OpATOM - An Open Source Toolbox for Atmospheric Tomography

Generated by Doxygen 1.8.17

Chapter 1

OpATOM - An Open Source Toolbox for Atmospheric Tomography

1.1 Description

OpAtom toolbox provides a tomographic algorithm that capable of estimate a 3D wet refractivity model in Eastern Europe. The sizes of the tomographic grid are specified in the input files listed below. This algorithm uses an approximate cartesian reference system in which the length of the rays can be easily calculated. This Cartesian reference system is defined in the getlocal.py file and must be modified in case it is used in another area.

1.2 Module Requirements

The toolbox has been tested on Ubuntu 20.04 using python 3.8. Module dependencies:

- NumPy
- SciPy
- Wget
- Matplotlib

1.3 Usage

```
gnssct.py [OPTION]
-s, --satellites   location of the satellite orbits file in .SP3 format
-S, --stations     location of the station coordinates file in Bernese .CRD format
    --gridp        location of the grid file in North-South direvtion in .csv format (degrees)
    --gridl        location of the grid file in East-West direction in .csv format (degrees)
    --gridh        location of the elevation grid file .csv format (metres)
-v, --vmf1loc      location of the VMF1 parameters grid files directory
-i, --initial_w    location of the initial wet refractivity values in .csv format
-e, --epoch        epoch in format YYYY-MM-DD-hh-mm-ss
Example:
python3 gnssct.py --satellites=./sample_data/orbit/CDU23005_00.EPH --
stations=./sample_data/METEONET.CRD --tropofile=./sample_data/TRP/CO24040C.TRP --
gridp=./sample_data/gridp.csv --gridl=./sample_data/gridl.csv --
gridh=./sample_data/gridh.csv --vmf1loc=./sample_data/vmf1/ --epoch=2024-2-9-2-0-0 --
initial_w=./sample_data/raobs/files/12843_2024-2-8_11.csv
```

The VMF1 parameters grid files must be placed in this directory, and the name format must be: YYYY/VMFG_Y↩
YYYMMDD.Hhh

1.4 Input files

For the tomographic processing, the following input files are required:

- The tomographic grid file (csv format)
  - Latitude
  - Longitude
  - Height
- GNSS station coordinates file (Bernese CRD format)
- Tropospheric delays file (Bernese TRP format)
- VMF1 grid parameters file (VMF1 grid file)
- Satellite orbit file (SP3 format)
- Initial wet refractivity values (csv format)

1.5 Tomographic grid files

Tomographic grid files define the size of the cells in each direction (latitude, longitude, height borders) over the entire area. Each file is a list of coordinates. In the case of latitude and longitude, the script expects the coordinates in degrees (WGS84), and the heights to be in meters.

```
45.5
46.2
46.9
47.6
48.3
49.0
49.7
```

1.6 GNSS station coordinates file

The GNSS station coordinates file contains all the GNSS stations and their coordinates for the given epoch in Bernese CRD format.

```
Weekly solution for Week 2310                                     04-FEB-24 05:50
--------------------------------------------------------------------------------
LOCAL GEODETIC DATUM: IGS14            EPOCH: 2024-01-31 12:00:00
NUM  STATION NAME           X (M)          Y (M)          Z (M)     FLAG
  1  BAIA              3945839.43919  1720428.58296  4691082.90436    A
143  BAJ1              4183093.74170  1439191.16597  4579512.35582    A
  2  BAJA              4183094.39352  1439190.59467  4579511.94882
140  BARA              3805783.52640  1629895.39810  4835969.94890
  4  BBYS              3980358.47759  1382292.41144  4772772.14404    A
```

1.7 Tropospheric delay file

For calculating Slant Wet Delay (SWD) values, the Zenith Wet Delays (ZWD) and Tropospheric Gradient Values are required for each station. These files must be in Bernese TRP format, where ZWDs are in column CORR_U and Tropospheric gradients are in CORR_E and CORR_N.

```
                                                                 09-FEB-24 02:42
------------------------------------------------------------------------------------------------------------------------
 A PRIORI MODEL:  -17   MAPPING FUNCTION:    8   GRADIENT MODEL:    4   MIN. ELEVATION:    5   TABULAR
       INTERVAL:  3600 / 86400
 STATION NAME      FLG   YYYY MM DD HH MM SS   YYYY MM DD HH MM SS   MOD_U   CORR_U  SIGMA_U TOTAL_U  CORR_N
       SIGMA_N  CORR_E  SIGMA_E
```

```
 BAIA            A    2024 02 08 13 00 00                         2.2278  0.09318 0.00093 2.32095 -0.00005
      0.00007 -0.00068 0.00007
 BAIA            A    2024 02 08 14 00 00                         2.2278  0.10194 0.00059 2.32972 -0.00008
      0.00006 -0.00068 0.00006
 BAIA            A    2024 02 08 15 00 00                         2.2278  0.09959 0.00062 2.32738 -0.00010
      0.00005 -0.00067 0.00006
 BAIA            A    2024 02 08 16 00 00                         2.2278  0.10264 0.00064 2.33044 -0.00012
      0.00004 -0.00067 0.00005
 BAIA            A    2024 02 08 17 00 00                         2.2278  0.10729 0.00050 2.33509 -0.00015
      0.00004 -0.00066 0.00004
 BAIA            A    2024 02 08 18 00 00                         2.2278  0.10523 0.00063 2.33304 -0.00017
      0.00003 -0.00065 0.00004
 BAIA            A    2024 02 08 19 00 00                         2.2271  0.11471 0.00059 2.34182 -0.00019
      0.00004 -0.00065 0.00004
 BAIA            A    2024 02 08 20 00 00                         2.2264  0.11173 0.00053 2.33814 -0.00022
      0.00004 -0.00064 0.00005
 BAIA            A    2024 02 08 21 00 00                         2.2257  0.12048 0.00065 2.34619 -0.00024
      0.00005 -0.00063 0.00005
 BAIA            A    2024 02 08 22 00 00                         2.2250  0.11663 0.00057 2.34164 -0.00026
      0.00005 -0.00063 0.00006
 BAIA            A    2024 02 08 23 00 00                         2.2243  0.11885 0.00075 2.34317 -0.00029
      0.00006 -0.00062 0.00007
 BAIA            A    2024 02 09 00 00 00                         2.2236  0.11231 0.00087 2.33593 -0.00031
      0.00007 -0.00061 0.00008
 BAIA            A    2024 02 09 01 00 00                         2.2236  0.11582 0.00167 2.33945  0.00096
      0.00017 -0.00177 0.00028
 BAJ1            A    2024 02 08 13 00 00                         2.2590  0.10221 0.00085 2.36121 -0.00009
      0.00006 -0.00084 0.00006
 BAJ1            A    2024 02 08 14 00 00                         2.2587  0.10047 0.00055 2.35917 -0.00010
      0.00005 -0.00075 0.00005
 BAJ1            A    2024 02 08 15 00 00                         2.2584  0.10198 0.00058 2.36038 -0.00012
      0.00005 -0.00066 0.00004
 BAJ1            A    2024 02 08 16 00 00                         2.2581  0.09861 0.00056 2.35672 -0.00013
      0.00004 -0.00057 0.00004
 BAJ1            A    2024 02 08 17 00 00                         2.2578  0.09972 0.00045 2.35753 -0.00014
      0.00003 -0.00048 0.00003
 BAJ1            A    2024 02 08 18 00 00                         2.2575  0.09471 0.00058 2.35222 -0.00016
      0.00003 -0.00039 0.00003
 BAJ1            A    2024 02 08 19 00 00                         2.2572  0.09693 0.00050 2.35413 -0.00017
      0.00003 -0.00030 0.00003
 BAJ1            A    2024 02 08 20 00 00                         2.2569  0.09271 0.00049 2.34961 -0.00018
      0.00003 -0.00021 0.00003
 BAJ1            A    2024 02 08 21 00 00                         2.2566  0.09063 0.00060 2.34723 -0.00020
      0.00004 -0.00012 0.00004
...
```

1.8 VMF1 grid files

The calculation of the SWDs requires a mapping function. For this purpose, the script uses the VMF1, which needs the aw coefficients. These coefficients are available on the website of the Vienna University of Technology. These parameters are provided in grid files for every 6 hours. For the hourly interpolation in time, the script expects two files.

```
! Version:            1.0
! Source:             J. Boehm, TU Vienna (created: 2024-02-14)
! Data_types:         VMF1 (lat lon ah aw zhd zwd)
! Epoch:              2024 02 15 00 00  0.0
! Scale_factor:       1.e+00
! Range/resolution:   -90 90 0 360 2 2.5
! Comment:            http://vmf.geo.tuwien.ac.at/trop_products/GRID/2.5x2/VMF1/VMF1_OP/
 90.0   0.0 0.00117044  0.00060490  2.2998  0.0204
 90.0   2.5 0.00117044  0.00060490  2.2998  0.0204
 90.0   5.0 0.00117044  0.00060490  2.2998  0.0204
 90.0   7.5 0.00117044  0.00060490  2.2998  0.0204
 90.0  10.0 0.00117044  0.00060490  2.2998  0.0204
 90.0  12.5 0.00117044  0.00060490  2.2998  0.0204
 90.0  15.0 0.00117044  0.00060490  2.2998  0.0204
 90.0  17.5 0.00117044  0.00060490  2.2998  0.0204
 90.0  20.0 0.00117044  0.00060490  2.2998  0.0204
 90.0  22.5 0.00117044  0.00060490  2.2998  0.0204
 90.0  25.0 0.00117044  0.00060490  2.2998  0.0204
 90.0  27.5 0.00117044  0.00060490  2.2998  0.0204
. . .
```

1.9 Satellite orbit file

To calculate the azimuth and elevation angle from the station to the satellite, besides the station coordinates, the satellite orbits are also required in SP3 format. The ultra-rapid satellite orbits for GPS, GLONASS, and Galileo constellations are available from the Center for Orbit Determination in Europe at the University of Bern.

```
#cP2024  2 12 18  0  0.00000000     577 d+D   IGS20 EXT AIUB
## 2301 151200.00000000   300.00000000 60352 0.7500000000000
+   78   G01G02G03G04G05G06G07G08G09G10G11G12G13G14G15G16G17
+        G18G19G20G21G22G23G24G25G26G27G28G29G30G31G32R01R02
+        R03R04R05R07R08R09R11R12R13R14R15R16R17R18R19R20R21
+        R22R24E02E03E04E05E07E08E09E10E11E12E13E14E15E18E19
+        E21E24E25E26E27E30E31E33E34E36  0  0  0  0  0  0  0
++         5  7  6  6  7  6  7  6  6  6  7  7  6  6  7  7  7
++         7  7  7  7  7  6  6  7  6  6  6  7  5  7  6  9  7
++         8  8  8  8  8  7  6  7  7  6  6  6  8  7  9  9  8
++         7  8  6  6  6  6  7  6  7  7  7  6  6  7  7  7  6
++         6  6  6  7  6 10  6  7  6  7  0  0  0  0  0  0  0
%c M  cc GPS ccc cccc cccc cccc cccc ccccc ccccc ccccc ccccc
%c cc cc ccc ccc cccc cccc cccc cccc ccccc ccccc ccccc ccccc
%f  1.2500000  1.025000000  0.00000000000  0.000000000000000
%f  0.0000000  0.000000000  0.00000000000  0.000000000000000
%i    0    0    0    0      0      0      0      0         0
%i    0    0    0    0      0      0      0      0         0
/* Center for Orbit Determination in Europe (CODE)
/* Ultra-rapid GRE orbits starting year-day 24043 18 hour
/* Observed/predicted: 24/24 hours (data used up to 044R)
/* PCV:IGS20      OL/AL:FES2014b NONE     YN ORB:CoN CLK:BRD
*  2024  2 12 18  0  0.00000000
PG01  10017.227962 -21757.155189 -11451.757387    169.286245
PG02  14675.739771 -21822.976616  -2052.019267   -486.484832
PG03   8469.065183 -12995.901194 -21686.432325    188.867787
PG04   3600.602597 -22237.722088 -13945.053317    290.029824
PG05 -20341.276134   7377.183933  15289.811367   -161.559994
PG06 -16369.020521  -2296.830107 -20745.620735    409.693247
PG07  -1894.045106 -18515.401534  19259.776594    -60.352554
PG08   8973.397509 -15470.338889  19399.830924   -166.689337
PG09  -7116.144195 -25433.994890  -2585.513683     89.933932
PG10  22629.595338  10998.556186   9210.119432      0.062606
PG11 -21242.149685   8170.890518 -13627.298892   -573.517674
PG12  -9883.448912  12694.506995 -21410.063052   -477.323298
PG13 -13937.859196   5903.727931  21603.132559    624.913034
PG14 -19730.979185 -13446.930472  11852.906315    323.976361
PG15  -7222.733526  16617.677119  18862.775701    127.009782
PG16  24409.248046   -401.590168  10670.590254   -364.118237
. . .
```

