solarenergy

Aug 01, 2021

Contents:

1 solarenergy package **1**
- 1.1 Submodules . . . 1
  - 1.1.1 solarenergy.constants module . . . 1
  - 1.1.2 solarenergy.radiation module . . . 1
- 1.2 Module contents . . . 6

2 Indices and tables **9**

Python Module Index **11**

Index **13**

CHAPTER 1

solarenergy package

1.1 Submodules

1.1.1 solarenergy.constants module

Constants for solar energy.

solarenergy.constants.**d2r = 0.017453292519943295**
Degrees to radians

solarenergy.constants.**pi = 3.141592653589793**
pi

solarenergy.constants.**pi2 = 6.283185307179586**
2 pi

solarenergy.constants.**pio2 = 1.5707963267948966**
pi/2

solarenergy.constants.**r2d = 57.29577951308232**
Radians to degrees

solarenergy.constants.**solConst = 1361.5**
Obsolescent; use sol_const instead!

solarenergy.constants.**sol_const = 1361.5**
https://en.wikipedia.org/wiki/Solar_constant

Type Solar constant

Type ~1361-1362 W/m^2 - Wikipedia

1.1.2 solarenergy.radiation module

Functions for solar energy dealing with (solar) radiation.

References

- Marc van der Sluys, Celestial mechanics in a nutshell, https://cmians.sourceforge.io (2014).

solarenergy.radiation.**airmass**(*sun_alt*, *return_value_below_horizon=False*)
Compute airmass as a function of Sun altitude.

Parameters

- **sun_alt** (*float*) – Altitude of the Sun (rad), can be an array.
- **return_value_below_horizon** (*bool*) – Return a very large value when the Sun is below the horizon, larger when the Sun is lower. This can be useful for solvers. Default: False.

Returns Airmass at sea level (AM~1 if the Sun is in the zenith, AM~38 near the horizon).

Return type float

solarenergy.radiation.**clearsky_bird**(*sun_alt*, *i_ext=1353*, *sun_dist=1*, *press=1013*, *uo=0.34*, *uw=1.42*, *ta5=0.2661*, *ta3=0.3538*, *ba=0.84*, *k1=0.1*, *rg=0.2*)
A simplified clear-sky model for direct and diffuse insolation on horizontal surfaces.

A.k.a. as "the Bird model".

This function is adapted from the libTheSky Fortran implementation (libthesky.sf.net).

See Bird & Hulstrom, A simplified clear-sky model for direct and diffuse insolation on horizontal surfaces, SERI/TR-642-761 (1981).

Note that the value of Taa does not agree with tabulated values from the paper, and hence neither do dependent values (except for AM~1). When I substitute their values for Taa, everything matches perfectly. Error in their formula, or (hopefully!) in their table?

Parameters

- **sun_alt** (*float*) – Sun altitude above the horizon (rad)
- **i_ext** (*float*) – Extraterrestrial radiation (at the top of the atmosphere; AM0; W/m^2 - optional, default: 1353 (1361.5))
- **sun_dist** (*float*) – Sun distance (AU - optional, default: 1)
- **press** (*float*) – Air pressure at the observer's site, corrected for altitude (hPa - optional, default: 1013)
- **uo** (*float*) – Ozone abundance in a vertical column (cm - optional, default: 0.34)
- **uw** (*float*) – Percipitable water-vapor abundance in a vertical column (cm - optional, default: 1.42)
- **ta5** (*float*) – Aerosol optical depth from surface in vertical path at 500 nm (optional, default: 0.2661)
- **ta3** (*float*) – Aerosol optical depth from surface in vertical path at 380 nm (optional, default: 0.3538)

- **ba** (*float*) – Aerosol forward-scattering ratio (optional, 0.82-0.86, default: 0.84)
- **k1** (*float*) – Aerosol-absorptance constant (optional, rural: 0.0933, urban: 0.385, default: 0.1)
- **rg** (*float*) – Ground albedo (optional, fraction - default: 0.2)

Returns

Tuple containing (rv1, rv2):

- i_tot (float): Total insolation on a horizontal surface (W/m^2)
- i_dir (float): Direct (beam) insolation on a horizontal surface (W/m^2)
- i_dif (float): Diffuse insolation on a horizontal surface (W/m^2)
- i_gr (float): Ground-reflection insolation from a horizontal surface (W/m^2)

Return type tuple (float,float,float,float)

`solarenergy.radiation.`**`computeSunPos`**(*geo_lon*, *geo_lat*, *year*, *month*, *day*, *hour*, *minute=0*, *second=0*, *timezone='UTC'*, *debug=False*)

Obsolescent wrapper for sun_position_from_date_and_time(). Use that function instead!

`solarenergy.radiation.`**`cosAngleSunPanels`**(*sp_az*, *sp_incl*, *sun_az*, *sun_alt*)

Obsolescent wrapper for cos_angle_sun_panels(). Use that function instead!

