information, can contain n points
- returntype (str) : describing system as GDZ, GEO, GSM, GSE, SM, GEI, MAG, SPH, RLL
- returncarsph (str) : cartesian or spherical units 'car', 'sph'

(section) Returns:

- xout (ndarray) : values after transformation in (n,3) dimensions

(section) Example:

```
>>> coord = SpaCo([[3,0,0],[2,0,0]], 'GEO', 'car')
>>> coord.ticktock = TickTock(['2002-02-02T12:00:00', '2002-02-02T12:10:00'], 'ISO')
>>> coord_trans(coord, 'GSM', 'car')
array([[ 2.8639301 , -0.01848784,  0.89306361],
[ 1.9124434 ,  0.07209424,  0.58082929]])
```

(section) See Also:

sph2car, car2sph

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 05-Mar-2010 (JK)

car2sph(*CARin*)

coordinate transformation from cartesian to spherical

(section) Input:

- CARin (list or ndarray) : coordinate points in (n,3) shape with n coordinate points in units of [Re, Re, Re] = [x,y,z]

(section) Returns:

- results (ndarray) : values after conversion to spherical coordinates in radius, latitude, longitude in units of [Re, deg, deg]

(section) Example:

```
>>> sph2car([1,45,0])
array([ 0.70710678,  0.        ,  0.70710678])
```

(section) See Also:

sph2car

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 05-Mar-2010 (JK)

sph2car(*SPHin*)

coordinate transformation from spherical to cartesian

(section) Input:

- SPHin (list or ndarray) : coordinate points in (n,3) shape with n coordinate points in units of [Re, deg, deg] = [r, latitude, longitude]

(section) Returns:

- results (ndarray) : values after conversion to cartesian coordinates x,y,z

(section) Example:

```
>>> sph2car([1,45,45])
array([ 0.5       ,  0.5       ,  0.70710678])
```

(section) See Also:

car2sph

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 05-Mar-2010 (JK)

get_sysaxes(*dtype*, *carsph*)

will return the sysaxes according to the onera_desp library

(section) Input:

- dtype (str) : coordinate system, possible values: GDZ, GEO, GSM, GSE, SM, GEI, MAG, SPH, RLL
- carsph (str) : cartesian or spherical, possible values: 'sph', 'car'

(section) Returns:

- sysaxes (int) : value after oner_desp library from 0-8 (or None if not available)

(section) Example:

```
>>> get_sysaxes('GSM', 'car')
2
```

(section) See Also:

get_dtype

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov Steve Morley, Los Alamos National Lab, smorley@lanl.gov

(section) Version:

V1: 05-Mar-2010 (JK) V2: 10-May-2010 (SM)

get_dtype(*sysaxes*)

will return the coordinate system type as string

(section) Input:

- sysaxes (int) : number according to the onera_desp_lib, possible values: 0-8

(section) Returns:

- dtype (str) : coordinate system GDZ, GEO, GSM, GSE, SM, GEI, MAG, SPH, RLL
- carsph (str) : cartesian or spherical 'car', 'sph'

(section) Example:

```
>>> get_dtype(3)
('GSE', 'car')
```

(section) See Also:

get_sysaxes

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 05-Mar-2010 (JK)

get_Lstar(*ticktock*, *spaco*, *alpha*, *extMag*=`'T01STORM'`, *options*=`[1, 0, 0, 0, 0]`, *omnivals*=`None`)

This will call make_lstar1 or make_lstar_shell_splitting_1 from the onera library and will lookup omni values for given time if not provided (optional). If pitch angles are provided, drift shell splitting will be calculated and "Bmirr" will be returned. If they are not provided, then no drift shell splitting is calculated and "Blocal" is returned.

(section) Input:

- ticktock (TickTock class) : containing time information
- spaco (Spaco class) : containing spatial information
- alpha (list or ndarray) : optional pitch angles in degrees; if provided will calculate shell splitting; max 25 values
- extMag (string) : optional; will choose the external magnetic field model possible values ['0', 'MEAD', 'T87SHORT', 'T87LONG', 'T89', 'OPQUIET', 'OPDYN', 'T96', 'OSTA', 'T01QUIET', 'T01STORM', 'T05', 'ALEX']
- options (optional list or array of integers length=5) : explained below
- omni values as dictionary (optional) : if not provided, will use lookup table

(section) Returns:

- results (dictionary) : containing keys: Lm, Lstar, Bmin, Blocal (or Bmirr), Xj, MLT if pitch angles provided in "alpha" then drift shells are calculated and "Bmirr" is returned if not provided, then "Blocal" at spacecraft is returned.

(section) Example:

```
>>> t = TickTock(['2002-02-02T12:00:00', '2002-02-02T12:10:00'], 'ISO')
>>> y = SpaCo([[3,0,0],[2,0,0]], 'GEO', 'car')
>>> spacepy.onerapy.Lstar(t,y)
{'Blocal': array([ 1020.40493286,  3446.08845227]),
    'Bmin': array([ 1019.98404311,  3437.63865243]),
    'Lm': array([ 3.08948304,  2.06022102]),
    'Lstar': array([ 2.97684043,  1.97868577]),
    'MLT': array([ 23.5728333 ,  23.57287944]),
    'Xj': array([ 0.00112884,  0.00286955])}
```

(section) External Magnetic Field:

- 0 : no external field
- MEAD : Mead & Fairfield [1975] (uses 0<=Kp<=9 - Valid for rGEO<=17. Re)
- T87SHORT: Tsyganenko short [1987] (uses 0<=Kp<=9 - Valid for rGEO<=30. Re)
- T87LONG : Tsyganenko long [1987] (uses 0<=Kp<=9 - Valid for rGEO<=70. Re)
- T89 : Tsyganenko [1989] (uses 0<=Kp<=9 - Valid for rGEO<=70. Re)
- OPQUIET : Olson & Pfitzer quiet [1977] (default - Valid for rGEO<=15. Re)
- OPDYN : Olson & Pfitzer dynamic [1988] (uses 5.<=dens<=50., 300.<=velo<=500., -100.<=Dst<=20. - Valid for rGEO<=60. Re)
- T96 : Tsyganenko [1996] (uses -100.<=Dst (nT)<=20., 0.5<=Pdyn (nPa)<10., |ByIMF| (nT)<1=0., |BzIMF| (nT)<=10. - Valid for rGEO<=40. Re)
- OSTA : Ostapenko & Maltsev [1997] (uses dst,Pdyn,BzIMF, Kp) T01QUIET: Tsyganenko [2002a,b] (uses -50.<Dst (nT)<20., 0.5<Pdyn (nPa)<=5., |ByIMF| (nT)<=5., |BzIMF| (nT)<=5., 0.<=G1<=10., 0.<=G2<=10. - Valid for xGSM>=-15. Re)
- T01STORM: Tsyganenko, Singer & Kasper [2003] storm (uses Dst, Pdyn, ByIMF, BzIMF, G2, G3 - there is no upper or lower limit for those inputs - Valid for xGSM>=-15. Re)
- T05 : Tsyganenko & Sitnov [2005] storm (uses Dst, Pdyn, ByIMF, BzIMF, W1, W2,

prep_onera(*ticktock*=`None`, *spaco*=`None`, *alpha*=`[]`, *extMag*=`'T01STORM'`, *options*=`[1, 0, 0, 0, 0]`, *omnivals*=`None`)

5.2 Variables

Name	Description
__log__	**Value:** `'\n$Log: onerapy.py,v $\nRevision 1.5 2010/05/17 17:38:5...`
__package__	**Value:** `'spacepy'`

6 Module spacepy.onerapylib

```
This module 'onerapylib' is auto-generated with f2py (version:2).
Functions:
  lm,lstar,blocal,bmin,xj,mlt = make_lstar1(ntime,kext,options,sysaxes,iyearsat,idoy,ut,xin1,xin2,xin3,m
  lm,lstar,blocal,bmin,xj,mlt = make_lstar_shell_splitting1(ntime,nipa,kext,options,sysaxes,iyearsat,ido
  blocal,bmir,xgeo = find_mirror_point1(kext,options,sysaxes,iyearsat,idoy,ut,xin1,xin2,xin3,alpha,magin
  bmin,posit = find_magequator1(kext,options,sysaxes,iyearsat,idoy,ut,xin1,xin2,xin3,maginput)
  bxgeo,bl = get_field1(kext,options,sysaxes,iyearsat,idoy,ut,xin1,xin2,xin3,maginput)
  xout = coord_trans1(sysaxesin,sysaxesout,iyr,idoy,secs,xin)
COMMON blocks:
  /index/ activ
  /magmod/ k_ext,k_l,kint
  /gener/ era,aquad,bquad
  /drivers/ density,speed,dst_nt,pdyn_npa,byimf_nt,bzimf_nt,g1_tsy01,g2_tsy01,fkp,g3_tsy01,w1_tsy04,w2_tsy
  /dip_ang/ tilt
  /a2000_time/ a2000_ut,a2000_iyear,a2000_imonth,a2000_iday
  /flag_l/ ilflag
.
```

Version: $Revision: $

6.1 Variables

Name	Description
__package__	**Value:** `None`
a2000_time	**Value:** `<fortran object at 0x1a62cb0>`
coord_trans1	**Value:** `<fortran object at 0x1a62b20>`
dip_ang	**Value:** `<fortran object at 0x1a62c88>`
drivers	**Value:** `<fortran object at 0x1a62530>`
find_magequator1	**Value:** `<fortran object at 0x1a62c38>`
find_mirror_point1	**Value:** `<fortran object at 0x1a62788>`
flag_l	**Value:** `<fortran object at 0x1a62b98>`
gener	**Value:** `<fortran object at 0x1a62350>`
get_field1	**Value:** `<fortran object at 0x1a62c60>`
index	**Value:** `<fortran object at 0x1a628a0>`
magmod	**Value:** `<fortran object at 0x1a62af8>`
make_lstar1	**Value:** `<fortran object at 0x1a62878>`
make_lstar_shell_splitting1	**Value:** `<fortran object at 0x1a62c10>`

7 Module spacepy.poppy

PoPPy – Point Processes in Python.