1.10 Initial wet refractivity file

The initial values of the 3D Wet Refractivity model are necessary to solve the equation system with the MART algorithm. Radiosonde (RS) profiles are used to calculate these values, and these profiles are expanded to cover the entire area. After the calculation of the Wet refractivity values, they are stored in csv format (Fig7).

```
WMOID,HEIGHT,DATE,TIME,HEIGHT,N_DRY,N_WET,TEMPERATURE,PRESSURE,DEWPOINT,RHOWV
12843,139,2024-02-01,11:00:00,139,279.0515,31.90512,278.56,10080,273.56,0.004893853
12843,209,2024-02-01,11:00:00,209,278.1079,30.57998,277.36,10000,272.86,0.00467139
12843,250,2024-02-01,11:00:00,250,276.9208,30.4007,277.16,9950,272.76,0.00464082
12843,440,2024-02-01,11:00:00,440,267.8434,32.1,279.76,9720,273.76,0.004943895
12843,601,2024-02-01,11:00:00,601,262.9998,31.27059,279.36,9530,273.36,0.004809611
12843,846,2024-02-01,11:00:00,846,257.0624,31.71184,277.36,9250,273.36,0.004844292
12843,1496,2024-02-01,11:00:00,1496,241.7266,25.98888,272.26,8530,270.16,0.003900615
. . .
```

1.11 Results

The results of the Tomographic Reconstruction are stored in .npy (NumPy) format as a 3D matrix in the results directory (results/refractivity/refractivity_YYYY-MM-DD-hh.npy). The matrix values represent the wet refractivity values. The matrix indexes are in the following order: latitude, longitude, height. The indices represent the number of the voxel in the specified direction corresponding to the given tomographic grid files.

1.12 Licenses

OpATOM project is under MIT license.

1.13 References

- Bender M, Dick G, Ge M, Deng Z, Wickert J, Kahle HG, Raabe A, Tetzlaff G (2011) Development of a GNSS water vapour tomography system using algebraic reconstruction techniques. Adv Space Res 47:1704–1720. https://doi.org/10.1016/j.asr.2010.05.034
- Boehm J, Kouba J, Schuh H (2009) Forecast Vienna mapping functions 1 for real-time analysis of space geodetic observations. J Geod 83:397–401. https://doi.org/10.1007/s00190-008-0216-y
- Dach R, Lutz S, Walser P, Fridez P (2015) Bernese GNSS Software Version 5.2 User manual. https://doi.org/10.7892/boris.72297
- Hilla S (2016) The extended standard product 3 orbit format (SP3-d). https://files.igs.org/pub/data/format/sp3d.pdf
- Horváth T, Viengdavanh R, Rózsa S (2014) Négydimenziós vízgőzmodellek előállítása GNSS tomográfiával (Construction of 4D water vapour models by means of GNSS tomography). Geomat Közlem 17:69–78. https://geomatika.epss.hun-ren.hu/storage/volumes/gk_XVII_1.pdf
- Lutz S, Beutler G, Schaer S, Dach R, Jäggi A (2014) CODE's new ultra-rapid orbit and ERP products for the IGS. GPS Solut 20:239–250. https://doi.org/10.1007/s10291-014-0432-2
- Niell AE (1996) Global mapping functions for the atmosphere delay at radio wavelengths. J Geophys Res Solid Earth 101:3227–3246. https://doi.org/10.1029/95JB03048
- re3data.org: VMF Data Server; editing status 2020-12-14 (2020) re3data.org–Registry of Research Data Repositories. https://doi.org/10.17616/R3RD2H
- Rózsa S, Khaldi A, Ács Á, Turák B (2021) Multi-GNSS near real-time precipitable water vapour estimation for severe weather prediction. Bull Științ Univ Nord Baia Mare Ser D 35:777–786. https://www.researchgate.net/publication/369649633_MULTI-GNSS_NEAR_REAL-TIME_PRECIPITABLE_WATER_VAPOUR_ESTIMATION_FOR_SEVERE_WEATHER_PREDICTION
- Turák B, Khaldi A, Rózsa S (2024) Tomographic reconstruction of atmospheric water vapor profiles using multi-GNSS observations. Period Polytech Civ Eng 68:155–168. https://doi.org/10.3311/PPci.20559
- Weber J (2024) GSL Radiosonde Database. https://ruc.noaa.gov/raobs/General_Information.html
-

1.14 Author(s)

- Bence Turák - Budapest University of Technology and Economics
- Abir Khaldi - Budapest University of Technology and Economics
- Szabolcs Rózsa - Budapest University of Technology and Economics

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

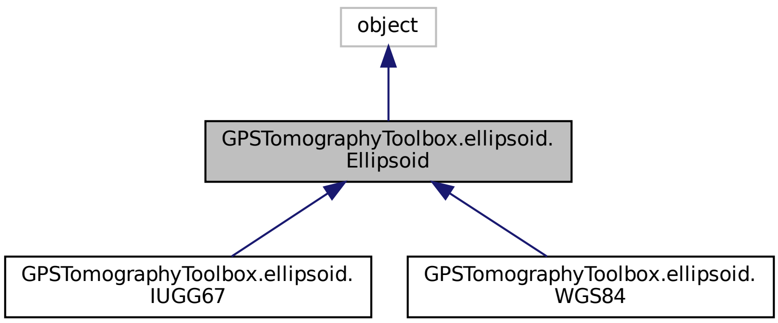
Chapter 5

Class Documentation

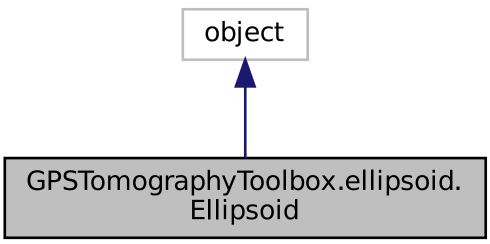
5.1 GPSTomographyToolbox.ellipsoid.Ellipsoid Class Reference

Ellipsoid class to define generic ellipsoidal coordinate system.

Inheritance diagram for GPSTomographyToolbox.ellipsoid.Ellipsoid:



Collaboration diagram for GPSTomographyToolbox.ellipsoid.Ellipsoid:



Public Member Functions

- def __init__ (self)

    Ellipsoid initializer.
- def e (self)

    First eccentricity getter.
- def ec (self)

    Second eccentricity getter.
- def f (self)
- def getXYZ (self, plh)

    Get cartesian (X,Y,Z) coordinates from geographical (longitude, latitude, altitude) coordinates.
- def getPLH (self, xyz)

    Get geographical (longitude, latitude, altitude) coordinates from cartesian (X,Y,Z) coordinates.

5.1.1 Detailed Description

Ellipsoid class to define generic ellipsoidal coordinate system.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 __init__()

```
def GPSTomographyToolbox.ellipsoid.Ellipsoid.__init__ (
              self )
```

Ellipsoid initializer.

5.1.3 Member Function Documentation

5.1.3.1 e()

```
def GPSTomographyToolbox.ellipsoid.Ellipsoid.e (
              self )
```

First eccentricity getter.

Returns

eccentrictiy (float)

5.1.3.2 ec()

```
def GPSTomographyToolbox.ellipsoid.Ellipsoid.ec (
              self )
```

Second eccentricity getter.

Returns

seconf eccentricity (float)

5.1.3.3 f()

```
def GPSTomographyToolbox.ellipsoid.Ellipsoid.f (
              self )
```

```
Flattening getter
@return flattening (float)
```

5.1.3.4 getPLH()

```
def GPSTomographyToolbox.ellipsoid.Ellipsoid.getPLH (
              self,
              xyz )
```

Get geographical (longitude, latitude, altitude) coordinates from cartesian (X,Y,Z) coordinates.

Parameters

xyz	(np.array (1,3)): cartesian coordinates [[x, y, z]]

Returns

plt_coords (np.array (1,3)): geographical coordinates [[phi, lambda, h]]

5.1.3.5 getXYZ()

```
def GPSTomographyToolbox.ellipsoid.Ellipsoid.getXYZ (
              self,
              plh )
```

Get cartesian (X,Y,Z) coordinates from geographical (longitude, latitude, altitude) coordinates.

Parameters

plh	(np.array (1,3)): geographical coordinates [[phi, lambda, h]]

Returns

XYZ_coords (np.array (1,3)): cartesian coordinates (numpy array (1,3)) [[x, y, z]]

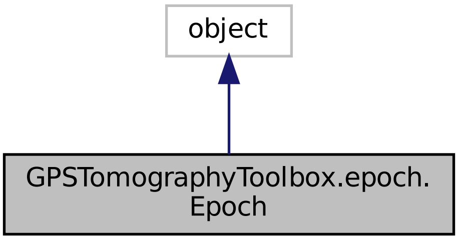
The documentation for this class was generated from the following file:

- ellipsoid.py

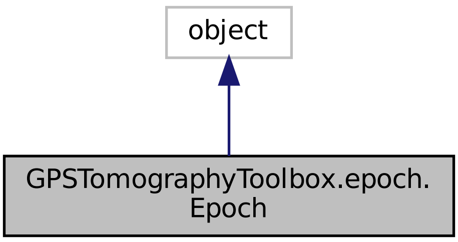
5.2 GPSTomographyToolbox.epoch.Epoch Class Reference

Epoch class to contain datetime and perform operations.

Inheritance diagram for GPSTomographyToolbox.epoch.Epoch:



Collaboration diagram for GPSTomographyToolbox.epoch.Epoch:



Public Member Functions

- def __init__ (self, dt=np.array([1, 0, 0, 0, 0, 0]), system=GPS, downloadLeapSec=False)
  Epoch initializaer.
- def getDateTime (self, system=GPS)
  get DateTime time getter
- def GPSweekTOW (self, week, tow)
  set time by GPS week and tow
- def GPSweek (self)
  get GPS week getter
- def TOW (self)
  get seconds on the GPS week getter
- def DOW (self)
  get day of GPS week getter
- def DOY (self)
  get day of year getter
- def year (self)
  get year getter
- def UTC (self)
  get datetime in UTC time system, getter available leapseconds are required in the given epoch
- def GPS (self)
  get datetime in GPS time system, getter, available leapseconds are required in the given epoch
- def MJD (self)
  get Modified Julian Date, getter
- def date (self)
  get date in formatted string
- def time (self)
  get time in formatted string
- def **floor** (self, n)
- def **ceil** (self, n)
- def **__eq__** (self, other)
- def **__neq__** (self, other)
- def **__gt__** (self, other)
- def **__lt__** (self, other)
- def **__ge__** (self, other)
- def **__le__** (self, other)
- def **__add__** (self, other)
- def **__sub__** (self, other)
- def **__repr__** (self)
- def **__str__** (self)

Public Attributes

- **dt**
- **months**

Static Public Attributes

- list **months** = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

5.2.1 Detailed Description

Epoch class to contain datetime and perform operations.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 __init__()

```
def GPSTomographyToolbox.epoch.Epoch.__init__ (
              self,
              dt = np.array([1,0,0,0,0,0]),
              system = GPS,
              downloadLeapSec = False )
```

Epoch initialiazer.