`solarenergy.radiation.`**`cos_angle_sun_panels`**(*sp_az*, *sp_incl*, *sun_az*, *sun_alt*)

Compute the cosine of the angle between the orientation vector of the solar panels and the position vector of the Sun.

This is the cosine of the angle under which the direct sunlight hits the solar panels. Multiply it with the DNI to obtain the direct insolation on the panels. See Celestial mechanics in a nutshell, Sect. 4.3: Insolation on an inclined surface (http://CMiaNS.sf.net).

Parameters

- **sp_az** (*float*) – Azimuth in which the solar panels are facing (rad; e.g. north or south = 0, same as sun_az).
- **sp_incl** (*float*) – Inclination ('zenith angle') of the solar panels w.r.t. the horizontal (rad).
- **sun_az** (*float*) – Azimuth of the Sun (rad; e.g. north or south = 0, same as sp_az).
- **sun_alt** (*float*) – Altitude of the Sun (rad).

Returns

The cosine between the normal vector of the solar panels and the position vector of the Sun (rad). Note that this value is zero (indicating radiation from behind the panels) or positive.

Return type float

solarenergy.radiation.**diffuseRad_from_globalRad_sunshine**(*glob_horiz*, *sun_frac*, *sun_alt*, *i_ext=1361.5*)

Obsolescent wrapper for diffuse_radiation_from_global_radiation_and_sunshine(). Use that function instead!

solarenergy.radiation.**diffuse_radiation_from_global_radiation_and_sunshine**(*glob_ho* *sun_fra* *sun_alt* *i_ext=1*

Compute the diffuse horizontal radiation from the global horizontal radiation, the fraction of sunshine and the Sun altitude.

Parameters

- **glob_horiz** (*float*) – Global horizontal radiation (W/m2).
- **sun_frac** (*float*) – Fraction of sunshine (e.g. fraction of cloud cover) (-; 0-1).
- **sun_alt** (*float*) – Sun altitude above the horizon (rad).
- **i_ext** (*float*) – Extraterrestrial radiation (W/m2). Defaults to solar constant.

Returns

Tuple containing (dif_horiz, beam_horiz, beam_norm):

- dif_horiz (float): Diffuse horizontal radiation = DHI (W/m2).
- beam_horiz (float): Beam (direct) horizontal radiation = BHI (W/m2).
- beam_norm (float): Beam (direct) normal radiation = BNI = DNI (W/m2).

Return type tuple (float,float,float)

solarenergy.radiation.**diffuse_radiation_projection_Perez87**(*doy*, *alt*, *surf_incl*, *theta*, *beam_norm*, *dif_horiz*)

Obsolescent wrapper for diffuse_radiation_projection_perez87(). Use that function instead!

solarenergy.radiation.**diffuse_radiation_projection_perez87**(*doy*, *sun_alt*, *surf_incl*, *theta*, *beam_norm*, *dif_horiz*)

Compute diffuse radiation on an inclined surface using the 1987 Perez model

This function is adapted from the libTheSky Fortran implementation (libthesky.sf.net).

See Perez et al. Solar Energy Vol. 39, Nr. 3, p. 221 (1987) - references to equations and tables are to this paper. Most equations can be found in the Nomenclature section at the end of the paper (p.230). I use a and c here, not b and d.

Parameters

- **doy** (*int*) – Day of year (Nday)
- **sun_alt** (*float*) – Altitude of the Sun (radians, may be an array)

- **surf_incl** (*float*) – Surface inclination wrt horizontal (radians) - 0 = horizontal, pi/2 = vertical
- **theta** (*float*) – Angle between surface normal vector and Sun position vector (radians, may be an array)
- **beam_norm** (*float*) – Beam (direct) normal radiation = DNI (W/m2; in the direction of the Sun, may be an array)
- **dif_horiz** (*float*) – Diffuse radiation on a horizontal surface = DHI (W/m2, may be an array)

Returns Diffuse irradiation on the inclined surface (W/m2) (may be an array)

Return type float

solarenergy.radiation.**extinctionFactor**(*airmass*, *return_value_below_horizon=False*)

Obsolescent wrapper for extinction_factor(). Use that function instead!

solarenergy.radiation.**extinction_factor**(*airmass*, *return_value_below_horizon=False*)

Compute the atmospheric extinction factor for sunlight from the air mass.

Parameters

- **airmass** (*float*) – Airmass at sea level (AM~1 if the Sun is in the zenith, AM~38 near the horizon), can be an array.
- **return_value_below_horizon** (*bool*) – Return a very large value when the Sun is below the horizon, larger when the Sun is lower. This can be useful for solvers. Default: False.