This module contains point process class types and a variety of functions for association analysis. The routines given here grew from work presented by Morley and Freeman (Geophysical Research Letters, 34, L08104, doi:10.1029/ 2006GL028891, 2007), which were originally written in IDL. This module is intended for application to discrete time series of events to assess statistical association between the series and to calculate confidence limits. Any mis-application or mis-interpretation by the user is the user's own fault.

Each instance must be initialized with:

```
>>> obj = poppy.PPro(series1, series2)
```

To perform association analysis

```
>>> obj.assoc(u=lags, h=halfwindow)
```

To plot

```
>>> obj.aaplot()
```

–++– By Steve Morley –++–

smorley@lanl.gov/morley_steve@hotmail.com, Los Alamos National Laboratory, ISR-1, PO Box 1663, Los Alamos, NM 87545

Author: Steve Morley (smorley@lanl.com/morley_steve@hotmail.com)

7.1 Functions

boots_ci(*data*, *n*, *inter*, *func*)

Construct bootstrap confidence interval - caution: slow!

Input: n is number of surrogates; data is data array (1D); inter is desired confidence interval (e.g. 95%); func is a user-defined function (lambda)

Example:

```
>>> data, n = numpy.random.lognormal(mean=5.1, sigma=0.3, size=3000), 4000.
>>> myfunc = lambda x: numpy.median(x)
>>> ci_low, ci_high = poppy.boots_ci(data, n, 95, myfunc)
>>> ci_low, numpy.median(data), ci_high
(163.96354196633686, 165.2393331896551, 166.60491435416566) iter. 1
... repeat
(162.50379144492726, 164.15218265100233, 165.42840588032755) iter. 2
```

For comparison:

```
>>> data = numpy.random.lognormal(mean=5.1, sigma=0.3, size=90000)
>>> numpy.median(data)
163.83888237895815
```

Note that the true value of the desired quantity may lie outside the 95% confidence interval one time in 20 realizations. This occurred for the first iteration - please feel free to play with this.

7.2 Variables

Name	Description
__package__	**Value:** `'spacepy'`

7.3 Class PPro

object ─┐
spacepy.poppy.PPro

PoPPy point process object

Initialize object with series1 and series2. These should be timeseries of events, given as lists, arrays, or lists of datetime objects. Includes method to perform association analysis of input series

Output can be nicely plotted with plot method

7.3.1 Methods

__init__(*self*, *process1*, *process2*, *lags*=`None`, *winhalf*=`None`)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

String Representation of PoPPy object

Overrides: object.__str__

__repr__(*self*)

String Representation of PoPPy object

Overrides: object.__repr__

__len__(*self*)

Calling len(obj) will return the number of points in process 1

swap(*self*)

Swaps process 1 and process 2

assoc(*self*, *u*=`None`, *h*=`None`)

Perform association analysis on input series

u = range of lags h = association window half-width

plot(*self*, *figsize*=`None`, *dpi*=`300`)

Method called to create basic plot of association analysis.

Inputs: Uses object attributes created by the obj.assoc() method.

Optional keyword(s): usrlimy (default = []) - override automatic y-limits on plot.

aa_ci(*self*, *inter*, *n_boots*=`1000`)

Get bootstrap confidence intervals for association number

Requires input of desired confidence interval, e.g.,

```
>>> obj.aa_ci(95)
```

Upper and lower confidence limits are added to the ci attribute

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

7.3.2 Properties

Name	Description
Inherited from object	
__class__	

8 Module spacepy.radbelt

Functions supporting radiation belt diffusion codes

Version: $Revision: 1.7 $, $Date: 2010/05/17 17:38:51 $

Author: J. Koller, Los Alamos National Lab (jkoller@lanl.gov)

8.1 Functions

diff_LL(*grid*, *f*, *Tdelta*, *params*=`None`)

calculate the diffusion in L at constant mu,K coordinates time units

get_modelop_L(*Lgrid*, *Tdelta*, *DLL*)

calculate the model oparator for a single small timestep

get_DLL(*Lgrid*, *params*, *DLL_model*=`'BA2000'`)

calculate the diffusion coefficient D_LL

get_local_accel(*Lgrid*, *params*, *SRC_model*=`'JK1'`)

calculate the diffusion coefficient D_LL

8.2 Variables

Name	Description
__log__	**Value:** `'\n$Log: radbelt.py,v $\nRevision 1.7 2010/05/17 17:38:5...`
__package__	**Value:** `'spacepy'`

8.3 Class RBmodel

```
object ─┐
        │
     spacepy.radbelt.RBmodel
```

1-D Radial diffusion class

This module contains a class for performing and visualizing 1-D radial diffusion simulations of the radiation belts.

Here is an example using the default settings of the model. Each instance must be initialized with (assuming import radbelt as rb):

```
>>> rmod = rb.RBmodel()
```

Next, set the start time, end time, and the size of the timestep:

```
>>> start = datetime.datetime(2003,10,14)
>>> end = datetime.datetime(2003,12,26)
>>> delta = datetime.timedelta(hours=1)
>>> rmod.setup_ticks(start, end, delta, dtype='UTC')
```

Now, run the model over the enitre time range using the evolve method:

```
>>> rmod.evolve()
```

Finally, visualize the results:

```
>>> rmod.plot_summary()
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 17-Mar-2010 (JK) v1.01 12-May-2010 (dtw)

8.3.1 Methods

__init__(*self*, *grid*=`'L'`)

format for grid e.g., L-PA-E

Overrides: object.__init__

__str__(*self*)

str(x)

Overrides: object.__str__

__repr__(*self*)

str(x)

Overrides: object.__repr__

__getitem__(*self*, *idx*)

setup_ticks(*self*, *start*, *end*, *delta*, *dtype*=`'ISO'`)

add time TickTock information to class instance

add_omni(*self*, *keylist*=`None`)

add omni data to instance according to the tickrange in ticktock

add_Lmax(*self*, *Lmax_model*)

add last closed drift shell Lmax

add_Lpp(*self*, *Lpp_model*)

add last closed drift shell Lmax

add_obs(*self*, *satlist*=`None`)

add observations from PSDdb using the ticktock list

evolve(*self*)

calculate the diffusion in L at constant mu,K coordinates

assimilate(*self*, *method*=`'enKF'`)

call data assimilation function in assimilate.py

plot(*self*, *Lmax*=`True`, *Lpp*=`False`, *Kp*=`True`, *Dst*=`True`, *clims*=`[0, 10]`, *title*=`'Summary Plot'`)

Create a summary plot of the RadBelt object distribution function. For reference, the last closed drift shell, Dst, and Kp are all included. These can be disabled individually using the corresponding boolean kwargs.

The clims kwarg can be used to manually set the color bar range. To use, set it equal to a two-element list containing minimum and maximum Log_10 value to plot. Default action is to use [0,10] as the log_10 of the color range. This is good enough for most applications.

The title of the top most plot defaults to 'Summary Plot' but can be customized using the title kwarg.

The figure object and all three axis objects (PSD axis, Dst axis, and Kp axis) are all returned to allow the user to further customize the plots as necessary. If any of the plots are excluded, None is returned in their stead.

Example:

```
>>> rb.plot(Lmax=False, Kp=False, clims=[2,10], title='Good work!')
```

This command would create the summary plot with a color bar range of 100 to 10^10. The Lmax line and Kp values would be excluded. The title of the topmost plot (phase space density) would be set to 'Good work!'.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

8.3.2 Properties

Name	Description
Inherited from object	
__class__	

9 Module spacepy.satdata

Spacecraft data functions

Version: $Revision: 1.1 $, $Date: 2010/05/14 23:06:25 $

Author: J. Koller, Los Alamos National Lab (jkoller@lanl.gov)

9.1 Functions

get_obs(*tickstart*, *tickend*, *MU*, *K*, *satlist*=`None`)

return all observations between tickstart and tickend possible MU values: 167, 462, 1051, 2083 possible K values: 0.01, 0.03, 0.1, 0.3, 1.0

convert2PSDclass(*fln*, *MUval*, *Kval*, *satname*)

updatePSDdb(*path*, *satlist*=`None`)

```
calls makePSDdb and updates all
    example: updatePSDdb('/Users/jkoller/Research/input_data/')
```

9.2 Variables

Name	Description
__log__	**Value:** `...`
PSDfln	**Value:** `__path__ [0]+ '/data/PSDdb.pbin'`
PSDdb	**Value:** `loadpickle(PSDfln)`

9.3 Class PSDdataClass

```
object ─┐
        spacepy.satdata.PSDdataClass
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 17-Mar-2010 (JK)

9.3.1 Methods

__init__(*self*, *MUval*, *Kval*, *satname*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

str(x)

Overrides: object.__str__

__repr__(*self*)

str(x)

Overrides: object.__repr__

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

9.3.2 Properties

Name	Description
Inherited from object	
__class__	

10 Module spacepy.seapy

SeaPy – Superposed Epoch in Python.

This module contains superposed epoch class types and a variety of functions for using on superposed epoch objects. Each instance must be initialized with (assuming import seapy as se):

```
>>> obj = se.Sea(data, times, epochs)
```

To perform a superposed epoch analysis

```
>>> obj.sea()
```

To plot

```
>>> obj.seplot()
```

If multiple SeaPy objects exist, these can be combined into a single object

```
>>> objdict = seadict([obj1, obj2],['obj1name','obj2name'])
```

and then used to create a multipanel plot

```
>>> seamulti(objdict)
```

For two-dimensional superposed epoch analyses, initialize an Sea2d() instance

```
>>> obj = se.Sea2d(data, times, epochs, y=[4., 12.])
```

All object methods are the same as for the 1D object. Also, the seamulti() function should accept both 1D and 2D objects, even mixed together. Currently, the seplot() method is recommended for 2D SEA.