Parameters

dt	(np.array): datetime in vector [year, month, day, hour, minute, second]
system	time system GPS, UTC (int), default GPS

5.2.3 Member Function Documentation

5.2.3.1 date()

```
def GPSTomographyToolbox.epoch.Epoch.date (
              self )
```

get date in formatted string

Returns

date (str)

5.2.3.2 DOW()

```
def GPSTomographyToolbox.epoch.Epoch.DOW (
              self )
```

get day of GPS week getter

Returns

DOW (int)

5.2.3.3 DOY()

```
def GPSTomographyToolbox.epoch.Epoch.DOY (
              self )
```

get day of year getter

Returns

DOY (int)

5.2.3.4 getDateTime()

```
def GPSTomographyToolbox.epoch.Epoch.getDateTime (
              self,
              system = GPS )
```

get DateTime time getter

Returns

datetime (np.array)

5.2.3.5 GPS()

```
def GPSTomographyToolbox.epoch.Epoch.GPS (
              self )
```

get datetime in GPS time system, getter, available leapseconds are required in the given epoch

Returns

utc (np.array)

5.2.3.6 GPSweek()

```
def GPSTomographyToolbox.epoch.Epoch.GPSweek (
              self )
```

get GPS week getter

Returns

gps_week (int)

5.2.3.7 GPSweekTOW()

```
def GPSTomographyToolbox.epoch.Epoch.GPSweekTOW (
              self,
              week,
              tow )
```

set time by GPS week and tow

Parameters

week	(int): GPSweek
tow	(float): time of week

5.2.3.8 MJD()

```
def GPSTomographyToolbox.epoch.Epoch.MJD (
              self )
```

get Modified Julian Date, getter

Returns

MJD (float)

5.2.3.9 time()

```
def GPSTomographyToolbox.epoch.Epoch.time (
              self )
```

get time in formatted string

Returns

time (str)

5.2.3.10 TOW()

```
def GPSTomographyToolbox.epoch.Epoch.TOW (
              self )
```

get seconds on the GPS week getter

Returns

TOW (float)

5.2.3.11 UTC()

```
def GPSTomographyToolbox.epoch.Epoch.UTC (
              self )
```

get datetime in UTC time system, getter available leapseconds are required in the given epoch

Returns

utc (np.array)

5.2.3.12 year()

```
def GPSTomographyToolbox.epoch.Epoch.year (
              self )
```

get year getter

Returns

year (int)

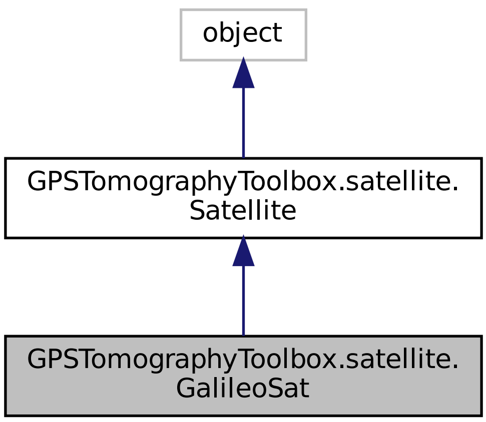
The documentation for this class was generated from the following file:

- epoch.py

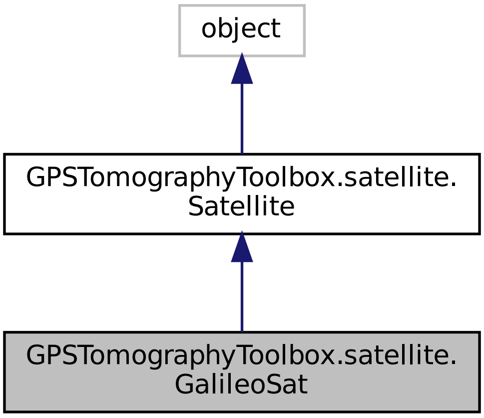
5.3 GPSTomographyToolbox.satellite.GalileoSat Class Reference

GLONASS Satellite class for contain and calculate position.

Inheritance diagram for GPSTomographyToolbox.satellite.GalileoSat:



Collaboration diagram for GPSTomographyToolbox.satellite.GalileoSat:



Public Member Functions

- def **__new__** (self, prn='', nav={})
- def getValidEph (self, epoch)

    get valid navigation message for an epoch
- def getSatPosNav (self, epoch)

    get satellite position in case of GPS satellite
- def T1 (self)

    get L1 period time
- def T5 (self)

    get L5 period time
- def T5a (self)

    get L5a period time
- def T5b (self)

    get L5b period time
- def T6 (self)

    get L6 period time

Static Public Attributes

- float **f1** = 1575.42∗10∗∗6
- float **f5** = 1191.795∗10∗∗6
- float **f5a** = 1176.45∗10∗∗6
- float **f5b** = 1207.14∗10∗∗6
- float **f6** = 1278.750∗10∗∗6

Additional Inherited Members

5.3.1 Detailed Description

GLONASS Satellite class for contain and calculate position.

5.3.2 Member Function Documentation

5.3.2.1 getSatPosNav()

```
def GPSTomographyToolbox.satellite.GalileoSat.getSatPosNav (
              self,
              epoch )
```

get satellite position in case of GPS satellite

Parameters

epoch	(Epoch): timestamp when we get the position of satellite

Returns

(Point): position of satellite at given epoch

Reimplemented from GPSTomographyToolbox.satellite.Satellite.

5.3.2.2 getValidEph()

```
def GPSTomographyToolbox.satellite.GalileoSat.getValidEph (
              self,
              epoch )
```

get valid navigation message for an epoch

Parameters

epoch	(Epoch): reference epoch

Returns

(list): valid nevigation message

5.3.2.3 T1()

```
def GPSTomographyToolbox.satellite.GalileoSat.T1 (
              self )
```

get L1 period time

Returns

(float): L1 period time in seconds

5.3.2.4 T5()

```
def GPSTomographyToolbox.satellite.GalileoSat.T5 (
              self )
```

get L5 period time

Returns

(float): L5 period time in seconds

5.3.2.5 T5a()

```
def GPSTomographyToolbox.satellite.GalileoSat.T5a (
              self )
```

get L5a period time

Returns

(float): L5a period time in seconds

5.3.2.6 T5b()

```
def GPSTomographyToolbox.satellite.GalileoSat.T5b (
              self )
```

get L5b period time

Returns

(float): L5b period time in seconds

5.3.2.7 T6()

```
def GPSTomographyToolbox.satellite.GalileoSat.T6 (
              self )
```

get L6 period time

Returns

(float): L6 period time in seconds

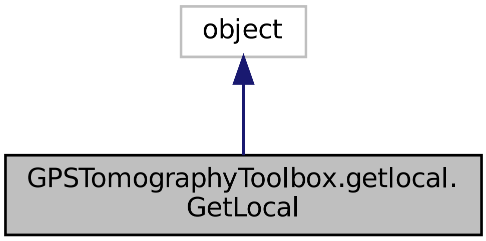
The documentation for this class was generated from the following file:

- satellite.py

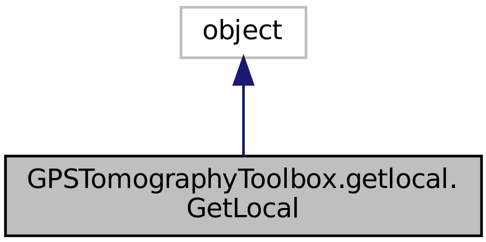
5.4 GPSTomographyToolbox.getlocal.GetLocal Class Reference

Transformation class from ellipsoidal coordinate system to a pre-defined cylindrical projection.

Inheritance diagram for GPSTomographyToolbox.getlocal.GetLocal:



Collaboration diagram for GPSTomographyToolbox.getlocal.GetLocal:



Public Member Functions

- def __init__ (self, min, max)

  GetLocal initializer to define the corners of the area to fit the best cylinder including heights.
- def x (self, lat)

  x coordinate in local coordinate system
- def y (self, lon)

  y coordinate in local coordinate system
- def z (self, h)

  z coordinate in local coordinate system
- def getLocalCoords (self, p)

  Transform ellipsoidal coordinates to local cylindrical coordinates.

Public Attributes

- **min**
- **max**
- **a**
- **b**
- **e**
- **ec**

5.4.1 Detailed Description

Transformation class from ellipsoidal coordinate system to a pre-defined cylindrical projection.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 __init__()

```
def GPSTomographyToolbox.getlocal.GetLocal.__init__ (
              self,
              min,
              max )
```

GetLocal initializer to define the corners of the area to fit the best cylinder including heights.

Parameters

min	(np.array (3,): corner of the fitted CRS
max	(np.array (3,): opposite corner of the fitted CRS

5.4.3 Member Function Documentation

5.4.3.1 getLocalCoords()

```
def GPSTomographyToolbox.getlocal.GetLocal.getLocalCoords (
              self,
              p )
```

Transform ellipsoidal coordinates to local cylindrical coordinates.

Parameters

p	(Point, Station): Point/Station object with available geographical coordinates

Returns

trnsformed_p (Point, Station): Point/Station object in cylindrical CRS.

5.4.3.2 x()

```
def GPSTomographyToolbox.getlocal.GetLocal.x (
              self,
              lat )
```

x coordinate in local coordinate system

Parameters

lat	(float): latitude (radians)

Returns

x (float): meters

5.4.3.3 y()

```
def GPSTomographyToolbox.getlocal.GetLocal.y (
              self,
              lon )
```

y coordinate in local coordinate system

Parameters

lon	(float): longitude (radians)

Returns

y (float): meters

5.4.3.4 z()

```
def GPSTomographyToolbox.getlocal.GetLocal.z (
              self,
              h )
```

z coordinate in local coordinate system

Parameters

h	(float): height (meters)

Returns

z (float): meters

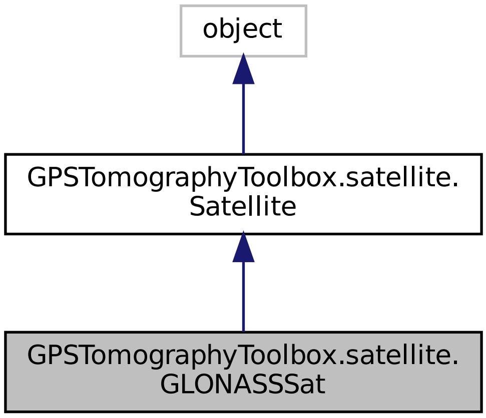
The documentation for this class was generated from the following file:

- getlocal.py

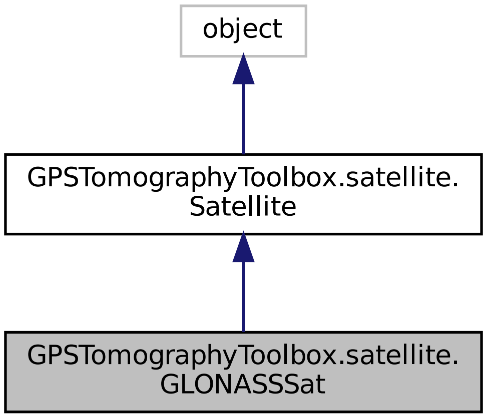
5.5 GPSTomographyToolbox.satellite.GLONASSSat Class Reference

GLONASS Satellite class for contain and calculate position.

Inheritance diagram for GPSTomographyToolbox.satellite.GLONASSSat:



Collaboration diagram for GPSTomographyToolbox.satellite.GLONASSSat:



Public Member Functions

- def **__new__** (self, prn='', nav={})
- def __init__ (self, prn='', nav={})

  GLONASSSat initilaizer.
- def f1 (self)

  get L1 frequency of the satellite
- def f2 (self)

get L2 frequency of the satellite
- def getValidEph (self, epoch)

  get valid navigation message for epoch
- def getSatPosNav (self, epoch)

  get satellite position in case of GLONASS satellite

Public Attributes

- **diffEqSolved**

5.5.1 Detailed Description

GLONASS Satellite class for contain and calculate position.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 __init__()

```
def GPSTomographyToolbox.satellite.GLONASSSat.__init__ (
              self,
              prn = '',
              nav = {} )
```

GLONASSSat initilaizer.

