soltrack

Marc van der Sluys

Nov 14, 2022

CONTENTS:

CHAPTER

ONE

SOLTRACK PACKAGE

1.1 Submodules

1.1.1 soltrack.data module

class soltrack.data.**Parameters**(*_use_degrees: bool = False*, *_use_north_equals_zero: bool = False*, *_compute_refr_equatorial: bool = True*, *_compute_distance: bool = True*)

Bases: object

Class containing SolTrack parameters/settings.

setParameters(*use_degrees=None*, *use_north_equals_zero=None*, *compute_refr_equatorial=None*, *compute_distance=None*)

This function is obsolescent and will be removed in a future version. Use set_parameters() instead.

set_parameters(*use_degrees=None*, *use_north_equals_zero=None*, *compute_refr_equatorial=None*, *compute_distance=None*)

Set the SolTrack parameters (settings).

Parameters

- **use_degrees** (*bool*) – Input (geographic position) and output are in degrees, rather than radians.
- **use_north_equals_zero** (*bool*) – Azimuth: 0 = South, pi/2 (90deg) = West -> 0 = North, pi/2 (90deg) = East.
- **compute_refr_equatorial** (*bool*) – Compute refraction-corrected equatorial coordinates (Hour angle, declination).
- **compute_distance** (*bool*) – Compute the distance to the Sun.

1.1.2 soltrack.location module

class soltrack.location.**Location**(*geo_longitude: float = 0.0*, *geo_latitude: float = 0.0*, *pressure: float = 101.0*, *temperature: float = 283.0*)

Bases: object

Class containing the geographic location to compute the Sun position for.

geo_latitude: float = **0.0**

Geographic latitude of the observer/site (>0 = northern hemisphere; radians or degrees).

geo_longitude: float = **0.0**

Geographic longitude of the observer/site (>0 = east of Greenwich; radians or degrees).

pressure: float = **101.0**

Air pressure at the site (kPa).

setLocation(*geo_longitude*, *geo_latitude*, *pressure=101.0*, *temperature=283.0*)

This function is obsolescent and will be removed in a future version. Use set_location() instead.

set_location(*geo_longitude*, *geo_latitude*, *pressure=101.0*, *temperature=283.0*)

Setter for the details of the observer/site location to compute the Sun position for.

Parameters

- **geo_longitude** (*float*) – Geographic longitude of the observer/site (>0 = east of Greenwich; radians or degrees).
- **geo_latitude** (*float*) – Geographic latitude of the observer/site (>0 = northern hemisphere; radians or degrees).
- **pressure** (*float*) – Air pressure at the site (kPa).
- **temperature** (*float*) – Air temperature at the site (K).

temperature: float = **283.0**

Air temperature at the site (K).

1.1.3 soltrack.position module

class soltrack.position.**Position**

Bases: *Parameters*

Class containing the position of the Sun and related attributes and methods.

altitude: float

Altitude of the Sun, corrected for refraction (radians)

azimuth: float

Azimuth of the Sun, corrected for refraction (radians)

computePosition()

This function is obsolescent and will be removed in a future version. Use compute_position() instead.

compute_position()

Method to compute the position of the Sun.

create_df(*utc=False*, *jd=False*, *ecl=False*, *eq=False*, *uncorr=False*, *rts_pos=False*)

Create a Pandas DataFrame with the results of the Sun position and rise/set data.

Parameters

- **utc** (*bool*) – Include utc, defaults to False.
- **jd** (*bool*) – Include Julian day, defaults to False.
- **ecl** (*bool*) – Include ecliptical coordinates, defaults to False.
- **eq** (*bool*) – Include equatorial coordinates, defaults to False.
- **uncorr** (*bool*) – Include coordinates uncorrected for refraction, defaults to False.
- **rts_pos** (*bool*) – Include the rise, transit and set positions, defaults to False.