–++– By Steve Morley –++–

smorley@lanl.gov/morley_steve@hotmail.com, Los Alamos National Laboratory, ISR-1, PO Box 1663, Los Alamos, NM 87545

Author: Steve Morley (smorley@lanl.gov/morley_steve@hotmail.com)

10.1 Functions

seadict(*objlist*, *namelist*)

Function to create dictionary of SeaPy.Sea objects.

Inputs: objlist - List of Sea objects. namelist - List of variable labels for input objects. Optional keyword(s): namelist = List containing names for y-axes.

multisea(*dictobj*, *n_cols*=`1`, *epochline*=`False`, *usrlimx*=`[]`, *usrlimy*=`[]`, *xunits*=`''`, *show*=`True`, *zunits*=`''`, *zlog*=`True`, *figsize*=`None`, *dpi*=`300`)

Function to create multipanel plot of superposed epoch analyses.

Inputs: Dictionary of Sea objects (from superposedepoch.seadict()). Optional keyword(s): epochline (default = False) - put vertical line at zero epoch. usrlimy (default = []) - override automatic y-limits on plot (same for all plots). show (default = True) - shows plot; set to false to output plot object to variable x/zunits - Units for labelling x and z axes, if required figsize - tuple of (width, height) in inches dpi (default=300) - figure resolution in dots per inch n_cols - Number of columns: not yet implemented.

Output: Plot of input object median and bounds (ci, mad, quartiles - see sea()). If keyword 'show' is False, output is a plot object.

readepochs(*fname*, *iso*=`False`, *isofmt*=`'%Y-%m-%dT%H:%M:%S'`)

Read epochs from text file assuming YYYY MM DD hh mm ss format

Input: Filename (include path) Optional inputs: iso (default = False), read in ISO date format isofmt (default is YYYY-mm-ddTHH:MM:SS, code is %Y-%m-%dT%H:%M:%S) Output: epochs (type=list)

10.2 Variables

Name	Description
__package__	**Value:** `'spacepy'`

10.3 Class Sea

```
object ─┐
        │
      spacepy.seapy.Sea
```

Known Subclasses: spacepy.seapy.Sea2d

SeaPy Superposed epoch analysis object

Initialize object with data, times, epochs, window (half-width) and delta (optional). 'times' and epochs should be in some useful format Includes method to perform superposed epoch analysis of input data series

Output can be nicely plotted with seaplot method, or for multiple objects use the seamulti function

10.3.1 Methods

__init__(*self*, *data*, *times*, *epochs*, *window*=`3.0`, *delta*=`1.0`)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

Define String Representation of Sea object

Overrides: object.__str__

__repr__(*self*)

Define String Representation of Sea object

Overrides: object.__repr__

__len__(*self*)

Calling len(obj) will return the number of epochs.

restoreepochs(*self*)

Replaces epoch times stored in obj.badepochs in the epochs attribute

Quite why you'd want this feature I don't know.

sea(*self*, *storedata*=`False`, *quartiles*=`True`, *ci*=`False`, *mad*=`False`, *ci_quan*=`'median'`)

Method called to perform superposed epoch analysis on data in object.

Inputs: Uses object attributes obj.data, obj.times, obj.epochs, obj.delta, obj.window, all of which must be available on instatiation. Optional keyword(s): storedata (default = False) - saves matrix of epoch windows as obj.datacube quartiles calculates the quartiles as the upper and lower bounds (and is default); ci will find the bootstrapped confidence intervals (and requires ci_quan to be set); mad will use +/- the median absolute deviation for the bounds; ci_quan can be set to 'median' or 'mean'

A basic plot can be raised with the obj.seplot() method

plot(*self*, *xquan*=`'Time Since Epoch'`, *yquan*=`''`, *xunits*=`''`, *yunits*=`''`, *epochline*=`False`, *usrlimy*=`[]`, *figsize*=`None`, *dpi*=`300`)

Method called to create basic plot of superposed epoch analysis.

Inputs: Uses object attributes created by the obj.sea() method.

Optional keyword(s): x(y)quan (default = 'Time since epoch' (None)) - x(y)-axis label. x(y)units (default = None (None)) - x(y)-axis units. epochline (default = False) - put vertical line at zero epoch. usrlimy (default = []) - override automatic y-limits on plot.

If both ?quan and ?units are supplied, axis label will read 'Quantity Entered By User [Units]'

seplot(*self*, *xquan*=`'Time Since Epoch'`, *yquan*=`''`, *xunits*=`''`, *yunits*=`''`, *epochline*=`False`, *usrlimy*=`[]`, *figsize*=`None`, *dpi*=`300`)

Method called to create basic plot of superposed epoch analysis.

Inputs: Uses object attributes created by the obj.sea() method.

Optional keyword(s): x(y)quan (default = 'Time since epoch' (None)) - x(y)-axis label. x(y)units (default = None (None)) - x(y)-axis units. epochline (default = False) - put vertical line at zero epoch. usrlimy (default = []) - override automatic y-limits on plot.

If both ?quan and ?units are supplied, axis label will read 'Quantity Entered By User [Units]'

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

10.3.2 Properties

Name	Description
Inherited from object	
__class__	

10.4 Class Sea2d

object ┐

spacepy.seapy.Sea ┐

spacepy.seapy.Sea2d

SeaPy 2D Superposed epoch analysis object

Initialize object with data, times, epochs, window (half-width), delta (optional), and y (two-element vector with max and min of y;optional) 'times' and epochs should be in some useful format Includes method to perform superposed epoch analysis of input data series

Output can be nicely plotted with seplot method, or for multiple objects use the seamulti function

10.4.1 Methods

__init__(*self*, *data*, *times*, *epochs*, *window*=`3.0`, *delta*=`1.0`, *y*=`[]`)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

sea(*self*, *storedata*=`False`, *quartiles*=`True`, *ci*=`False`, *mad*=`False`, *ci_quan*=`'median'`, *nmask*=`1`)

Method called to perform 2D superposed epoch analysis on data in object.

Inputs: Uses object attributes obj.data, obj.times, obj.epochs, obj.delta, obj.window, all of which must be available on instatiation. Optional keyword(s): storedata (default = False) - saves matrix of epoch windows as obj.datacube quartiles calculates the interquartile range to show the spread (and is default); ci will find the bootstrapped confidence interval (and requires ci_quan to be set); mad will use the median absolute deviation for the spread; ci_quan can be set to 'median' or 'mean'

A basic plot can be raised with the obj.seplot() method

Overrides: spacepy.seapy.Sea.sea

plot(*self*, *xquan*=`'Time Since Epoch'`, *yquan*=`''`, *xunits*=`''`, *yunits*=`''`, *zunits*=`''`, *epochline*=`False`, *usrlimy*=`[]`, *show*=`True`, *zlog*=`True`, *figsize*=`None`, *dpi*=`300`)

Method called to create basic plot of 2D superposed epoch analysis.

Inputs: Uses object attributes created by the obj.sea() method. Optional keyword(s): x(y)quan (default = 'Time since epoch' (None)) - x(y)-axis label. x(y/z)units (default = None (None)) - x(y/z)-axis units. epochline (default = False) - put vertical line at zero epoch. usrlimy (default = []) - override automatic y-limits on plot. show (default = True) - shows plot; set to false to output plot object to variable figsize - tuple of (width, height) in inches dpi (default=300) - figure resolution in dots per inch If both ?quan and ?units are supplied, axis label will read 'Quantity Entered By User [Units]'

Overrides: spacepy.seapy.Sea.plot

seplot(*self*, *xquan*=`'Time Since Epoch'`, *yquan*=`''`, *xunits*=`''`, *yunits*=`''`, *zunits*=`''`, *epochline*=`False`, *usrlimy*=`[]`, *show*=`True`, *zlog*=`True`, *figsize*=`None`, *dpi*=`300`)

Method called to create basic plot of 2D superposed epoch analysis.

Inputs: Uses object attributes created by the obj.sea() method. Optional keyword(s): x(y)quan (default = 'Time since epoch' (None)) - x(y)-axis label. x(y/z)units (default = None (None)) - x(y/z)-axis units. epochline (default = False) - put vertical line at zero epoch. usrlimy (default = []) - override automatic y-limits on plot. show (default = True) - shows plot; set to false to output plot object to variable figsize - tuple of (width, height) in inches dpi (default=300) - figure resolution in dots per inch If both ?quan and ?units are supplied, axis label will read 'Quantity Entered By User [Units]'

Overrides: spacepy.seapy.Sea.seplot

Inherited from spacepy.seapy.Sea(Section 10.3)

__len__(), __repr__(), __str__(), restoreepochs()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

10.4.2 Properties

Name	Description
Inherited from object	

continued on next page

Name	Description
__class__	

11 Module spacepy.setup

Version: $Revision: 1.12 $, $Date: 2010/05/14 21:02:27 $

Author: Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

11.1 Functions

compile_oneralib()

subst(*pattern*, *replacement*, *filestr*, *pattern_matching_modifiers*=`None`)

replace pattern by replacement in file pattern_matching_modifiers: re.DOTALL, re.MULTILINE, etc.