Parameters

prn	(str): PRN number
nav	(dict): navigation messages

5.5.3 Member Function Documentation

5.5.3.1 f1()

```
def GPSTomographyToolbox.satellite.GLONASSSat.f1 (
              self )
```

get L1 frequency of the satellite

Returns

(float): L1 frequency in Hz

5.5.3.2 f2()

```
def GPSTomographyToolbox.satellite.GLONASSSat.f2 (
              self )
```

get L2 frequency of the satellite

Returns

(float): L2 frequency in Hz

5.5.3.3 getSatPosNav()

```
def GPSTomographyToolbox.satellite.GLONASSSat.getSatPosNav (
              self,
              epoch )
```

get satellite position in case of GLONASS satellite

Parameters

epoch	(Epoch): timestamp when of the position of satellite

Returns

(Point): position of satellite at given epoch

Reimplemented from GPSTomographyToolbox.satellite.Satellite.

5.5.3.4 getValidEph()

```
def GPSTomographyToolbox.satellite.GLONASSSat.getValidEph (
              self,
              epoch )
```

get valid navigation message for epoch

Parameters

epoch	(Epoch): timestamp what of valid nav message for (Epoch)

Returns

(list): valid navigation message

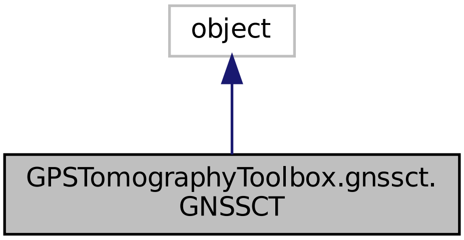
The documentation for this class was generated from the following file:

- satellite.py

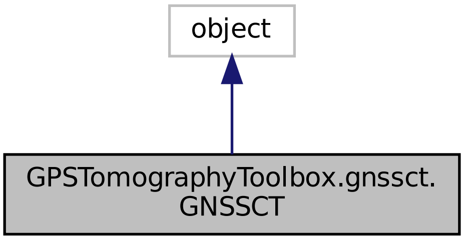
5.6 GPSTomographyToolbox.gnssct.GNSSCT Class Reference

GNSSCT class to handle all input, output and parameter files together to star the tomographic procession.

Inheritance diagram for GPSTomographyToolbox.gnssct.GNSSCT:



Collaboration diagram for GPSTomographyToolbox.gnssct.GNSSCT:



Public Member Functions

- def __init__ (self, gridp, gridl, gridh, x0_3D_w, network, troposphere, mapping_function, ep, constellation=('G', 'R', 'E'), max_iter=3000, tolerance=2.7, output_root="./")

  GNSSCT class initializer.
- def writeNw2npy (self, fname)

  Write reconstructed wet refractivity model to file in .npy (numpy) format.
- def run (self)

  Run tomograhpic procession and adjust observation get reconstructed wet refractivity model to the selected area.

Public Attributes

- **gridp**
- **gridl**
- **gridh**
- **cellX**
- **cellY**
- **cellZ**
- **x0_3D_w**
- **network**
- **troposphere**
- **ep**
- **mapping_function**
- **constellation**
- **max_iter**
- **tolerance**
- **Nw_3D**
- **Nh_3D**
- **output_root**

5.6.1 Detailed Description

GNSSCT class to handle all input, output and parameter files together to star the tomographic procession.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 __init__()

```
def GPSTomographyToolbox.gnssct.GNSSCT.__init__ (
              self,
              gridp,
              gridl,
              gridh,
              x0_3D_w,
              network,
              troposphere,
              mapping_function,
              ep,
              constellation = ('G', 'R', 'E'),
              max_iter = 3000,
              tolerance = 2.7,
              output_root = "./" )
```

GNSSCT class initializer.

Parameters

gridp	(np.array): tomographic grid (longitude) in radians
gridl	(np.array): tomographic grid (latitude) in radians

Parameters

gridh	(np.array): tomographic grid (height) in meters
x0_3D_w	(np.array): intital 3D wet refractivity model
network	(Network): network object that contains all the reference stations and satellite orbits in the reference epoch
trpopsphere	(ReadTRP): parsed troposheric delays from Benese TRP file in the reference epoch
mapping_function	(VMF1): Vienna Mapping Funtion 1 with the recent VMF1 parameters
ep	(Epoch): refernce epoch
constellations	(tuple): list of applied GNSS constellations (G => GPS, R => GLONASS, E => Galileo), diffault: ('G', 'R', 'E')
max_iter	(int): number of maximum iteration for MART algorithm, default: 3000
tolerance	(float): value of tolerance for MART algorithm: 2.7
output_root	(str): location of output files, default: "./"

5.6.3 Member Function Documentation

5.6.3.1 run()

```
def GPSTomographyToolbox.gnssct.GNSSCT.run (
              self )
```

Run tomograhpic procession and adjust observation get reconstructed wet refractivity model to the selected area.

The method uses the given poarameters of the object and the results will be stored in Nw_3D parameter of the object.

5.6.3.2 writeNw2npy()

```
def GPSTomographyToolbox.gnssct.GNSSCT.writeNw2npy (
              self,
              fname )
```

Write reconstructed wet refractivity model to file in .npy (numpy) format.

Parameters

fname	(str): file name

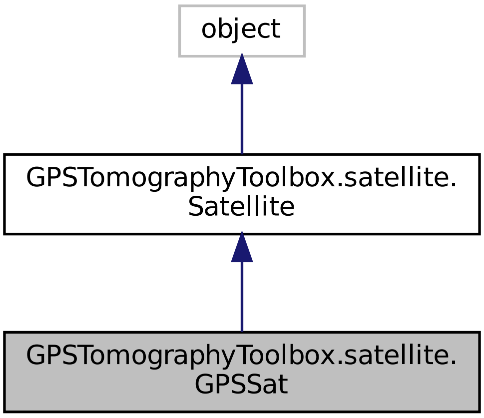
The documentation for this class was generated from the following file:

- gnssct.py

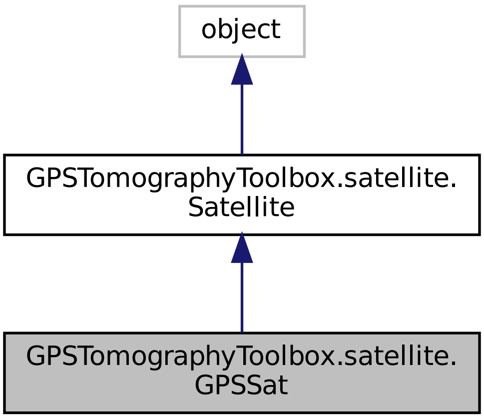
5.7 GPSTomographyToolbox.satellite.GPSSat Class Reference

GPS Satellite class for contain and calculate position.

Inheritance diagram for GPSTomographyToolbox.satellite.GPSSat:



Collaboration diagram for GPSTomographyToolbox.satellite.GPSSat:



Public Member Functions

- def **__new__** (self, prn='', nav={})
- def getValidEph (self, epoch)

get valid navigation message for an epoch
- def getSatPosNav (self, epoch)

  get satellite position in case of GPS satellite
- def T1 (self)

  get L1 period time
- def T2 (self)

  get L2 period time
- def T5 (self)

  get L5 period time

Static Public Attributes

- float **f1** = 1575.42*10**6
- float **f2** = 1227.60*10**6
- float **f5** = 1176.45*10**6

Additional Inherited Members

5.7.1 Detailed Description

GPS Satellite class for contain and calculate position.

5.7.2 Member Function Documentation

5.7.2.1 getSatPosNav()

```
def GPSTomographyToolbox.satellite.GPSSat.getSatPosNav (
            self,
            epoch )
```

get satellite position in case of GPS satellite

Parameters

epoch	(Epoch): timestamp when we get the position of satellite

Returns

(Point): position of satellite at given epoch

Reimplemented from GPSTomographyToolbox.satellite.Satellite.

5.7.2.2 getValidEph()

```
def GPSTomographyToolbox.satellite.GPSSat.getValidEph (
              self,
              epoch )
```

get valid navigation message for an epoch

Parameters

epoch	(Epoch): reference epoch

Returns

(list): valid nevigation message

5.7.2.3 T1()

```
def GPSTomographyToolbox.satellite.GPSSat.T1 (
              self )
```

get L1 period time

Returns

(float): L1 period time in seconds

5.7.2.4 T2()

```
def GPSTomographyToolbox.satellite.GPSSat.T2 (
              self )
```

get L2 period time

Returns

(float): L2 period time in seconds

5.7.2.5 T5()

```
def GPSTomographyToolbox.satellite.GPSSat.T5 (
              self )
```

get L5 period time

Returns

(float): L5 period time in seconds

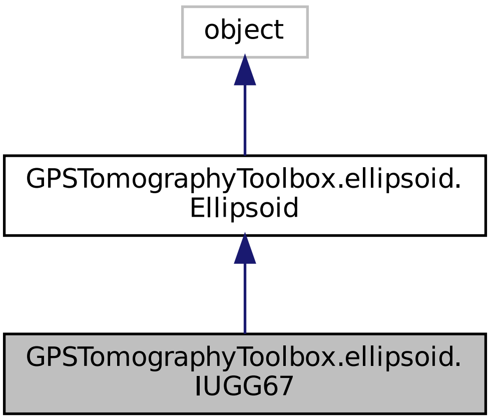
The documentation for this class was generated from the following file:

- satellite.py

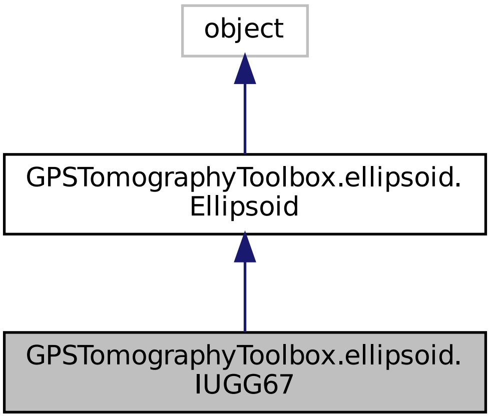
5.8 GPSTomographyToolbox.ellipsoid.IUGG67 Class Reference

IUGG67 class to define IUGG67 ellipsoidal coordinate system.

Inheritance diagram for GPSTomographyToolbox.ellipsoid.IUGG67:



Collaboration diagram for GPSTomographyToolbox.ellipsoid.IUGG67:



Static Public Attributes

- float **a** = 6378160.000
- float **b** = 6356774.516

Additional Inherited Members

5.8.1 Detailed Description

IUGG67 class to define IUGG67 ellipsoidal coordinate system.

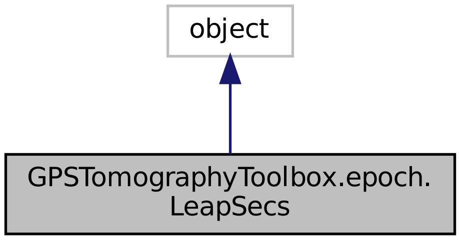
The documentation for this class was generated from the following file:

- ellipsoid.py

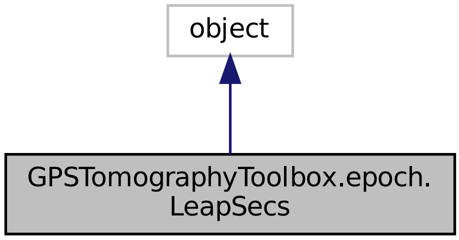
5.9 GPSTomographyToolbox.epoch.LeapSecs Class Reference

LeapSecs class to handle leap seconds.

Inheritance diagram for GPSTomographyToolbox.epoch.LeapSecs:



Collaboration diagram for GPSTomographyToolbox.epoch.LeapSecs:



Public Member Functions

- def __init__ (self, fileName='Leap_Second.dat', url='https://hpiers.obspm.fr/iers/bul/bulc/Leap_Second.dat', download=False)

    LeapSecs initializer.
- def **getLeapSecsAt** (self, epoch, fr=GPS)

Public Attributes

- **fileName**
- **leapSecs**
- **fid**

5.9.1 Detailed Description

LeapSecs class to handle leap seconds.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 __init__()

```
def GPSTomographyToolbox.epoch.LeapSecs.__init__ (
              self,
              fileName = 'Leap_Second.dat',
              url = 'https://hpiers.obspm.fr/iers/bul/bulc/Leap_Second.dat',
              download = False )
```

LeapSecs initializer.