Returns

The extinciton factor for sunlight in the atmosphere. Divide the extraterrestrial (AM) radiation (or, if unknown, solar constant) by this number to obtain the DNI.

Return type float

solarenergy.radiation.**sun_position_from_date_and_time**(*geo_lon*, *geo_lat*, *year*, *month*, *day*, *hour*, *minute=0*, *second=0*, *timezone='UTC'*, *debug=False*)

Compute the Sun local position (azimuth, altitude and distance) for the given geographical location and date and time (ymd, hms) using SolTrack.

Parameters

- **geo_lon** (*float*) – Geographic longitude to compute the Sun position for (rad).
- **geo_lat** (*float*) – Geographic latitude to compute the Sun position for (rad).
- **year** (*int*) – Year (CE) to compute the Sun position for.
- **month** (*int*) – Month to compute the Sun position for.
- **day** (*int*) – Day of month to compute the Sun position for.

- **hour** (*int*) – Hour of day to compute the Sun position for (local time!).
- **minute** (*int*) – Minute to compute the Sun position for (optional; default = 0).
- **second** (*int*) – Second to compute the Sun position for (optional; default = 0).
- **timezone** (*str*) – Timezone for which date and time are provided (optional; default = 'UTC').
- **debug** (*bool*) – Switch to write detailed output to stdout (optional; default = False).

Returns

Tuple containing (azimuth, altitude, distance):

- azimuth (float): Azimuth of the Sun (rad; south = 0 on the northern hemisphere).
- altitude (float): Altitude of the Sun (rad).
- distance (float): Distance Sun-Earth (AU).

Return type tuple (float,float,float)

`solarenergy.radiation.`**`sun_position_from_datetime`**(*geo_lon*, *geo_lat*, *date_time*, *debug=False*)

Compute the Sun local position (azimuth, altitude and distance) for the given geographical location and a Python datetime using SolTrack.

Parameters

- **geo_lon** (*float*) – Geographic longitude to compute the Sun position for (rad).
- **geo_lat** (*float*) – Geographic latitude to compute the Sun position for (rad).
- **date_time** (*datetime*) – Date and time to compute the Sun position for (timezone aware and/or UTC, i.e. must be UTC if timezone naive. CHECK: must be UTC if a list or (numpy) array?).
- **debug** (*bool*) – Switch to write detailed output to stdout (optional; default = False).

Returns

Tuple containing (azimuth, altitude, distance):

- azimuth (float): Azimuth of the Sun (rad; south = 0 on the northern hemisphere).
- altitude (float): Altitude of the Sun (rad).
- distance (float): Distance Sun-Earth (AU).

Return type tuple (float,float,float)

1.2 Module contents

SolarEnergy module

SolarEnergy contains a Python module to do simple modelling in the field of solar energy. The code is being developed by Marc van der Sluys of the department of Astrophysics at the Radboud University Nijmegen, the Netherlands and the department of Sustainable energy of the HAN University of Applied Sciences in Arnhem, the Netherlands. SolarEnergy can be used under the conditions of the GPLv3 licence. These pages contain the API documentation. For more information on the Python package, licence, source code and data files, see the SolarEnergy GitHub page.

The SolarEnergy code is based on the libTheSky Fortran library. Information on the theory behind this code can be found in the document Celestial mechanics in a nutshell.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

solarenergy, 6
solarenergy.constants, 1
solarenergy.radiation, 1

Index

A

airmass() (*in module solarenergy.radiation*), 2

C

clearsky_bird() (*in module solarenergy.radiation*), 2
computeSunPos() (*in module solarenergy.radiation*), 3
cos_angle_sun_panels() (*in module solarenergy.radiation*), 3
cosAngleSunPanels() (*in module solarenergy.radiation*), 3

D

d2r (*in module solarenergy.constants*), 1
diffuse_radiation_from_global_radiation_and_sunshine() (*in module solarenergy.radiation*), 4
diffuse_radiation_projection_Perez87() (*in module solarenergy.radiation*), 4
diffuse_radiation_projection_perez87() (*in module solarenergy.radiation*), 4
diffuseRad_from_globalRad_sunshine() (*in module solarenergy.radiation*), 3

E

extinction_factor() (*in module solarenergy.radiation*), 5
extinctionFactor() (*in module solarenergy.radiation*), 5

P

pi (*in module solarenergy.constants*), 1
pi2 (*in module solarenergy.constants*), 1
pio2 (*in module solarenergy.constants*), 1

R

r2d (*in module solarenergy.constants*), 1

S

sol_const (*in module solarenergy.constants*), 1
solarenergy (*module*), 6
solarenergy.constants (*module*), 1
solarenergy.radiation (*module*), 1
solConst (*in module solarenergy.constants*), 1
sun_position_from_date_and_time() (*in module solarenergy.radiation*), 5
sun_position_from_datetime() (*in module solarenergy.radiation*), 6