11.2 Variables

Name	Description
__log__	**Value:** `...`
pkg_files	**Value:** `['onerapylib.so', 'data/omnidata.pbin', 'data/tai-utc.dat...`
docfiles	**Value:** `[df for df in docfiles if not re.search('CVS', df)]`

12 Module spacepy.spacetime

Implementation of TickTock class and SpaCo class functions

Version: $Revision: 1.14 $, $Date: 2010/05/17 17:38:51 $

Author: Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

12.1 Functions

doy2date(*year*, *doy*, *dtobj*=`False`)

convert day-of-year doy into a month and day after
http://pleac.sourceforge.net/pleac_python/datesandtimes.html

(section) Input:

- year (int or array of int) : year
- doy (int or array of int) : day of year

(section) Returns:

- month (int or array of int) : month as integer number
- day (int or array of int) : as integer number

(section) Example:

```
>>> month, day = doy2date(2002, 186)
>>> dts = doy2date([2002]*4, range(186,190), dtobj=True)
```

(section) See Also:

getDOY

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 24-Jan-2010: can handle arrays as input (JK) V2: 02-Apr-2010: option to return date objects (SM) V3: 07-Apr-2010: modified to return datetime objects (SM)

tickrange(*start*, *end*, *deltadays*, *dtype*='ISO')

return a TickTock range given the start, end, and delta

(section) Input:

- start (string or number) : start time
- end (string or number) : end time (inclusive)
- deltadays: step in units of days (float); or datetime timedelta object
- (optional) dtype (string) : data type for start, end; e.g. ISO, UTC, RTD, etc. see TickTock for all options

(section) Returns:

- ticks (TickTock instance)

(section) Example:

```
>>> ticks = st.tickrange('2002-02-01T00:00:00', '2002-02-10T00:00:00', deltadays = 1)
>>> ticks
TickTock( ['2002-02-01T00:00:00', '2002-02-02T00:00:00', '2002-02-03T00:00:00',
'2002-02-04T00:00:00'] ), dtype=ISO
```

(section) See Also:

TickTock

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 10-Mar-2010: (JK) V1.1: 16-Mar-2010: fixed bug with floating point precision (JK) V1.2: 28-Apr-2010: added timedelta support for increment (SM)

(section) Notes:

This, and TickTock, do not currently support sub-second precision. This should be fixed.

12.2 Variables

Name	Description
__log__	**Value:** `'\n$Log: spacetime.py,v $\nRevision 1.14 2010/05/17 17:3...`

continued on next page

Name	Description
__package__	**Value:** `'spacepy'`

12.3 Class SpaCo

object ─┐
　　　spacepy.spacetime.SpaCo

a = Spaco(data, dtype, carsph, [units, ticktock])

A class holding spatial coordinates in cartesian/spherical in units of Re and degrees

(section) Input:

- data (list or ndarray, dim = (n,3)) : coordinate points
- dtype (string) :coordinate system, possible are GDZ, GEO, GSM, GSE, SM, GEI MAG, SPH, RLL
- carsph (string) : cartesian or spherical, 'car' or 'sph'
- optional units (list of strings) : standard are ['Re', 'Re', 'Re'] or ['Re', 'deg', 'deg'] depending on the carsph content
- optional ticktock (TickTock instance) : used for coordinate transformations (see a.convert)

(section) Returns:

- instance with a.data, a.carsph, etc.

(section) Example:

```
>>> coord = SpaCo([[1,2,4],[1,2,2]], 'GEO', 'car')
>>> coord.x  # returns all x coordinates
array([1, 1])
>>> coord.ticktock = TickTock(['2002-02-02T12:00:00', '2002-02-02T12:00:00'], 'ISO') # add ticktock
>>> newcoord = coord.convert('GSM', 'sph')
>>> newcoord
```

(section) See Also:

spacetime.TickTock class

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 05-Mar-2010 (JK)

12.3.1 Methods

__init__(*self*, *data*, *dtype*, *carsph*, *units*=None, *ticktock*=None)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

a.__str__() or a

Will be called when printing SpaCo instance a

(section) Input:

- a SpaCo class instance

(section) Returns:

- output (string)

(section) Example:

```
>>> y = SpaCo([[1,2,4],[1,2,2]], 'GEO', 'car')
>>> y
SpaCo( [[1 2 4]
 [1 2 2]] ), dtype=GEO,car, units=['Re', 'Re', 'Re']
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 05-Mar-2010 (JK)

Overrides: object.__str__

__repr__(*self*)

a.__str__() or a

Will be called when printing SpaCo instance a

(section) Input:

- a SpaCo class instance

(section) Returns:

- output (string)

(section) Example:

```
>>> y = SpaCo([[1,2,4],[1,2,2]], 'GEO', 'car')
>>> y
SpaCo( [[1 2 4]
 [1 2 2]] ), dtype=GEO,car, units=['Re', 'Re', 'Re']
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 05-Mar-2010 (JK)

Overrides: object.__repr__

__getitem__(*self*, *idx*)

a.__getitem__(idx) or a[idx]

Will be called when requesting items in this instance

(section) Input:

- a SpaCo class instance
- idx (int) : interger numbers as index

(section) Returns:

- vals (numpy array) : new values

(section) Example:

```
>>> y = SpaCo([[1,2,4],[1,2,2]], 'GEO', 'car')
>>> y[0]
array([1, 2, 4])
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 05-Mar-2010 (JK) V1.1: 19-Apr-2010: now returns a SpaCo instance

__setitem__(*self*, *idx*, *vals*)

a.__setitem__(idx, vals) or a[idx] = vals

Will be called setting items in this instance

(section) Input:

- a SpaCo class instance
- idx (int) : interger numbers as index
- vals (numpy array or list) : new values

(section) Example:

```
>>> y = SpaCo([[1,2,4],[1,2,2]], 'GEO', 'car')
>>> y[1] = [9,9,9]
>>> y
SpaCo( [[1 2 4]
 [9 9 9]] ), dtype=GEO,car, units=['Re', 'Re', 'Re']
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 05-Mar-2010 (JK)

__len__(*self*)

a.__len__() or len(a)

Will be called when requesting the length, i.e. number of items

(section) Input:

- a SpaCo class instance

(section) Returns:

- length (int number)

(section) Example:

```
>>> y = SpaCo([[1,2,4],[1,2,2]], 'GEO', 'car')
>>> len(y)
2
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 05-Mar-2010 (JK)

convert(*a*, *returntype*, *returncarsph*)

Can be used to create a new SpaCo instance with new coordinate types

(section) Input:

- a SpaCo class instance
- returntype (string) : coordinate system, possible are GDZ, GEO, GSM, GSE, SM, GEI MAG, SPH, RLL
- returncarsph (string) : coordinate type, possible 'car' for cartesian and 'sph' for spherical

(section) Returns:

- a SpaCo object

(section) Example:

```
>>> y = SpaCo([[1,2,4],[1,2,2]], 'GEO', 'car')
>>> y.ticktock = TickTock(['2002-02-02T12:00:00', '2002-02-02T12:00:00'], 'ISO')
>>> x = y.convert('SM','car')
>>> x
SpaCo( [[ 0.81134097  2.6493305   3.6500375 ]
 [ 0.92060408  2.30678864  1.68262126]] ), dtype=SM,car, units=['Re', 'Re', 'Re']
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 05-Mar-2010 (JK)

__getstate__(*self*)

Is called when pickling Author: J. Koller See Also: http://docs.python.org/library/pickle.html

__setstate__(*self*, *dict*)

Is called when unpickling Author: J. Koller See Also: http://docs.python.org/library/pickle.html

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

12.3.2 Properties

Name	Description
Inherited from object	
__class__	

12.4 Class TickDelta

```
object ─┐
        spacepy.spacetime.TickDelta
```

TickDelta(**kwargs)

TickDelta class holding timedelta similar to datetime.timedelta This can be used to add/substract from TickTock objects

(section) Input:

- days, hours, minutes and/or seconds (int or float) : time step

(section) Returns:

- instance with self.days, self.secs, self.timedelta

(section) Example:

```
>>> dt = TickDelta(days=3.5, hours=12)
>>> dt
TickDelta( days=4.0 )
```

(section) See Also:

spacetime.TickTock class

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

12.4.1 Methods

__init__(***kwargs*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

dt.__str__() or dt

Will be called when printing TickDelta instance dt

(section) Input:

- a TickDelta class instance

(section) Returns:

- output (string)

(section) Example:

```
>>> dt = TickDelta(3)
>>> dt
TickDelta( days=3 )
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

Overrides: object.__str__

__repr__(*self*)

dt.__str__() or dt

Will be called when printing TickDelta instance dt

(section) Input:

- a TickDelta class instance

(section) Returns:

- output (string)

(section) Example:

```
>>> dt = TickDelta(3)
>>> dt
TickDelta( days=3 )
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

Overrides: object.__repr__

__add__(*self*, *other*)

see TickTock.__add__

__sub__(*self*, *other*)

see TickTock.__sub__

__mul__(*self*, *other*)

see TickTock.__sub__

__getstate__(*self*)

Is called when pickling Author: J. Koller See Also: http://docs.python.org/library/pickle.html

__setstate__(*self*, *dict*)

Is called when unpickling Author: J. Koller See Also: http://docs.python.org/library/pickle.html

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

12.4.2 Properties

Name	Description
Inherited from object	
__class__	

12.5 Class TickTock

```
object ─┐
        │
        spacepy.spacetime.TickTock
```

TickTock(data, dtype)

TickTock class holding various time coordinate systems (TAI, UTC, ISO, JD, MJD, UNX, RDT, CDF, DOY, eDOY)

Possible data types: ISO: ISO standard format like '2002-02-25T12:20:30' UTC: datetime object with UTC time TAI: elapsed seconds since 1958/1/1 (includes leap seconds) UNX: elapsed seconds since 1970/1/1 (all days have 86400 secs sometimes unequal lenghts) JD: Julian days elapsed MJD: Modified Julian days RDT: Rata Die days elapsed since 1/1/1 CDF: CDF epoch: milliseconds since 1/1/0000

(section) Input:

- data (int, datetime, float, string) : time stamp
- dtype (string) : data type for data, possible values: CDF, ISO, UTC, TAI, UNX, JD, MJD, RDT

(section) Returns:

- instance with self.data, self.dtype, self.UTC etc

(section) Example:

```
>>> x=TickTock(2452331.0142361112, 'JD')
>>> x.ISO
'2002-02-25T12:20:30'
>>> x.DOY # Day of year
56
```

(section) See Also:

a.getCDF, a.getISO, a.getUTC, etc.