Parameters

fileName	(str): default Leap_second.dat
url	(str): url of leapsec file, default `https://hpiers.obspm.fr/iers/bul/bulc/Leap_Second.dat` (IERS bulletin C)
download	(boolean): download the leapsec file?, default False

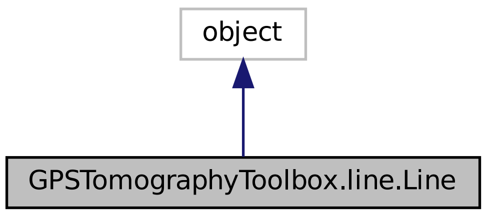
The documentation for this class was generated from the following file:

- epoch.py

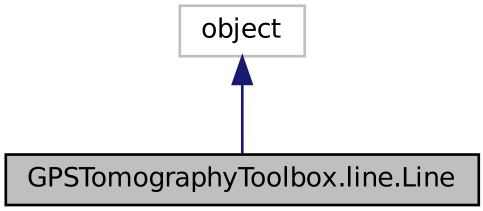
5.10 GPSTomographyToolbox.line.Line Class Reference

Line object to to define line in 3d cartesian coordinate system.

Inheritance diagram for GPSTomographyToolbox.line.Line:



Collaboration diagram for GPSTomographyToolbox.line.Line:



Public Member Functions

- def __init__ (self, p, alpha, e)

  Line initializer.
- def getPointAtT (self, t)

  Get contained Point of the Line where 't' parameter is.
- def getTwhereX (self, x)

  Get 't' paramater of the eqations where X coordinate is.
- def getTwhereY (self, y)

  Get 't' paramater of the eqations where Y coordinate is.
- def getTwhereZ (self, z)

  Get 't' paramater of the eqations where Z coordinate is.

Public Attributes

- **x**
- **y**
- **z**
- **xr**
- **yr**
- **zr**

5.10.1 Detailed Description

Line object to to define line in 3d cartesian coordinate system.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 __init__()

```
def GPSTomographyToolbox.line.Line.__init__ (
              self,
              p,
              alpha,
              e )
```

Line initializer.

Parameters

p	(Point): point contained by the line
alpha	(float): angle from the axis X (azimuth) in radians
e	(float): elevation angle in radians

5.10.3 Member Function Documentation

5.10.3.1 getPointAtT()

```
def GPSTomographyToolbox.line.Line.getPointAtT (
              self,
              t )
```

Get contained Point of the Line where 't' parameter is.

Parameters

t	(tuple)(float): list t parameters

Returns

point (Point): coordinates of line

5.10.3.2 getTwhereX()

```
def GPSTomographyToolbox.line.Line.getTwhereX (
              self,
              x )
```

Get 't' paramater of the eqations where X coordinate is.

Parameters

x	(float): X coordinate

Returns

t (float): t parameter

5.10.3.3 getTwhereY()

```
def GPSTomographyToolbox.line.Line.getTwhereY (
              self,
              y )
```

Get 't' paramater of the eqations where Y coordinate is.

Parameters

y	(float): Y coordinate

Returns

t (float): t parameter

5.10.3.4 getTwhereZ()

```
def GPSTomographyToolbox.line.Line.getTwhereZ (
              self,
              z )
```

Get 't' paramater of the eqations where Z coordinate is.

Parameters

z	(float): Z coordinate

Returns

t (float): t parameter

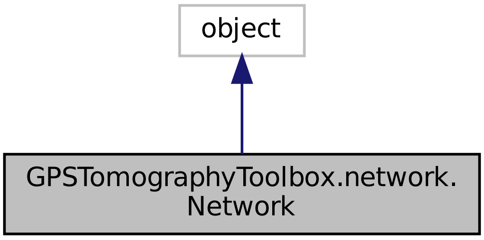
The documentation for this class was generated from the following file:

- line.py

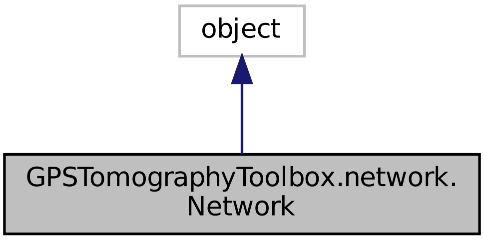
5.11 GPSTomographyToolbox.network.Network Class Reference

Network class to collect stations and satellites.

Inheritance diagram for GPSTomographyToolbox.network.Network:



Collaboration diagram for GPSTomographyToolbox.network.Network:



Public Member Functions

- def __init__ (self)

    Network initializer.
- def getStations (self)

    get stations method generator function
- def getSatellites (self)

    get satellites method generator function
- def getStationBy4digitId (self, id)

    get an exact station, select by the 4 digit ID
- def addStation (self, st)

    add station to the network
- def addSatellite (self, sat)

    add satellite to the network
- def getStationsMatrix (self)

    get stations' ids and coordinates in matrix

Public Attributes

- **stations**
- **satellites**

5.11.1 Detailed Description

Network class to collect stations and satellites.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 __init__()

```
def GPSTomographyToolbox.network.Network.__init__ (
              self )
```

Network initializer.

5.11.3 Member Function Documentation

5.11.3.1 addSatellite()

```
def GPSTomographyToolbox.network.Network.addSatellite (
              self,
              sat )
```

add satellite to the network

Parameters

sat	(Satellite): satellite

5.11.3.2 addStation()

```
def GPSTomographyToolbox.network.Network.addStation (
              self,
              st )
```

add station to the network

Parameters

st	(Point,Station): station

5.11.3.3 getSatellites()

```
def GPSTomographyToolbox.network.Network.getSatellites (
              self )
```

get satellites method generator function

Returns

network_satellites (Satellite): list of satellites, generator

5.11.3.4 getStationBy4digitId()

```
def GPSTomographyToolbox.network.Network.getStationBy4digitId (
              self,
              id )
```

get an exact station, select by the 4 digit ID

Parameters

id	(str): 4 digit ID

Returns

station (Station, Point): station

5.11.3.5 getStations()

```
def GPSTomographyToolbox.network.Network.getStations (
              self )
```

get stations method generator function

Returns

network_stations (Station/Point): list of stations, generator

5.11.3.6 getStationsMatrix()

```
def GPSTomographyToolbox.network.Network.getStationsMatrix (
             self )
```

get stations' ids and coordinates in matrix

Returns

(tuple {ids: (Str), coords: numpy array (n,3)}): ids and coordinates of stations

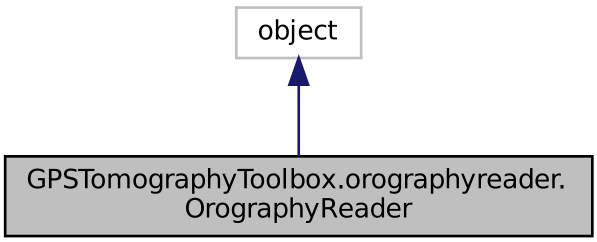
The documentation for this class was generated from the following file:

- network.py

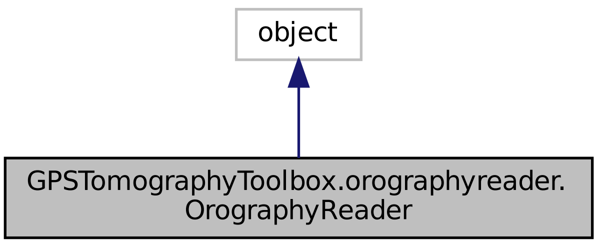
5.12 GPSTomographyToolbox.orographyreader.OrographyReader Class Reference

OrographyReader class to read Orography grid file.

Inheritance diagram for GPSTomographyToolbox.orographyreader.OrographyReader:



Collaboration diagram for GPSTomographyToolbox.orographyreader.OrographyReader:



Public Member Functions

- def __init__ (self, fileName)
- def getOro (self, st)

    get orography at the given station

Public Attributes

- **fileName**
- **grid**
- **epochs**
- **fid**
- **p_min**
- **p_max**
- **l_min**
- **l_max**
- **p_d**
- **l_d**
- **phi**
- **lam**

5.12.1 Detailed Description

OrographyReader class to read Orography grid file.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 __init__()

```
def GPSTomographyToolbox.orographyreader.OrographyReader.__init__ (
              self,
              fileName )
```

```
OrograpgyReader initializer
@param fileName (string): name of Orography file
```

5.12.3 Member Function Documentation

5.12.3.1 getOro()

```
def GPSTomographyToolbox.orographyreader.OrographyReader.getOro (
              self,
              st )
```

get orography at the given station

Parameters

st	(Point, Station): station

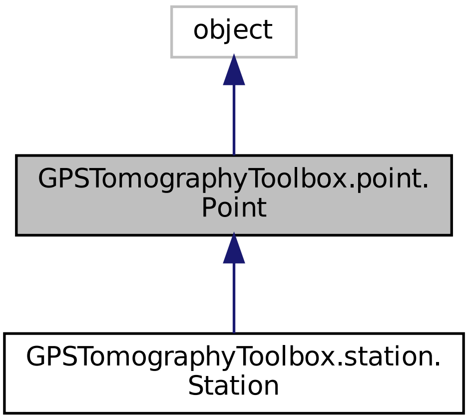
The documentation for this class was generated from the following file:

- orographyreader.py

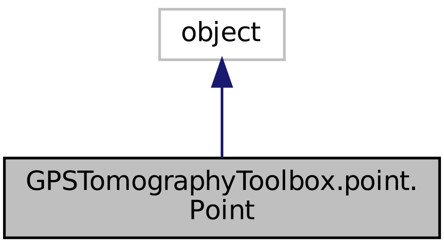
5.13 GPSTomographyToolbox.point.Point Class Reference

Point class to store and make calculations on points in cartesian and geographical coordinate system.

Inheritance diagram for GPSTomographyToolbox.point.Point:



Collaboration diagram for GPSTomographyToolbox.point.Point:



Public Member Functions

- def __init__ (self, id='', code='', coord=np.array([[0.0],[0.0],[0.0]]), type=XYZ, system=None, other=None)

  Point inizializer.
- def getXYZ (self)

  get coordinates in cartesian system.
- def getPLH (self)
- def **xyz** (self)
- def **xyz** (self, c)
- def **plh** (self)
- def **plh** (self, c)
- def **id** (self)
- def **other** (self)
- def dist (self, other)

  get distance from another Point
- def **__add__** (self, other)
- def **__sub__** (self, other)
- def **__repr__** (self)
- def **__str__** (self)

Public Attributes

- **code**
- **system**

5.13.1 Detailed Description

Point class to store and make calculations on points in cartesian and geographical coordinate system.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 __init__()

```
def GPSTomographyToolbox.point.Point.__init__ (
                self,
                id = '',
                code = '',
                coord = np.array([[0.0],[0.0],[0.0]]),
                type = XYZ,
                system = None,
                other = None )
```

Point inizializer.

Parameters

id	(str): point ID, default: ''
code	(str): point coode (Str), default: ''
coord	(numpy array (3,1)): coordinates (cartesian or geographical), default: [[0, 0, 0]]
type	(int): type of coordinate system, variable: XYZ/PLH , default: XYZ
system	(Ellipsoid object): base ellipsoid, default: None

5.13.3 Member Function Documentation

5.13.3.1 dist()

```
def GPSTomographyToolbox.point.Point.dist (
              self,
              other )
```

get distance from another Point

Parameters

other	point (Point object)

Returns

: distance between the two points (float)

5.13.3.2 getPLH()

```
def GPSTomographyToolbox.point.Point.getPLH (
              self )
```

```
get coordinates in ellipsoidal system. For the transformation to set up system is required
@return coord: ellipsoidal coordinates (numpy array (3,1))
```

5.13.3.3 getXYZ()

```
def GPSTomographyToolbox.point.Point.getXYZ (
              self )
```

get coordinates in cartesian system.