Note that if a desired variable is not available, the request will be silently ignored.

declination: float

Declination of the Sun, corrected for refraction (radians)

distance: float

Distance Earth-Sun (AU)

hour_angle: float

Hour angle of the Sun, corrected for refraction (radians)

julian_day: float

The Julian day for the desired instant

longitude: float

Ecliptical longitude of the Sun (radians)

lt: float

The local date/time for the desired instant, if any

utc: float

The universal date/time (UTC) for the desired instant

1.1.4 soltrack.riseset module

class soltrack.riseset.**RiseSet**

Bases: *Parameters*

Class concerning the rise, transit and set times and positions of the Sun and related attributes and methods.

computeRiseSet(*rs_alt=0.0*, *accur=1e-05*, *return_datetimes=True*)

This function is obsolescent and will be removed in a future version. Use compute_rise_set() instead.

compute_rise_set(*rs_alt=0.0*, *accur=1e-05*, *return_datetimes=True*)

Compute rise, transit and set times for the Sun, as well as their azimuths/altitude.

Parameters

- **rs_alt** (*float*) – Altitude to return rise/set data for (radians; optional, default=0.0 meaning actual rise/set). Set rs_alt>pi/2 to compute transit only.
- **accur** – (float): Accuracy (rad). Default: 1e-5 rad ~ 0.14s. Don't make this smaller than 1e-16.
- **return_datetimes** (*bool*) – Return times as datetimes rather than decimal hours. Defaults to True.

Note:

- rise/set/transit times are in the LOCAL timezone used for the input (hence UTC if UTC was used).
- if rs_alt == 0.0, actual rise and set times are computed
- if rs_alt != 0.0, the routine calculates when alt = rs_alt is reached
- returns times, rise/set azimuth and transit altitude in the class riseSet

See:

- subroutine riset() in riset.f90 from libTheSky (libthesky.sf.net) for more info

---

rise_azimuth: *float*

Rise azimuth of the Sun (radians)

rise_time: *float*

Rise time of the Sun (hours LT or UTC)

set_azimuth: *float*

Set azimuth of the Sun (radians)

set_time: *float*

Set time of the Sun (hours LT or UTC)

transit_altitude: *float*

Transit altitude of the Sun (radians)

transit_time: *float*

Transit time of the Sun (hours LT or UTC)

1.1.5 soltrack.time module

class soltrack.time.**Time**

Bases: object

Class containing the date and time (in UTC) to compute the Sun position for.

now()

Return the current system time as a SolTrack time object.

Returns
Current system date and time in a SolTrack time object.

setDateAndTime(*year=2000*, *month=1*, *day=1*, *hour=12*, *minute=0*, *second=0.0*)

This function is obsolescent and will be removed in a future version. Use set_date_and_time() instead.

setDateTime(*dt_obj*, *utc=False*)

This function is obsolescent and will be removed in a future version. Use setDateTime() instead.

set_date_and_time(*year=2000*, *month=1*, *day=1*, *hour=12*, *minute=0*, *second=0.0*)

Set the SolTrack date and time using UTC year, month, day, hour, minute and second.

Parameters

- **year** (*int*) – year of date.
- **month** (*int*) – month of date.
- **day** (*int*) – day of date.
- **hour** (*int*) – hour of day (default=0).
- **minute** (*int*) – minute of time (default=0).
- **second** (*float*) – second of time (default=0).

Note: Use set_date_time() instead if you have a Python datetime object.

---

set_date_time(*dt_obj*, *utc=False*)

Set the SolTrack date and time using a (local) Python datetime object.

Parameters
dt_obj (*datetime(64)*) – Date and time in a Python datetime object (UTC if timezone naive).

Returns
Date and time in a SolTrack time object.

Return type
Time

Note: Use set_date_and_time() instead if you have year, month,day, hour, minute and second as separate variables.

1.2 Module contents

SolTrack module

SolTrack is a simple, free, fast and accurate Python package to compute the position of the Sun, as well as its rise and set times. SolTrack can be used under the conditions of the EUPL 1.2 licence. These pages contain the API documentation. For more information on the Python package, licence, source code and data files, see the [SolTrack homepage](http://soltrack.sf.net) and [Van der Sluys & Van Kan (2022)](https://arxiv.org/abs/2209.01557) (open access scientific paper with all technical details).

class soltrack.**SolTrack**(*geo_longitude*, *geo_latitude*, *use_degrees=None*, *use_north_equals_zero=None*, *compute_refr_equatorial=None*, *compute_distance=None*)

Bases: *Location*, *Time*, *Position*, *RiseSet*

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

T

U