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010 V2: 25-Jan-2010: includes array support (JK) V3: 25-feb-2010: pulled functions into class (JK)

12.5.1 Methods

__init__(*data*, *dtype*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

__str__(*self*)

a.__str__() or a

Will be called when printing TickTock instance a

(section) Input:

- a TickTock class instance

(section) Returns:

- output (string)

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:03', 'ISO')
>>> a
TickTock( ['2002-02-02T12:00:00'] ), dtype=ISO
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

Overrides: object.__str__

__repr__(*self*)

a.__str__() or a

Will be called when printing TickTock instance a

(section) Input:

- a TickTock class instance

(section) Returns:

- output (string)

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:03', 'ISO')
>>> a
TickTock( ['2002-02-02T12:00:00'] ), dtype=ISO
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

Overrides: object.__repr__

__getstate__(*self*)

Is called when pickling Author: J. Koller See Also: http://docs.python.org/library/pickle.html

__setstate__(*self, dict*)

Is called when unpickling Author: J. Koller See Also: http://docs.python.org/library/pickle.html

__getitem__(*self*, *idx*)

a.__getitem__(idx) or a[idx]

Will be called when requesting items in this instance

(section) Input:

- a TickTock class instance
- idx (int) : interger numbers as index

(section) Returns:

- TickTock instance with requested values

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:03', 'ISO')
>>> a[0]
'2002-02-02T12:00:00'
```

(section) See Also:

a.__setitem__

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1.0: 03-Mar-2010 (JK) V1.1: 23-Mar-2010: now returns TickTock instance and can be indexed with arrays (JK)

__setitem__(*self*, *idx*, *vals*)

a.__setitem__(idx, vals) or a[idx] = vals

Will be called setting items in this instance

(section) Input:

- a TickTock class instance
- idx (int) : interger numbers as index
- vals (float, string, datetime) : new values

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:03', 'ISO')
>>> a[0] = '2003-03-03T00:00:00'
```

(section) See Also:

a.__getitem__

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

__len__(*self*)

a.__len__() or len(a)

Will be called when requesting the length, i.e. number of items

(section) Input:

- a TickTock class instance

(section) Returns:

- length (int number)

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:03', 'ISO')
>>> a.len
1
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

__cmp__(*a*, *other*)

Will be called when two TickTock instances are compared

(section) Input:

- a TickTock class instance
- other (TickTock instance)

(section) Returns:

- True or False

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:03', 'ISO')
>>> b = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a > b
True
```

(section) See Also:

a.__add__, a.__sub__

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

__gt__(*self*, *other*)

__lt__(*self*, *other*)

__ge__(*self*, *other*)

__le__(*self*, *other*)

__eq__(*self*, *other*)

__ne__(*self*, *other*)

__sub__(*a*, *other*)

Will be called if a TickDelta object is substracted to this instance and returns a new TickTock instance

(section) Input:

- a TickTock class instance
- other (TickDelta instance)

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> dt = TickDelta(3)
>>> a - dt
TickTock( ['2002-02-05T12:00:00'] ), dtype=ISO
```

(section) See Also:

__sub__

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

__add__(*a*, *other*)

Will be called if a TickDelta object is added to this instance and returns a new TickTock instance

(section) Input:

- a TickTock class instance
- other (TickDelta instance)

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> dt = TickDelta(3)
>>> a + dt
TickTock( ['2002-02-05T12:00:00'] ), dtype=ISO
```

(section) See Also:

__sub__

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

__getattr__(*a*, *name*)

Will be called if attribute ”name” is not found in TickTock class instance. It will add TAI, RDT, etc

(section) Input:

- a TickTock class instance
- name (string) : a string from the list of time systems ’UTC’, ’TAI’, ’ISO’, ’JD’, ’MJD’, ’UNX’, ’RDT’, ’CDF’, ’DOY’, ’eDOY’, ’leaps’

(section) Returns:

- requested values as either list/numpy array

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

update_items(*a*, *b*, *attrib*)

After changing the self.data attribute by either __setitem__ or __add__ etc this function will update all other attributes. This function is called automatically in __add__ and __setitem__

(section) Input:

- a TickTock class instance

(section) See Also:

__setitem__, __add__, __sub__

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 03-Mar-2010 (JK)

convert(*a*, *dtype*)

convert a TickTock instance into a new time coordinate system provided in dtype

(section) Input:

- a TickTock class instance
- dtype (string) : data type for new system, possible values are CDF, ISO, UTC, TAI, UNX, JD, MJD, RDT

(section) Returns:

- newobj (TickTock instance) with new time coordinates

(section) Example:

```
>>> a = TickTock(['2002-02-02T12:00:00', '2002-02-02T12:00:00'], 'ISO')
>>> s = a.convert('TAI')
>>> type(s)
<class 'spacetime.TickTock'>
>>> s
TickTock( [1391342432 1391342432] ), dtype=TAI
```

(section) See Also:

a.CDF, a.ISO, a.UTC, etc.

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 05-Mar-2010 (JK)

append(*a*, *other*)

Will be called when another TickTock instance has to be appended to the current one

(section) Input:

- a TickTock class instance
- other (TickTock instance)

(section) Example:

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 23-Mar-2010 (JK)

getCDF(*self*)

a.getCDF() or a.CDF

Return CDF time which is milliseconds since 01-Jan-0000 00:00:00.000. Year zero" is a convention chosen by NSSDC to measure epoch values. This date is more commonly referred to as 1 BC. Remember that 1 BC was a leap year. The CDF date/time calculations do not take into account the changes to the Gregorian calendar, and cannot be directly converted into Julian date/times.

(section) Input:

- a TickTock class instance

(section) Returns:

- CDF (numpy array) : days elapsed since Jan. 1st

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.CDF
array([  6.31798704e+13])
```

(section) See Also:

getUTC, getUNX, getRDT, getJD, getMJD, getISO, getTAI, getDOY, geteDOY

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 02-Feb-2010 (JK)

getDOY(*self*)

a.DOY or a.getDOY()

extract DOY (days since January 1st of given year)

(section) Input:

- a TickTock class instance

(section) Returns:

- DOY (numpy array int) : day of the year

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.DOY
array([ 33])
```

(section) See Also:

getUTC, getUNX, getRDT, getJD, getMJD, getCDF, getTAI, getISO, geteDOY

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 25-Jan-2010 (JK) V1.1: 20-Apr-2010: returns true DOY per definition as integer (JK)

geteDOY(*self*)

a.eDOY or a.geteDOY()

extract eDOY (elapsed days since midnight January 1st of given year)

(section) Input:

- a TickTock class instance

(section) Returns:

- eDOY (numpy array) : days elapsed since midnight bbedJan. 1st

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.eDOY
array([ 32.5])
```

(section) See Also:

getUTC, getUNX, getRDT, getJD, getMJD, getCDF, getTAI, getISO, getDOY

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Apr-2010 (JK)

getJD(*self*)

a.JD or a.getJD()

convert dtype data into Julian Date (JD)

(section) Input:

- a TickTock class instance

(section) Returns:

- JD (numpy array) : elapsed days since 12:00 January 1, 4713 BC

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.JD
array([ 2452308.])
```

(section) See Also:

getUTC, getUNX, getRDT, getISO, getMJD, getCDF, getTAI, getDOY, geteDOY

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010 (JK) V2: 25-Jan-2010: added array support (JK)

getMJD(*self*)

a.MJD or a.getMJD()

convert dtype data into MJD (modified Julian date)

(section) Input:

- a TickTock class instance

(section) Returns:

- MJD (numpy array) : elapsed days since November 17, 1858 (Julian date was 2,400 000)

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.MJD
array([ 52307.5])
```

(section) See Also:

getUTC, getUNX, getRDT, getJD, getISO, getCDF, getTAI, getDOY, geteDOY

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010 (JK) V2: 25-Jan-2010: added support for arrays (JK)

getUNX(*self*)

a.UNX or a.getUNX()

convert dtype data into Unix Time (Posix Time) seconds since 1970-Jan-1 (not counting leap seconds)

(section) Input:

- a TickTock class instance

(section) Returns:

- UNX (numpy array) : elapsed secs since 1970/1/1 (not counting leap secs)

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.UNX
array([  1.01265120e+09])
```

(section) See Also:

getUTC, getISO, getRDT, getJD, getMJD, getCDF, getTAI, getDOY, geteDOY

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010 (JK) V2: 25-Jan-2010: added array support (JK)

getRDT(*self*)

a.RDT or a.RDT()

convert dtype data into Rata Die (lat.) Time (days since 1/1/0001)

(section) Input:

- a TickTock class instance

(section) Returns:

- RDT (numpy array) : elapsed days since 1/1/1

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.RDT
array([ 730883.5])
```

(section) See Also:

getUTC, getUNX, getISO, getJD, getMJD, getCDF, getTAI, getDOY, geteDOY

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010 (JK) V2: 25-Jan-2010: added array support (JK)

getUTC(*self*)

a.UTC or a.getUTC()

convert dtype data into UTC object a la datetime()

(section) Input:

- a TickTock class instance

(section) Returns:

- UTC (list of datetime objects) : datetime object in UTC time

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.UTC
[datetime.datetime(2002, 2, 2, 12, 0)]
```

(section) See Also:

getISO, getUNX, getRDT, getJD, getMJD, getCDF, getTAI, getDOY, geteDOY

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010 (JK) V2: 25-Jan-2010: added array support (JK)

getGPS(*self*)

a.GPS or a.getGPS()

return GPS epoch (0000 UT (midnight) on January 6, 1980)

(section) Input:

- a TickTock class instance

(section) Returns:

- GPS (numpy array) : elapsed secs since 6Jan1980 (excludes leap secs)

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.GPS
array([])
```

(section) See Also:

getUTC, getUNX, getRDT, getJD, getMJD, getCDF, getISO, getDOY, geteDOY

(section) Author:

Brian Larsen, Los Alamos National Lab (balarsen@lanl.gov)

(section) Version:

V1: 20-Jan-2010 (BAL)

getTAI(*self*)

a.TAI or a.getTAI()

return TAI (International Atomic Time)

(section) Input:

- a TickTock class instance

(section) Returns:

- TAI (numpy array) : elapsed secs since 1958/1/1 (includes leap secs, i.e. all secs have equal lengths)