For the transformation to set up system is required

Returns

coord: cartesian coordinates (numpy array (3,1))

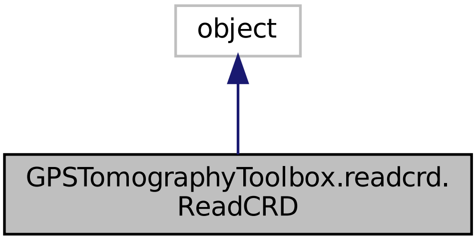
The documentation for this class was generated from the following file:

- point.py

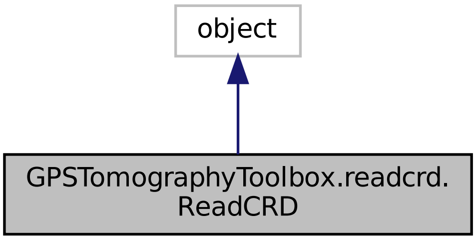
5.14 GPSTomographyToolbox.readcrd.ReadCRD Class Reference

ReadCRD class to read Bernese CRD format file The content of the Bernese CRD file will be sored and can be used inv the newtork parameter of the class in a Network object.

Inheritance diagram for GPSTomographyToolbox.readcrd.ReadCRD:



Collaboration diagram for GPSTomographyToolbox.readcrd.ReadCRD:



Public Member Functions

- def __init__ (self, fileName)

  ReadCRD constructor.

Public Attributes

- **fileName**
- **network**
- **fid**

5.14.1 Detailed Description

ReadCRD class to read Bernese CRD format file The content of the Bernese CRD file will be sored and can be used inv the newtork parameter of the class in a Network object.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 __init__()

```
def GPSTomographyToolbox.readcrd.ReadCRD.__init__ (
              self,
              fileName )
```

ReadCRD constructor.

Parameters

fileName	(str): name of CRD file

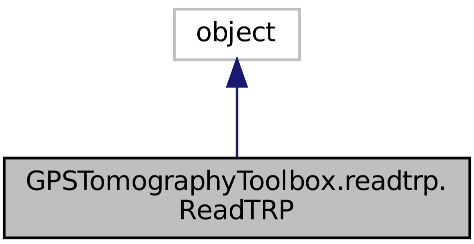
The documentation for this class was generated from the following file:

- readcrd.py

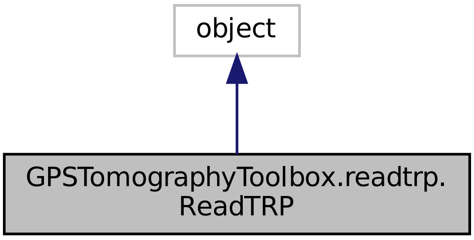
5.15 GPSTomographyToolbox.readtrp.ReadTRP Class Reference

ReadTRP class to read Bernese TRP (troposphere) format file.

Inheritance diagram for GPSTomographyToolbox.readtrp.ReadTRP:



Collaboration diagram for GPSTomographyToolbox.readtrp.ReadTRP:



Public Member Functions

- def __init__ (self, fileName=None, database=None, table=None, type=TXT)
  ReadTRP class initializer.
- def get_MOD_U (self, digit4Id, ep)
  get MOD_U (ZHD) value at the given station and epoch
- def get_CORR_U (self, digit4Id, ep)
  get CORR_U (ZWD) value at the given station and epoch
- def get_SIGMA_U (self, digit4Id, ep)
  get SIGMA_U (standard deviation CORR_U) value at the given station and epoch
- def get_TOTAL_U (self, digit4Id, ep)
  get TOTAL_U (ZTD) value at the given station and epoch
- def get_CORR_N (self, digit4Id, ep)
  get CORR_N (tropospheric gradient north) value at the given station and epoch
- def get_SIGMA_N (self, digit4Id, ep)
  get SIGMA_N (tropospheric gradient STD north) value at the given station and epoch
- def get_CORR_E (self, digit4Id, ep)
  get CORR_E (tropospheric gradient east) value at the given station and epoch
- def get_SIGMA_E (self, digit4Id, ep)
  get SIGMA_E (tropospheric gradient STD east) value at the given station and epoch
- def **getTropoByStationEpoch** (self, digit4Id, ep)

Public Attributes

- **fileName**
- **troposphere**
- **fid**
- **database**
- **table**
- **type**

5.15.1 Detailed Description

ReadTRP class to read Bernese TRP (troposphere) format file.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 __init__()

```
def GPSTomographyToolbox.readtrp.ReadTRP.__init__ (
              self,
              fileName = None,
              database = None,
              table = None,
              type = TXT )
```

ReadTRP class initializer.

Parameters

fileName	(str): location of Brenese toposphere file (TRP), default: None
database	(mysql.connector): connected mysql database, default: None
table	(str): name of table in case of database, default: None
type	(int): in case of text file: 1, in casa of database: 2, dafault: TXT

5.15.3 Member Function Documentation

5.15.3.1 get_CORR_E()

```
def GPSTomographyToolbox.readtrp.ReadTRP.get_CORR_E (
              self,
              digit4Id,
              ep )
```

get CORR_E (tropospheric gradient east) value at the given station and epoch

Parameters

digit4Id	(str): station ID
ep	(epoch): epoch

Returns

(float): CORR_E (tropospheric gradient east) value at the given stgation and epoch

5.15.3.2 get_CORR_N()

```
def GPSTomographyToolbox.readtrp.ReadTRP.get_CORR_N (
              self,
              digit4Id,
              ep )
```

get CORR_N (tropospheric gradient north) value at the given station and epoch

Parameters

digit4↩ Id	(str): station ID
ep	(epoch): epoch

Returns

(float): CORR_N (tropospheric gradient north) value at the given stgation and epoch

5.15.3.3 get_CORR_U()

```
def GPSTomographyToolbox.readtrp.ReadTRP.get_CORR_U (
              self,
              digit4Id,
              ep )
```

get CORR_U (ZWD) value at the given station and epoch

Parameters

digit4↩ Id	(str): station ID
ep	(epoch): epoch

Returns

(float): MOD_U (ZHD) value at the given stgation and epoch

5.15.3.4 get_MOD_U()

```
def GPSTomographyToolbox.readtrp.ReadTRP.get_MOD_U (
              self,
              digit4Id,
              ep )
```

get MOD_U (ZHD) value at the given station and epoch

Parameters

digit4↩ Id	(str): station ID
ep	(epoch): epoch

Returns

(float): MOD_U (ZHD) value at the given stgation and epoch

5.15.3.5 get_SIGMA_E()

```
def GPSTomographyToolbox.readtrp.ReadTRP.get_SIGMA_E (
              self,
              digit4Id,
              ep )
```

get SIGMA_E (tropospheric gradient STD east) value at the given station and epoch

Parameters

digit4↩ Id	(str): station ID
ep	(epoch): epoch

Returns

(float): SIGMA_E (tropospheric gradient STD east) value at the given stgation and epoch

5.15.3.6 get_SIGMA_N()

```
def GPSTomographyToolbox.readtrp.ReadTRP.get_SIGMA_N (
              self,
              digit4Id,
              ep )
```

get SIGMA_N (tropospheric gradient STD north) value at the given station and epoch

Parameters

digit4↩ Id	(str): station ID
ep	(epoch): epoch

Returns

(float): SIGMA_N (tropospheric gradient STD north) value at the given stgation and epoch

5.15.3.7 get_SIGMA_U()

```
def GPSTomographyToolbox.readtrp.ReadTRP.get_SIGMA_U (
              self,
              digit4Id,
              ep )
```

get SIGMA_U (standard deviation CORR_U) value at the given station and epoch

Parameters

digit4←↩ Id	(str): station ID
ep	(epoch): epoch

Returns

(float): SIGMA_U (standard deviation CORR_U) value at the given stgation and epoch

5.15.3.8 get_TOTAL_U()

```
def GPSTomographyToolbox.readtrp.ReadTRP.get_TOTAL_U (
              self,
              digit4Id,
              ep )
```

get TOTAL_U (ZTD) value at the given station and epoch

Parameters

digit4←↩ Id	(str): station ID
ep	(epoch): epoch

Returns

(float): TOTAL_U (ZTD) value at the given stgation and epoch

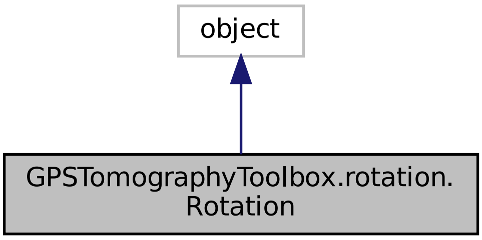
The documentation for this class was generated from the following file:

- readtrp.py

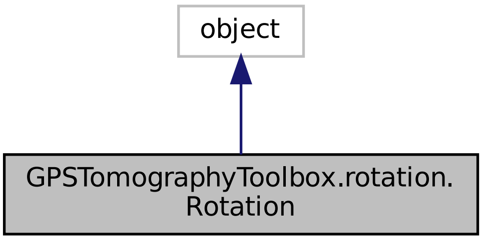
5.16 GPSTomographyToolbox.rotation.Rotation Class Reference

Rotation class to transform points from a CRD to another.

Inheritance diagram for GPSTomographyToolbox.rotation.Rotation:



Collaboration diagram for GPSTomographyToolbox.rotation.Rotation:



Public Member Functions

- def __init__ (self, x=0, y=0, z=0, order='xyz')
- def setRot (self, R)

  set up rotation matrix directly
- def **__mul__** (self, other)
- def **__repr__** (self)
- def **__str__** (self)

Public Attributes

- **matrix**

5.16.1 Detailed Description

Rotation class to transform points from a CRD to another.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 __init__()

```
def GPSTomographyToolbox.rotation.Rotation.__init__ (
              self,
              x = 0,
              y = 0,
              z = 0,
              order = 'xyz' )
```

```
Rotation initializer
@param x (float): rotation angle around axis x in radian, default: 0
@param y (float): rotation angle around axis y in radian, default: 0
@param z (float): rotation angle around axis z in radian, default: 0
@param order (str): order of rotations' axis, default: 'xyz'
```

5.16.3 Member Function Documentation

5.16.3.1 setRot()

```
def GPSTomographyToolbox.rotation.Rotation.setRot (
              self,
              R )
```

set up rotation matrix directly

Parameters

R	(numpy array (3,3)): rotation matrix

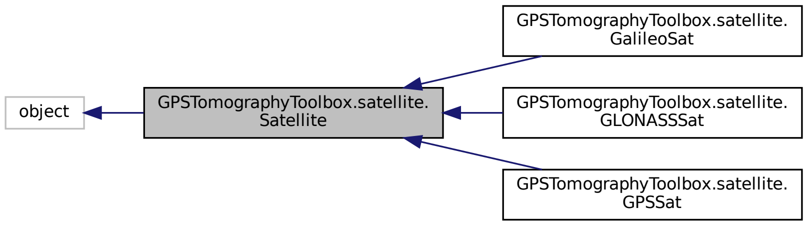
The documentation for this class was generated from the following file:

- rotation.py

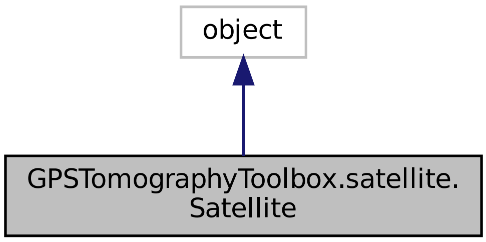
5.17 GPSTomographyToolbox.satellite.Satellite Class Reference

Satellite class for contain and calc position.