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.TAI
array([1391342432])
```

(section) See Also:

getUTC, getUNX, getRDT, getJD, getMJD, getCDF, getISO, getDOY, geteDOY

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010 (JK) V2: 25-Jan-2010: include array support (JK)

getISO(*self*)

a.ISO or a.getISO()

convert dtype data into ISO string

(section) Input:

- a TickTock class instance

(section) Returns:

- ISO (list of strings) : date in ISO format

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.ISO
['2002-02-02T12:00:00']
```

(section) See Also:

getUTC, getUNX, getRDT, getJD, getMJD, getCDF, getTAI, getDOY, geteDOY

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010 (JK) V2: 25-Jan-2010: included arary support (JK) V3: 10-May-2010: speedup, arange to xrange (BAL)

getleapsecs(*self*)

a.leaps or a.getleapsecs()

retrieve leapseconds from lookup table, used in getTAI

(section) Input:

- a TickTock class instance

(section) Returns:

- secs (numpy array) : leap seconds

(section) Example:

```
>>> a = TickTock('2002-02-02T12:00:00', 'ISO')
>>> a.leaps
array([32])
```

(section) See Also:

getTAI

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010: includes array support (JK)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

12.5.2 Properties

Name	Description
Inherited from object	
__class__	

13 Module spacepy.testing

test suite for spacepy

Version: $Revision: 1.1 $, $Date: 2010/03/06 00:15:57 $

Author: Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

13.1 Functions

all()

test all spacepy routines

(section) Returns:

- nFAIL (int) : number of failures

(section) Example:

```
>>> all()
test_ticktock: PASSED TEST 1
test_ticktock: PASSED TEST 2
0
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 24-Jan-2010

ticktock()

test all time conversions

(section) Returns:

- nFAIL (int) : number of failures

(section) Example:

```
>>> ticktock()
testing ticktock: PASSED TEST 1
testing ticktock: PASSED TEST 2
0
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010

toolbox()

test all spacepy routines

(section) Returns:

- nFAIL (int) : number of failures

(section) Example:

```
>>> toolbox()
test_ticktock: PASSED TEST 1
test_ticktock: PASSED TEST 2
0
```

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 24-Jan-2010

13.2 Variables

Name	Description
__log__	**Value:** `'\n$Log:  testing.py,v $\nRevision 1.1 2010/03/06 00:15:5...`
__package__	**Value:** `None`

14 Module spacepy.toolbox

Toolbox of various functions and generic utilities.

Version: $Revision: 1.43 $, $Date: 2010/05/17 20:20:54 $

Author: S. Morley and J. Koller

14.1 Functions

doy2md(*year*, *doy*)

Convert day-of-year to month and day

(section) Inputs:

Year, Day of year (Jan 1 = 001)

(section) Returns:

Month, Day (e.g. Oct 11 = 10, 11)

Note: Implements full and correct leap year rules.

(section) Author:

Steve Morley, Los Alamos National Lab,
smorley@lanl.gov/morley_steve@hotmail.com

Modification history: ===================== Created by Steve Morley in July '05, rewritten for Python in October 2009

sec2hms(*sec*, *rounding*=True, *days*=False, *dtobj*=False)

```
Convert seconds of day to hours, minutes, seconds

Inputs:
=======
Seconds of day
Keyword arguments:
    rounding (True|False) - set for integer seconds
    days (True|False) - set to wrap around day (i.e. modulo 86400)
    dtobj (True|False) - set to return a timedelta object

Returns:
========
[hours, minutes, seconds] or datetime.timedelta

Author:
=======
Steve Morley, Los Alamos National Lab, smorley@lanl.gov/morley_steve@hotmail.com

Modification history:
====================
v1. Created by Steve Morley in March 2010
v1.1 Datetime timedelta output added; 17-May-2010 (SM)
```

tOverlap(*ts1*, *ts2*)

Finds the overlapping elements in two lists of datetime objects

(section) Returns:

- indices of 1 within interval of 2, & vice versa

(section) Example:

- Given two series of datetime objects, event_dates and omni['Time']:

```
>>> import spacepy.toolbox as tb
>>> [einds,oinds] = tb.tOverlap(event_dates, omni['Time'])
>>> omni_time = omni['Time'][oinds[0]:oinds[-1]+1]
>>> print omni_time
[datetime.datetime(2007, 5, 5, 17, 57, 30), datetime.datetime(2007, 5, 5, 18, 2, 30),
... , datetime.datetime(2007, 5, 10, 4, 57, 30)]
```

(section) Author:

Steve Morley, Los Alamos National Lab,
smorley@lanl.gov/morley_steve@hotmail.com

(section) Modifications:

Apr-2010: Add sanity check so when there are no overlapping elements, returns tuple of Nones Apr-2010: Additional sanity check for data equidistant from st and end pts

(section) Notes:

Can probably be rewritten to explicitly use datetime objects and keep method...

tCommon(*ts1*, *ts2*, *mask_only*=`True`)

Finds the elements in a list of datetime objects present in another

(section) Returns:

- Two element tuple of truth tables (of 1 present in 2, & vice versa)

(section) Author:

Steve Morley, Los Alamos National Lab,
smorley@lanl.gov/morley_steve@hotmail.com

loadpickle(*fln*)

load a pickle and return content as dictionary

(section) Input:

- fln (string) : filename

(section) Returns:

- d (dictionary) : dictionary with content from file

(section) Example:

```
>>> d = loadpickle('test.pbin')
```

(section) See also:

savepickle

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010

savepickle(*fln*, *dict*)

save dictionary variable dict to a pickle with filename fln Author: Josef Koller, jkoller@lanl.gov

(section) Inputs:

- fln (string) : filename
- dict (dictionary) : container with stuff

(section) Example:

```
>>> d = {'grade':[1,2,3], 'name':['Mary', 'John', 'Chris']}
>>> savepickle('test.pbin', d)
```

(section) See also:

loadpickle

(section) Author:

Josef Koller, Los Alamos National Lab (jkoller@lanl.gov)

(section) Version:

V1: 20-Jan-2010

assemble(*fln_pattern*, *outfln*)

assembles all pickled files matching fln_pattern into single file and save as outfln. Pattern may contain simple shell-style wildcards *? a la fnmatch file will be assembled along time axis TAI, etc in dictionary

(section) Inputs:

- fln_pattern (string) : pattern to match filenames
- outfln (string) : filename to save combined files to

(section) Outputs:

- dcomb (dict) : dictionary with combined values

(section) Example:

```
>>> assemble('input_files_*.pbin', 'combined_input.pbin')
adding input_files_2001.pbin
adding input_files_2002.pbin
adding input_files_2004.pbin
writing: combined_input.pbin
```

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 20-Jan-2010

feq(*x*, *y*)

compare two floating point values if they are equal after:
http://www.lahey.com/float.htm

further info at:

```
http://docs.python.org/tut/node16.html
http://www.velocityreviews.com/forums/t351983-precision-for-equality-of-two-float
http://www.boost.org/libs/test/doc/components/test_tools/floating_point_compariso
http://howto.wikia.com/wiki/Howto_compare_float_numbers_in_the_C_programming_langua
```

(section) Input:

- x (float) : a number
- y (array of floats) : float value or array of floats

(section) Returns:

- boolean value: True or False

(section) Example:

```
>>> index = where( feq(Lpos,Lgrid) ) # use float point comparison
```

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 20-Jan-2010

dictree(*in_dict*, *verbose*=`False`, *spaces*=`None`, *levels*=`True`)

pretty print a dictionary tree

(section) Input:

- in_dict (dictionary) : a complex dictionary (with substructures)
- spaces (string) : string will added for every line
- levels (default False) : number of levels to recurse through (True means all)
- boolean value: True or False

(section) Example:

```
>>> d = {'grade':{'level1':[4,5,6], 'level2':[2,3,4]}, 'name':['Mary', 'John', 'Chris']}
>>> dictree(d)
+
|____grade
    |____level1
    |____level2
|____name
```

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 20-Jan-2010 V1.1: 24-Feb-2010 S. Morley, added verbose option v1.2: 17-May-2010 S. Morley, added levels option

printfig(*fignum*, *saveonly*=`False`, *pngonly*=`False`, *clean*=`False`)

save current figure to file and call lpr (print).

This routine will create a total of 3 files (png, ps and c.png) in the current working directory with a sequence number attached. Also, a time stamp and the location of the file will be imprinted on the figure. The file ending with c.png is clean and no directory or time stamp are attached (good for powerpoint presentations).

(section) Input:

- fignum (integer or array/list of integer) : matplotlib figure number
- optional
  - saveonly (boolean) : True (don't print and save only to file) False (print and save)
  - pngonly (boolean) : True (only save png files and print png directly) False (print ps file, and generate png, ps; can be slow)
  - clean (boolean) : True (print and save only clean files without directory info) False (print and save directory location as well)

(section) Example:

```
>>> pylab.plot([1,2,3],[2,3,2])
>>> spacepy.printfig(1)
```

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 20-Jan-2010 V2: 19-Feb-2010: added pngonly and clean options, array/list support (JK)

update(*all*=`True`, *omni*=`False`, *leapsecs*=`False`)

Download and update local database for omni, leapsecs etc

(section) Input:

- all (bol) : if True, update all of them
- omni (bol) : if True. update only onmi
- leapsecs (bol) : if True, update only leapseconds

(section) Example:

```
>>> update(omni=True)
```

(section) Author:

Josef Koller, Los Alamos National Lab, jkoller@lanl.gov

(section) Version:

V1: 20-Jan-2010

windowMean(*data*, *time*=`[]`, *winsize*=`0`, *overlap*=`0`, *st_time*=`None`)

Windowing mean function, window overlap is user defined

Inputs: data - 1D series of points; time - series of timestamps, optional (format as numeric or datetime); For non-overlapping windows set overlap to zero. e.g.,

```
>>> wsize, olap = datetime.timedelta(1), datetime.timedelta(0,3600)
```

```
>>> outdata, outtime = windowmean(data, time, winsize=wsize, overlap=olap)
```

where the time, winsize and overlap are either numberic or datetime objects, in this example the window size is 1 day and the overlap is 1 hour.

Caveats: This is a quick and dirty function - it is NOT optimised, at all.

(section) Author:

Steve Morley, Los Alamos National Lab, smorley@lanl.gov/morley_steve@hotmail.com

medAbsDev(*series*)

Calculate median absolute deviation of a given input series

Median absolute deviation (MAD) is a robust and resistant measure of the spread of a sample (same purpose as standard deviation). The MAD is preferred to the interquartile range as the interquartile range only shows 50% of the data whereas the MAD uses all data but remains robust and resistant. See e.g. Wilks, Statistical methods for the Atmospheric Sciences, 1995, Ch. 3.

This implementation is robust to presence of NaNs

Example: Find the median absolute deviation of a data set. Here we use the log- normal distribution fitted to the population of sawtooth intervals, see Morley and Henderson, Comment, Geophysical Research Letters, 2009.

```
>>> data = numpy.random.lognormal(mean=5.1458, sigma=0.302313, size=30)
>>> print data
array([ 181.28078923,  131.18152745, ... , 141.15455416, 160.88972791])
>>> utils.medabsdev(data)
28.346646721370192
```

(section) Author:

Steve Morley, Los Alamos National Lab, smorley@lanl.gov/morley_steve@hotmail.com

makePoly(*x*, *y1*, *y2*, *face*='blue', *alpha*=0.5)

Make filled polygon for plotting

Equivalent functionality to built-in matplotlib function fill_between

```
>>> poly0c = makePoly(x, ci_low, ci_high, face='red', alpha=0.8)
>>> ax0.add_patch(poly0qc)
```

(section) Author:

Steve Morley, Los Alamos National Lab, smorley@lanl.gov/morley_steve@hotmail.com

binHisto(*data*)

Calculates bin width and number of bins for histogram using Freedman-Diaconis rule

(section) Inputs:

data - list/array of data values

(section) Outputs:

binw - calculated width of bins using F-D rule nbins - number of bins (nearest integer) to use for histogram

(section) Example:

```
>>> import numpy, spacepy
>>> import matplotlib.pyplot as plt
>>> data = numpy.random.randn(100)
>>> binw, nbins = spacepy.toolbox.binHisto(data)
>>> plt.hist(data, bins=nbins, histtype='step', normed=True)
```

(section) Author:

Steve Morley, Los Alamos National Lab, smorley@lanl.gov/morley_steve@hotmail.com

smartTimeTicks(*time*)

Returns major ticks, minor ticks and format for time-based plots

smartTimeTicks takes a list of datetime objects and uses the range to calculate the best tick spacing and format.

(section) Inputs:

time - list of datetime objects

(section) Outputs:

Mtick - major ticks mtick - minor ticks fmt - format

(section) Example:

? Meh.

(section) Author:

Dan Welling, Los Alamos National Lab, dwelling@lanl.gov/dantwelling@gmail.com

t_common(*ts1*, *ts2*, *mask_only*=`True`)

Finds the elements in a list of datetime objects present in another

(section) Returns:

- Two element tuple of truth tables (of 1 present in 2, & vice versa)

(section) Author:

Steve Morley, Los Alamos National Lab, smorley@lanl.gov/morley_steve@hotmail.com

t_overlap(*ts1*, *ts2*)

Finds the overlapping elements in two lists of datetime objects

(section) Returns:

- indices of 1 within interval of 2, & vice versa

(section) Example:

- Given two series of datetime objects, event_dates and omni['Time']:

```
>>> import spacepy.toolbox as tb
>>> [einds,oinds] = tb.tOverlap(event_dates, omni['Time'])
>>> omni_time = omni['Time'][oinds[0]:oinds[-1]+1]
>>> print omni_time
[datetime.datetime(2007, 5, 5, 17, 57, 30), datetime.datetime(2007, 5, 5, 18, 2, 30),
... , datetime.datetime(2007, 5, 10, 4, 57, 30)]
```

(section) Author:

Steve Morley, Los Alamos National Lab,
smorley@lanl.gov/morley_steve@hotmail.com

(section) Modifications:

Apr-2010: Add sanity check so when there are no overlapping elements, returns tuple of Nones Apr-2010: Additional sanity check for data equidistant from st and end pts

(section) Notes:

Can probably be rewritten to explicitly use datetime objects and keep method...

smart_timeticks(*time*)

Returns major ticks, minor ticks and format for time-based plots

smartTimeTicks takes a list of datetime objects and uses the range to calculate the best tick spacing and format.

(section) Inputs:

time - list of datetime objects

(section) Outputs:

Mtick - major ticks mtick - minor ticks fmt - format

(section) Example:

? Meh.

(section) Author:

Dan Welling, Los Alamos National Lab, dwelling@lanl.gov/dantwelling@gmail.com

logspace(*min*, *max*, *num*, ***kwargs*)

Returns log spaced bins. Same as numpy logspace except the min and max are the ,min and max not log10(min) and log10(max)

logspace(min, max, num)

(section) Inputs:

min - minimum value max - maximum value num - number of log spaced bins

(section) Outputs:

num log spaced bins from min to max in a numpy array

(section) Example:

logspace(1, 100, 5) Out[2]: array([1. , 3.16227766, 10. , 31.6227766 , 100.])

(section) Author:

Brian Larsen, Los Alamos National Lab, balarsen@lanl.gov

arraybin(*array*, *bins*)

```
Given an array and a set of bins return the indices that are less than the
smallest bin between each set and larger than the largest bin
bins should be sorted

Inputs:
=======
array - the input array to slice
bins - the bins to slice along (may be array or list)

Outputs:
========
list of indices
     first element is less than fisrt bin and last bin is larger than last bin

Example:
========
arraybin(arange(10), [4.2])
Out[4]: [(array([0, 1, 2, 3, 4]),), (array([5, 6, 7, 8, 9]),)]

Author:
=======
Brian Larsen, Los Alamos National Lab, balarsen@lanl.gov
```

mlt2rad(*mlt*, *midnight*=`False`)

Convert mlt values to radians for polar plotting transform mlt angles to radians from -pi to pi referenced from noon by default

(section) Inputs:

mlt - array of mlt values midnight=False - reference to midnioght instead of noon

(section) Outputs:

array of radians

(section) Example:

mlt2rad(array([3,6,9,14,22])) Out[9]: array([-2.35619449, -1.57079633, -0.78539816, 0.52359878, 2.61799388])

(section) Author:

Brian Larsen, Los Alamos National Lab, balarsen@lanl.gov

rad2mlt(*rad*, *midnight*=`False`)

Convert radian values to mlt transform radians from -pi to pi to mlt referenced from noon by default

(section) Inputs:

rad - array of rad values midnight=False - reference to midnioght instead of noon

(section) Outputs:

array of mlt

(section) Example:

rad2mlt(array([0,pi, pi/2.])) Out[8]: array([12., 24., 18.])

(section) Author:

Brian Larsen, Los Alamos National Lab, balarsen@lanl.gov

leap_year(*year*, *numdays*=False)

```
return an array of boolean leap year,
a lot faster than the mod method that is normally seen

Inputs:
=======
year - array of years
numdays=False - optionally return the number of days in the year

Outputs:
========
an array of boolean leap year, or array of number of days

Example:
========
leap_year(arange(15)+1998)
Out[10]:
array([False, False,  True, False, False, False,  True, False, False,
   False,  True, False, False, False,  True], dtype=bool)

Author:
=======
Brian Larsen, Los Alamos National Lab, balarsen@lanl.gov
```

pmm(*a*, **b*)

print min and max of input arrays

(section) Inputs:

a - input array *b - some additional number of arrays

(section) Outputs:

list of min, max for each array

(section) Example:

pmm(arange(10), arange(10)+3) Out[12]: [(0, 9), (3, 12)]

(section) Author:

Brian Larsen, Los Alamos National Lab, balarsen@lanl.gov

timestamp(*position*=`1.=...`, *0.01*=`...`, *size*=`'xx-small'`, ***kwargs*)

print a timestamp on the current plot, vertical lower right

(section) Inputs:

(all optional) position - position for the timestamp size - text size draw - call draw to make sure it appears kwargs - other keywords to axis.annotate

(section) Outputs:

timestamp written to the current plot

(section) Example:

plot(arange(11)) Out[13]: [<matplotlib.lines.Line2D object at 0x49072b0>] timestamp()

(section) Author:

Brian Larsen, Los Alamos National Lab, balarsen@lanl.gov

query_yes_no(*question*, *default*='yes')

```
Ask a yes/no question via raw_input() and return their answer.

"question" is a string that is presented to the user.
"default" is the presumed answer if the user just hits <Enter>.
    It must be "yes" (the default), "no" or None (meaning
    an answer is required of the user).

The "answer" return value is one of "yes" or "no".

Inputs:
=======
question - string that is the question to ask
default - the default answer (yes)

Outputs:
========
answer ('yes' or 'no')

Example:
========
query_yes_no('Ready to go?')
Ready to go? [Y/n] y
Out[17]: 'yes'


Author:
=======
Brian Larsen, Los Alamos National Lab, balarsen@lanl.gov
```

14.2 Variables

Name	Description
__log__	**Value:** `'\n$Log:  toolbox.py,v $\nRevision 1.43 2010/05/17 20:20:...`
__package__	**Value:** `'spacepy'`

15 Module spacepy.utils

Set of generic utilities.

–++– By Steve Morley –++–

smorley@lanl.gov/morley_steve@hotmail.com, Los Alamos National Laboratory, ISR-1, PO Box 1663, Los Alamos, NM 87545

15.1 Functions

doy2md(*year*, *doy*)

Convert day-of-year to month and day

(section) Inputs:

Year, Day of year (Jan 1 = 001)

(section) Returns:

Month, Day (e.g. Oct 11 = 10, 11)

Note: Implements full and correct leap year rules.