Inheritance diagram for GPSTomographyToolbox.satellite.Satellite:



Collaboration diagram for GPSTomographyToolbox.satellite.Satellite:



Public Member Functions

- def **__new__** (self, prn='', nav={})
- def __init__ (self, prn='', nav={}, coords=[ ])
- def l1 (self)

    get wavelength of L1 frequency
- def l2 (self)

    get wavelength of L2 frequency
- def l5 (self)

    get wavelength of L5 frequency
- def **getTimeFrameByElevAzimuthMask** (self, elevation, azimuth, st)
- def addNavMess (self, nav)
- def addSP3coords (self, coords)
- def getElevAzimuth (self, st, epoch)
- def getEpochsInValidTimeFrame (self, timeDiff=Epoch(np.array([0, 0, 0, 0, 15, 0])))
- def getSatPosSP3 (self, epoch)

    get position of the satellite at the given epoch
- def getSatPosNav (self, epoch)

    get position of the satellite at the given epoch
- def getSatPos (self, epoch)

    get position of the satellite at the given epoch

Public Attributes

- **system**
- **prn**
- **coords**
- **navigationDatas**
- **source**

5.17.1 Detailed Description

Satellite class for contain and calc position.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 __init__()

```
def GPSTomographyToolbox.satellite.Satellite.__init__ (
              self,
              prn = '',
              nav = {},
              coords = [] )
```

```
Satellite constructor
@param prn (str): satellite PRN, default: ''
@param nav (dict): navigation message, default: {}
@partam coords (numoy array): list of coordinates, default: []
```

5.17.3 Member Function Documentation

5.17.3.1 addNavMess()

```
def GPSTomographyToolbox.satellite.Satellite.addNavMess (
              self,
              nav )
```

```
add new navigation message of an epoch

@param nav (dictionary): navigation message
```

5.17.3.2 addSP3coords()

```
def GPSTomographyToolbox.satellite.Satellite.addSP3coords (
              self,
              coords )
```

```
add new coordinates to satellite

@param coords (numpy array): list of coordinates from SP3 file
```

5.17.3.3 getElevAzimuth()

```
def GPSTomographyToolbox.satellite.Satellite.getElevAzimuth (
              self,
              st,
              epoch )
```

```
get elevetion and azimuth angle from point at epoch
@param st (Point): refernce station
@param epoch (Epoch): reference epoch
@return (numpy array): elevation and azimuth angle in radian
```

5.17.3.4 getEpochsInValidTimeFrame()

```
def GPSTomographyToolbox.satellite.Satellite.getEpochsInValidTimeFrame (
              self,
              timeDiff = Epoch(np.array([0,0,0,0,15,0])) )
```

```
method to get epochs in the given messages valid time frame

@param timeDiff (Epoch): difference between 2 epoch (Epoch), default: Epoch(np.array([0,0,0,0,15,0]))
@return (list, Epoch): list of epochs in the valid time frame
```

5.17.3.5 getSatPos()

```
def GPSTomographyToolbox.satellite.Satellite.getSatPos (
              self,
              epoch )
```

get position of the satellite at the given epoch

Parameters

epoch	(Epoch): reference epoch

Returns

(Point): position of the satellite at the given apoch

5.17.3.6 getSatPosNav()

```
def GPSTomographyToolbox.satellite.Satellite.getSatPosNav (
            self,
            epoch )
```

get position of the satellite at the given epoch

Parameters

epoch	(Epoch): reference epoch

Returns

(Point): position of the satellite at the given apoch

Reimplemented in GPSTomographyToolbox.satellite.GalileoSat, GPSTomographyToolbox.satellite.GLONASSSat, and GPSTomographyToolbox.satellite.GPSSat.

5.17.3.7 getSatPosSP3()

```
def GPSTomographyToolbox.satellite.Satellite.getSatPosSP3 (
            self,
            epoch )
```

get position of the satellite at the given epoch

Parameters

epoch	(Epoch): reference epoch

Returns

(Point): position of the satellite at the given apoch

5.17.3.8 l1()

```
def GPSTomographyToolbox.satellite.Satellite.l1 (
              self )
```

get wavelength of L1 frequency

Returns

(float): wavelength of L1 frquency

5.17.3.9 l2()

```
def GPSTomographyToolbox.satellite.Satellite.l2 (
              self )
```

get wavelength of L2 frequency

Returns

(float): wavelength of L2 frquency

5.17.3.10 l5()

```
def GPSTomographyToolbox.satellite.Satellite.l5 (
              self )
```

get wavelength of L5 frequency

Returns

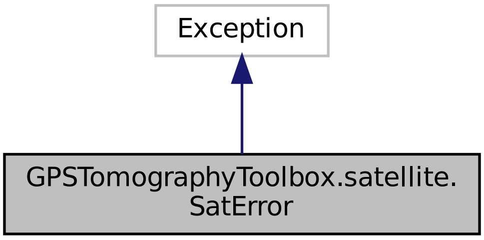
(float): wavelength of L5 frquency

The documentation for this class was generated from the following file:

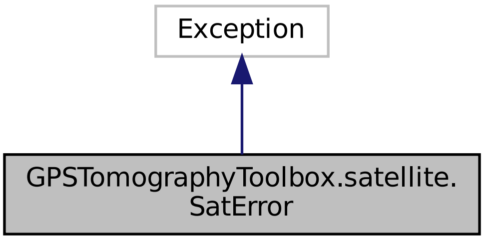
- satellite.py

5.18 GPSTomographyToolbox.satellite.SatError Class Reference

Inheritance diagram for GPSTomographyToolbox.satellite.SatError:



Collaboration diagram for GPSTomographyToolbox.satellite.SatError:



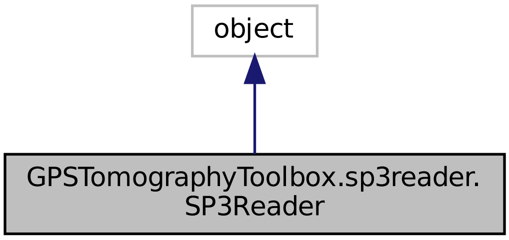
The documentation for this class was generated from the following file:

- satellite.py

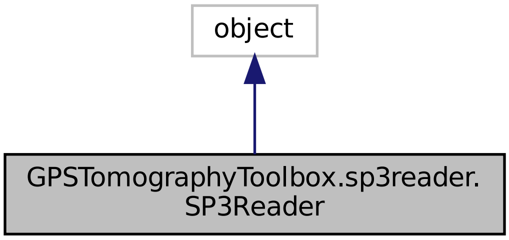
5.19 GPSTomographyToolbox.sp3reader.SP3Reader Class Reference

SP3Reader class to read and parse SP3 format satellite orbit file.

Inheritance diagram for GPSTomographyToolbox.sp3reader.SP3Reader:



Collaboration diagram for GPSTomographyToolbox.sp3reader.SP3Reader:



Public Member Functions

- def __init__ (self, fileName)
- def getSatellite (self, prn)

    get satellite by PRN number
- def getSatellites (self)

    get list of satellites with the orbit (generator method)
- def **headerRow1** (self, line)
- def **headerRow2** (self, line)
- def **headerRow3** (self, line)
- def **headerRow4** (self, line)
- def **headerRow5** (self, line)
- def **headerRow6** (self, line)
- def **headerRow7** (self, line)
- def **headerRow8** (self, line)
- def **headerRow9** (self, line)
- def **headerRow10** (self, line)
- def **headerRow11** (self, line)
- def **headerRow12** (self, line)

- def **headerRow13** (self, line)
- def **headerRow14** (self, line)
- def **headerRow15** (self, line)
- def **headerRow16** (self, line)
- def **headerRow17** (self, line)
- def **headerRow18** (self, line)
- def **headerRow19** (self, line)
- def **headerRow20** (self, line)
- def **headerRow21** (self, line)
- def **headerRow22** (self, line)

Public Attributes

- **fileName**
- **comments**
- **numOfSats**
- **positions**
- **accuracy**
- **fid**
- **version**
- **posVerFlag**
- **startDate**
- **numOfEpochs**
- **dataUsed**
- **coordinateSystem**
- **orbitType**
- **agency**
- **GPSweek**
- **secondsOfWeek**
- **epochInterval**
- **MDF**
- **fractionalDay**

5.19.1 Detailed Description

SP3Reader class to read and parse SP3 format satellite orbit file.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 __init__()

```
def GPSTomographyToolbox.sp3reader.SP3Reader.__init__ (
              self,
              fileName )
```

```
SP3Reader initializer
@param fileName (str): location of SP3 orbit file
```

5.19.3 Member Function Documentation

5.19.3.1 getSatellite()

```
def GPSTomographyToolbox.sp3reader.SP3Reader.getSatellite (
              self,
              prn )
```

get satellite by PRN number

Parameters

prn	(str): PRN number of the satellite

Returns

(Satellite): Satellite object with the orbit of the given PRN number

5.19.3.2 getSatellites()

```
def GPSTomographyToolbox.sp3reader.SP3Reader.getSatellites (
              self )
```

get list of satellites with the orbit (generator method)

Returns

(Satellite): Satellite object with the orbit of the given PRN number

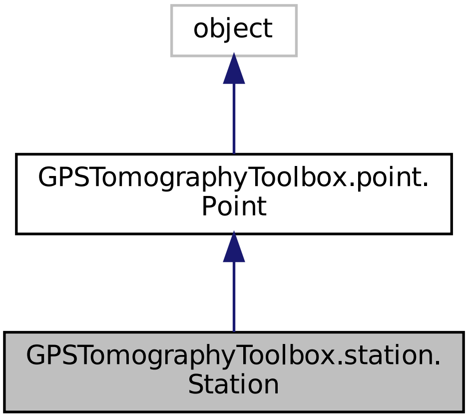
The documentation for this class was generated from the following file:

- sp3reader.py

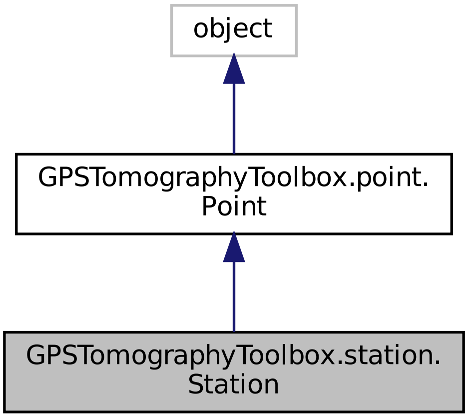
5.20 GPSTomographyToolbox.station.Station Class Reference

Station class to store and make calculations on points in cartesian and geographical coordinate system.

Inheritance diagram for GPSTomographyToolbox.station.Station:



Collaboration diagram for GPSTomographyToolbox.station.Station:



Public Member Functions

- def __init__ (self, id='', code='', coord=np.array([[0.0],[0.0],[0.0]]), type=point.XYZ, system=None)

    Station initializer.

Public Attributes

- **troposphere**

5.20.1 Detailed Description

Station class to store and make calculations on points in cartesian and geographical coordinate system.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 __init__()

```
def GPSTomographyToolbox.station.Station.__init__ (
              self,
              id = '',
              code = '',
              coord = np.array([[0.0],[0.0],[0.0]]),
              type = point.XYZ,
              system = None )
```

Station initializer.

Parameters

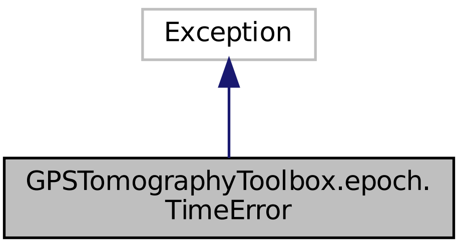
id	(str): point ID, default: ''
code	(str): point coode (Str), default: ''
coord	(numpy array (3,1)): coordinates (cartesian or geographical), default: [[0, 0, 0]]
type	(int): type of coordinate system, variable: XYZ/PLH , default: XYZ
system	(Ellipsoid object): base ellipsoid, default: None

The documentation for this class was generated from the following file:

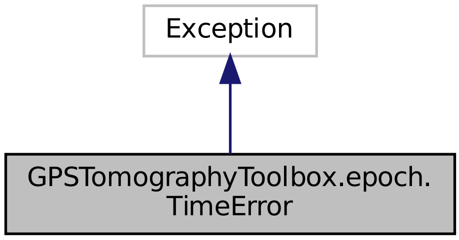
- station.py

5.21 GPSTomographyToolbox.epoch.TimeError Class Reference

Inheritance diagram for GPSTomographyToolbox.epoch.TimeError:



Collaboration diagram for GPSTomographyToolbox.epoch.TimeError:



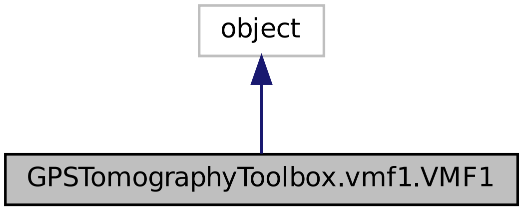
The documentation for this class was generated from the following file:

- epoch.py

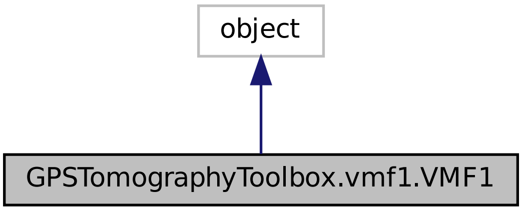
5.22 GPSTomographyToolbox.vmf1.VMF1 Class Reference

Vienna Mapping Function 1 class to calculate slant hydrostatic and wet delay at a GNSS station to any direction in a topocentric coordinate system.