Modification history: Created by Steve Morley (in July '05), rewritten for Python in October 2009

t_overlap(*ts1*, *ts2*)

Finds the overlapping elements in two lists of datetime objects

(section) Returns:

- indices of 1 within interval of 2, & vice versa

(section) Example:

- Given two series of datetime objects, event_dates and omni['Time']:

```
>>> import spacepy.utils as utils
>>> [einds,oinds] = utils.t_overlap(event_dates, omni['Time'])
>>> omni_time = omni['Time'][oinds[0]:oinds[-1]+1]
>>> print omni_time
[datetime.datetime(2007, 5, 5, 17, 57, 30), datetime.datetime(2007, 5, 5, 18, 2, 30),
... , datetime.datetime(2007, 5, 10, 4, 57, 30)]
```

t_common(*ts1*, *ts2*, *mask_only*=`True`)

Finds the elements in a list of datetime objects present in another

(section) Returns:

- Two element tuple of truth tables (of 1 present in 2, & vice versa)

ShueMP(*P*, *Bz*)

Calculates the Shue et al. (1997) subsolar magnetopause radius

Ported from Drew Turner's (LASP) MatLab script

(section) Inputs:

SW ram pressure [nPa], IMF Bz (GSM) [nT]

(section) Output:

Magnetopause (sub-solar point) standoff distance [Re]

MoldwinPP(*Kp*)

Calculates the Moldwin et al. (2002) subsolar magnetopause radius

Inputs: Kp index (uses max Kp from prev 12 hrs) Output: Plasmapause radius [Re]

readOMNI(*fname*)

Read amalgamated OMNI2 file into dictionary of data arrays

Input: filename (str)

Output: dictionary with datetime objects and data arrays

readOMNIhi(*fname*)

Read high-resOMNI file into dictionary of data arrays

Input: filename (str)

Output: dictionary with datetime objects and data arrays

mu_ai(*energy*, *b*=`1e-07`, *rme*=`0.511`)

Calculate 1st adiabatic invariant given energy in [MeV/G]

Input: energy (req'd) in MeV

Uses E = Ek + E0, E = sqrt(p^2c^2 + E0^2); Then uses mu = p^2/2mB to arrive at mu = (p^2c^2)/(2*E0*B)

dm_ll(*kp*=`2.7`, *l*=`6.6`)

Magnetic field diffusion coefficient from Brautigam and Albert

windowmean(*data*, *time*=`[]`, *winsize*=`0`, *overlap*=`0`, *pts*=`True`)

Windowing mean function, window overlap is user defined

Inputs: data - 1D series of points; time - series of timestamps, optional (format as numeric or datetime); For non-overlapping windows set overlap to zero. e.g.,

```
>>> wsize, olap = datetime.timedelta(1), datetime.timedelta(0,3600)
```

```
>>> outtime, outdata = windowmean(data, time, winsize=wsize, overlap=olap)
```

where the time, winsize and overlap are either numberic or datetime objects, in this example the window size is 1 day and the overlap is 1 hour.

Caveats: This is a quick and dirty function - it is NOT optimised, at all.

medabsdev(*series*)

Calculate median absolute deviation of a given input series

Median absolute deviation (MAD) is a robust and resistant measure of the spread of a sample (same purpose as standard deviation). The MAD is preferred to the interquartile range as the interquartile range only shows 50% of the data whereas the MAD uses all data but remains robust and resistant. See e.g. Wilks, Statistical methods for the Atmospheric Sciences, 1995, Ch. 3.

This implementation is robust to presence of NaNs

Example: Find the median absolute deviation of a data set. Here we use the log- normal distribution fitted to the population of sawtooth intervals, see Morley and Henderson, Comment, Geophysical Research Letters, 2009.

```
>>> data = numpy.random.lognormal(mean=5.1458, sigma=0.302313, size=30)
>>> print data
array([ 181.28078923,  131.18152745, ... , 141.15455416, 160.88972791])
>>> utils.medabsdev(data)
28.346646721370192
```

makePoly(*x*, *y1*, *y2*, *face*=`'blue'`, *alpha*=`0.5`)

Make filled polygon for plotting

Can be replaced by built-in matplotlib function fill_between

```
>>> poly0c = makePoly(x, ci_low, ci_high, face='red', alpha=0.8)
>>> ax0.add_patch(poly0qc)
```

binHisto(*data*)

Calculates bin width and number of bins for histogram using Freedman-Diaconis rule

15.2 Variables

Name	Description
__package__	**Value:** `'spacepy'`

Index

spacepy *(package)*, 2–3
  spacepy.constants *(module)*, 4
    spacepy.constants.eIon *(function)*, 4
    spacepy.constants.vAlfven *(function)*, 4
    spacepy.constants.vIon *(function)*, 4
  spacepy.empiricals *(module)*, 5
    spacepy.empiricals.get_Lmax *(function)*, 5
    spacepy.empiricals.get_plasma_pause *(function)*, 5
    spacepy.empiricals.ShueMP *(function)*, 5
  spacepy.omni *(module)*, 6–7
    spacepy.omni.getG123 *(function)*, 7
    spacepy.omni.getomni *(function)*, 6
    spacepy.omni.pickleomni *(function)*, 6
  spacepy.onerapy *(module)*, 8–17
    spacepy.onerapy.car2sph *(function)*, 11
    spacepy.onerapy.coord_trans *(function)*, 10
    spacepy.onerapy.find_Bmirror *(function)*, 8
    spacepy.onerapy.find_magequator *(function)*, 9
    spacepy.onerapy.get_Bfield *(function)*, 8
    spacepy.onerapy.get_dtype *(function)*, 14
    spacepy.onerapy.get_Lstar *(function)*, 15
    spacepy.onerapy.get_sysaxes *(function)*, 13
    spacepy.onerapy.prep_onera *(function)*, 16
    spacepy.onerapy.sph2car *(function)*, 12
  spacepy.onerapylib *(module)*, 18
  spacepy.poppy *(module)*, 19–22
    spacepy.poppy.boots_ci *(function)*, 20
    spacepy.poppy.PPro *(class)*, 20–22
  spacepy.radbelt *(module)*, 23–26
    spacepy.radbelt.diff_LL *(function)*, 23
    spacepy.radbelt.get_DLL *(function)*, 23
    spacepy.radbelt.get_local_accel *(function)*, 23
    spacepy.radbelt.get_modelop_L *(function)*, 23
    spacepy.radbelt.RBmodel *(class)*, 23–26
  spacepy.satdata *(module)*, 27–28
    spacepy.satdata.convert2PSDclass *(function)*, 27
    spacepy.satdata.get_obs *(function)*, 27
    spacepy.satdata.PSDdataClass *(class)*, 27–28
    spacepy.satdata.updatePSDdb *(function)*, 27
  spacepy.seapy *(module)*, 29–35
    spacepy.seapy.multisea *(function)*, 29
    spacepy.seapy.readepochs *(function)*, 30
    spacepy.seapy.Sea *(class)*, 30–32
    spacepy.seapy.Sea2d *(class)*, 32–35
    spacepy.seapy.seadict *(function)*, 29
  spacepy.setup *(module)*, 36
    spacepy.setup.compile_oneralib *(function)*, 36
    spacepy.setup.subst *(function)*, 36
  spacepy.spacetime *(module)*, 37–73
    spacepy.spacetime.doy2date *(function)*, 37
    spacepy.spacetime.SpaCo *(class)*, 39–46
    spacepy.spacetime.TickDelta *(class)*, 46–49
    spacepy.spacetime.tickrange *(function)*, 37
    spacepy.spacetime.TickTock *(class)*, 49–73
  spacepy.testing *(module)*, 74–76
    spacepy.testing.all *(function)*, 74
    spacepy.testing.ticktock *(function)*, 74
    spacepy.testing.toolbox *(function)*, 75
  spacepy.toolbox *(module)*, 77–96
    spacepy.toolbox.arraybin *(function)*, 91
    spacepy.toolbox.assemble *(function)*, 81
    spacepy.toolbox.binHisto *(function)*, 87
    spacepy.toolbox.dictree *(function)*, 83
    spacepy.toolbox.doy2md *(function)*, 77
    spacepy.toolbox.feq *(function)*, 82
    spacepy.toolbox.leap_year *(function)*, 93

spacepy.toolbox.loadpickle *(function)*, 79
spacepy.toolbox.logspace *(function)*, 91
spacepy.toolbox.makePoly *(function)*, 87
spacepy.toolbox.medAbsDev *(function)*, 86
spacepy.toolbox.mlt2rad *(function)*, 92
spacepy.toolbox.pmm *(function)*, 94
spacepy.toolbox.printfig *(function)*, 84
spacepy.toolbox.query_yes_no *(function)*, 95
spacepy.toolbox.rad2mlt *(function)*, 93
spacepy.toolbox.savepickle *(function)*, 80
spacepy.toolbox.sec2hms *(function)*, 77
spacepy.toolbox.smartTimeTicks *(function)*, 88, 90
spacepy.toolbox.tCommon *(function)*, 79, 89
spacepy.toolbox.timestamp *(function)*, 94
spacepy.toolbox.tOverlap *(function)*, 78, 89
spacepy.toolbox.update *(function)*, 85
spacepy.toolbox.windowMean *(function)*, 86
spacepy.utils *(module)*, 97–100
spacepy.utils.binHisto *(function)*, 100
spacepy.utils.dm_ll *(function)*, 99
spacepy.utils.doy2md *(function)*, 97
spacepy.utils.makePoly *(function)*, 100
spacepy.utils.medabsdev *(function)*, 99
spacepy.utils.MoldwinPP *(function)*, 98
spacepy.utils.mu_ai *(function)*, 98
spacepy.utils.readOMNI *(function)*, 98
spacepy.utils.readOMNIhi *(function)*, 98
spacepy.utils.ShueMP *(function)*, 98
spacepy.utils.t_common *(function)*, 97
spacepy.utils.t_overlap *(function)*, 97
spacepy.utils.windowmean *(function)*, 99