Inheritance diagram for GPSTomographyToolbox.vmf1.VMF1:



Collaboration diagram for GPSTomographyToolbox.vmf1.VMF1:



Public Member Functions

- def __init__ (self, vmf1grid)

  VMF1 class initializer.
- def heightCorrection (self, e)

  get height correction
- def c_h (self, st, ep)

  Calculate the hydrostatic "c" parameter at the given station and epoch.
- def **fun_h** (self, st, e, ep)
- def **fun_h_der** (self, st, e, ep)
- def **fun_w** (self, st, e, ep)
- def **fun_w_der** (self, st, e, ep)
- def slantDelay_h (self, zd, st, alpha, e, ep, grad_n=0, grad_e=0)

  Calculate the slant hydrostatic delay concerning the troposheric gradients.
- def slantDelay_w (self, zd, st, alpha, e, ep, grad_n=0, grad_e=0)

  Calculate the slant wet delay concerning the troposheric gradients.

Public Attributes

- **vmf1grid**

Static Public Attributes

- float **a_ht** = 2.53*10**-5
- float **b_ht** = 5.49*10**-3
- float **c_ht** = 1.14*10**-3
- float **b_h** = 0.0029
- float **b_w** = 0.00146
- float **c0** = 0.062
- **c10** = np.array([0.002, 0.001])
- **c11** = np.array([0.007, 0.005])
- float **c_w** = 0.04391
- **PSZI** = np.array([np.pi, 0])

5.22.1 Detailed Description

Vienna Mapping Function 1 class to calculate slant hydrostatic and wet delay at a GNSS station to any direction in a topocentric coordinate system.

5.22.2 Section

hydrostatic "a" parameter of VMF1

5.22.3 Constructor & Destructor Documentation

5.22.3.1 __init__()

```
def GPSTomographyToolbox.vmf1.VMF1.__init__ (
              self,
              vmf1grid )
```

VMF1 class initializer.

Parameters

vmf1grid	(VMF1GridReader): parsed VMF1 grid

5.22.4 Member Function Documentation

5.22.4.1 c_h()

```
def GPSTomographyToolbox.vmf1.VMF1.c_h (
              self,
              st,
              ep )
```

Calculate the hydrostatic "c" parameter at the given station and epoch.

Parameters

st	(Station): station
ep	(Epoch): epoch

Returns

(float): hydrostatic "c" paramater of VMF1

5.22.4.2 heightCorrection()

```
def GPSTomographyToolbox.vmf1.VMF1.heightCorrection (
              self,
              e )
```

get height correction

Parameters

e	(float): elevation angle

Returns

(float): height correction in meter

5.22.4.3 slantDelay_h()

```
def GPSTomographyToolbox.vmf1.VMF1.slantDelay_h (
              self,
              zd,
              st,
              alpha,
              e,
              ep,
              grad_n = 0,
              grad_e = 0 )
```

Calculate the slant hydrostatic delay concerning the troposheric gradients.

Parameters

zd	(float): zenith hydrostatic delay
st	(Station): station
alpha	(float): azimuth angle in radians
e	(float): elevation angle in radians
ep	(Epoch): epoch
grad_n	(float): tropospheric gradient to the direction North
grad_e	(float): tropospheric gradient to the direction East

Returns

(float): slant hydrostatic delay

5.22.4.4 slantDelay_w()

```
def GPSTomographyToolbox.vmf1.VMF1.slantDelay_w (
              self,
              zd,
              st,
              alpha,
              e,
              ep,
              grad_n = 0,
              grad_e = 0 )
```

Calculate the slant wet delay concerning the troposheric gradients.

Parameters

zd	(float): zenith wet delay
st	(Station): station
alpha	(float): azimuth angle in radians
e	(float): elevation angle in radians
ep	(Epoch): epoch
grad_n	(float): tropospheric gradient to the direction North
grad_e	(float): tropospheric gradient to the direction East

Returns

(float): slant wet delay

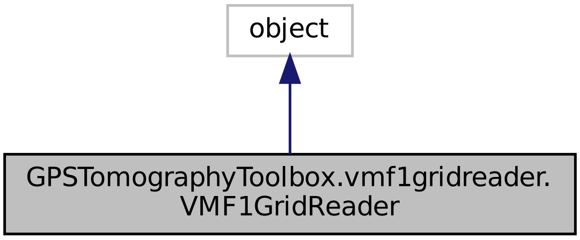
The documentation for this class was generated from the following file:

- vmf1.py

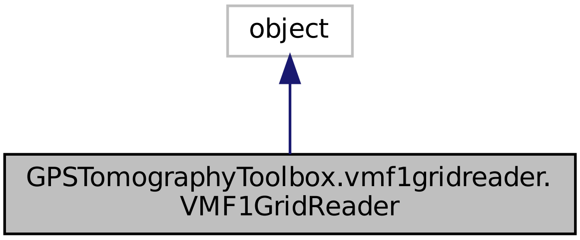
5.23 GPSTomographyToolbox.vmf1gridreader.VMF1GridReader Class Reference

VMF1GridReader class to read VMF1 (Vienna Mapping Function) grid file format file.

Inheritance diagram for GPSTomographyToolbox.vmf1gridreader.VMF1GridReader:



Collaboration diagram for GPSTomographyToolbox.vmf1gridreader.VMF1GridReader:



Public Member Functions

- def __init__ (self, fileNames, oro)
- def **getA_h** (self, st, ep)
- def **getA_w** (self, st, ep)
- def **getZhd** (self, st, ep)
- def **getZwd** (self, st, ep)

Public Attributes

- **fileNames**
- **oro**
- **grid**
- **phi**
- **lam**
- **epochs**
- **fid**
- **a_h**
- **a_w**
- **zdh**
- **zdw**

5.23.1 Detailed Description

VMF1GridReader class to read VMF1 (Vienna Mapping Function) grid file format file.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 __init__()

```
def GPSTomographyToolbox.vmf1gridreader.VMF1GridReader.__init__ (
              self,
              fileNames,
              oro )
```

```
VMF1GridReader constructor
@param fileName (str): name of VMF1 file
@param oro (str): name of orography file
```

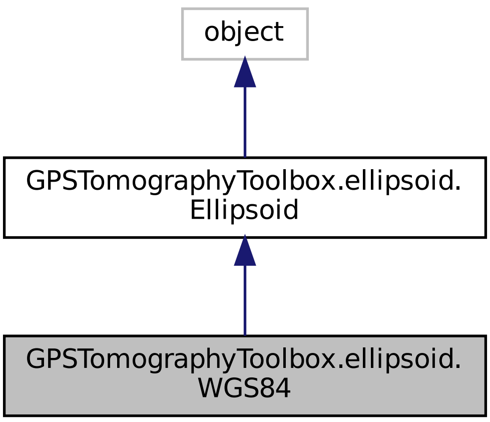
The documentation for this class was generated from the following file:

- vmf1gridreader.py

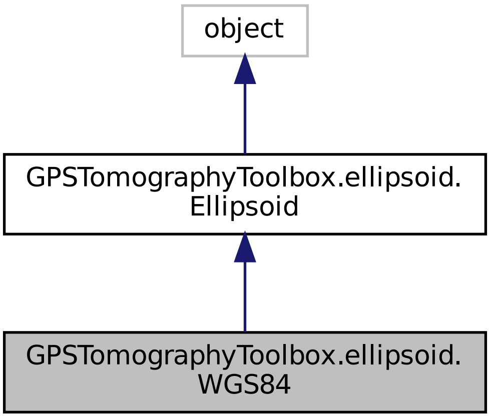
5.24 GPSTomographyToolbox.ellipsoid.WGS84 Class Reference

WGS84 class to define WGS84 ellipsoidal coordinate system.

Inheritance diagram for GPSTomographyToolbox.ellipsoid.WGS84:



Collaboration diagram for GPSTomographyToolbox.ellipsoid.WGS84:



Static Public Attributes

- float **a** = 6378137.000
- float **b** = 6356752.314

Additional Inherited Members

5.24.1 Detailed Description

WGS84 class to define WGS84 ellipsoidal coordinate system.

The documentation for this class was generated from the following file:

- ellipsoid.py

Chapter 6

File Documentation

6.1 mart.py File Reference

Wet refractivity reconstruction using Multiplicative Algebraic Reconstuction Technique.

Functions

- def GPSTomographyToolbox.mart.mart (A, b, maxIter, x0, tol)

    Wet refractivity reconstruction using Multiplicative Algebraic Reconstuction Technique.

6.1.1 Detailed Description

Wet refractivity reconstruction using Multiplicative Algebraic Reconstuction Technique.

6.1.2 Function Documentation

6.1.2.1 mart()

```
def GPSTomographyToolbox.mart.mart (
              A,
              b,
              maxIter,
              x0,
              tol )
```

Wet refractivity reconstruction using Multiplicative Algebraic Reconstuction Technique.

Parameters

A	(np.array): design matrix
b	(np.array): vector of observations
maxIter	(int): max iteration of MART algorithm
x0	(np.array): inital wet refractivity model
tol	(float): tolerance for MART algorithm

Returns

x (np.array): reconstructed wet rafractivity

iter (int): number of iteration during the procession

6.2 tomography.py File Reference

Calculate each rays' length through the tomographic grid and Slant Wet Delay using Vienna Mapping Function 1 for setting up the design matrix a measurements vector of the equation system.

Functions

- def GPSTomographyToolbox.tomography.tomography (proj, gridp, gridl, gridh, network, tropo, mapping_↩ function, ep, constellation=('G', 'R', 'E'), ignore_stations=[ ])

  Calculate each rays' length through the tomographic grid and Slant Wet Delay using Vienna Mapping Function 1 for setting up the design matrix a measurements vector of the equation system.

6.2.1 Detailed Description

Calculate each rays' length through the tomographic grid and Slant Wet Delay using Vienna Mapping Function 1 for setting up the design matrix a measurements vector of the equation system.

6.2.2 Function Documentation

6.2.2.1 tomography()

```
def GPSTomographyToolbox.tomography.tomography (
              proj,
              gridp,
              gridl,
              gridh,
              network,
              tropo,
              mapping_function,
              ep,
              constellation = ('G','R','E'),
              ignore_stations = [] )
```

Calculate each rays' length through the tomographic grid and Slant Wet Delay using Vienna Mapping Function 1 for setting up the design matrix a measurements vector of the equation system.

Parameters

proj	(GetLocal): projections class to get local coordinates from ECEF coordinates
gridp	(np.array): tomographic grid (longitude) in radians
gridl	(np.array): tomographic grid (latitude) in radians

Parameters

gridh	(np.array): tomographic grid (height) in meters
network	(Network): network object that contains all the reference stations and satellite orbits in the reference epoch
tropo	(ReadTRP): parsed troposheric delays from Benese TRP file in the reference epoch
mapping_function	(VMF1): Vienna Mapping Funtion 1 with the recent VMF1 parameters
ep	(Epoch): refernce epoch
constellations	(tuple): list of applied GNSS constellations (G => GPS, R => GLONASS, E => Galileo), diffault: ('G', 'R', 'E')
ignore_stations	(list of station IDs): list of station IDs to be ignored, default: []

Returns

A (numpy array (n,m)): design matrix (length of rays in each cell)

b (numpy array (n))): measuremnts vector (10^6*SWD values from each station to each satellite)

stations (list (n)): list of stations to the correponding rays @retrun satellites (list (n)): list of satellites to the correponding rays

elevation_azimuth (numpy array (n,2)): elevation and azimuth angles to the correponding rays

Index