Food and Agriculture Organization of the United Nations

WaPOR Data Manual

Evapotranspiration v2

Final Draft 2.2

eleaf FEED THE WORLD

vito

ITC

UNIVERSITY OF TWENTE.

WATERWATCH FOUNDATION

Preface

Achieving Food Security in the future while using water resources in a sustainable manner will be a major challenge for current and future generations. Increasing population, economic growth and climate change all add to increasing pressure on available resources. Agriculture is a key water user and careful monitoring of water productivity in agriculture and exploring opportunities to increase it is required. Improving water productivity often represents the most important avenue to cope with increased water demand in agriculture. Systematic monitoring of water productivity through the use of Remote Sensing techniques can help to identify water productivity gaps and evaluate appropriate solutions to close these gaps.

The FAO portal to monitor Water Productivity through Open access of Remotely sensed derived data (WaPOR) provides access to 10 years of continued observations over Africa and the Near East. The portal provides open access to various spatial data layers related to land and water use for agricultural production and allows for direct data queries, time series analyses, area statistics and data download of key variables to estimate water and land productivity gaps in irrigated and rain fed agriculture.

WaPOR Version 2 became available starting from June 2019. This manual explains the processing chain for the production of the Evapotranspiration data components distributed through WaPOR portal. It can be used in combination with the WaPOR Database methodology documents.

This version 2.2 document differs from version 2.1 for additional clarification on i) LST retrieval from Landsat products and ii) Proba-V / Sentinel 2 calibration.

Acknowledgements

FAO, in partnership with and funded by the Government of the Netherlands, is developing a programme to monitor and improve the use of water in agricultural production. This document is part of the final output of the programme: the operational methodology for an open-access database to monitor land and water productivity.

The methodology was developed by the FRAME[1] consortium, consisting of eLEAF, VITO, ITC, University of Twente and Waterwatch foundation, commissioned by and in partnership with the Land and Water Division of FAO.

Substantial contributions to the eventual methodology were provided during the first Methodology Review workshop, held in FAO Headquarters in October 2016, during the second *beta* methodology review workshop in January 2018 and during a project review meeting held in March 2019. Participants in these workshops were: Pauline Jaquot, Henk Pelgrum, Karin Viergever, Maurits Voogt and Steven Wonink (eLEAF), Sergio Bogazzi, Amy Davidson, Jippe Hoogeveen, Michela Marinelli, Karl Morteo, Livia Peiser, Pasquale Steduto, Erik Van Ingen (FAO), Megan Blatchford, Chris Mannaerts, Sammy Muchiri Njuki, Hamideh Nouri, Zeng Yijan (ITC), Lisa-Maria Rebelo (IWMI), Job Kleijn (Ministry of Foreign Affairs, the Netherlands), Wim Bastiaanssen, Gonzalo Espinoza, Marloes Mul, Jonna Van Opstal (IHE Delft), Herman Eerens, Sven Gilliams, Laurent Tits (VITO) and Koen Verberne (Waterwatch foundation).

[1]For more information regarding FRAME, contact eLEAF (http://www.eleaf.com/). Contact persons. FRAME project manager: Steven Wonink (steven.wonink@eleaf.com). Managing Director: Maurits Voogt (maurits.voogt@eleaf.com)

Contents

Preface 2
Acknowledgements 3
Abbreviations 6
1 Introduction 7
2 Overview of the processing chain 7
2.1 Actual E, T and I 7
2.2 RET 8
3 Input data 9
3.1 Sensor input data 12
3.1.1 MODIS TERRA and AQUA 12
3.1.2 Proba-V 13
3.1.3 Sentinel-2 14
3.1.4 Landsat 15
3.2 Model input data 16
3.2.1 MSG 16
3.2.2 GEOS-5 17
3.2.3 CHIRPS 17
3.3 Static input data 18
3.3.1 Elevation, slope and aspect (topography) 18
3.3.2 Land Cover 18
4 Pre-processing steps 19
4.1 NDVI 19
4.2 Surface Albedo 21
4.3 Weather data 22
4.4 Solar radiation 23
4.5 Land Surface Temperature 23
5 Description of evapotranspiration data component functions 24
5.1 Reference Evapotranspiration 24
5.1.1 Meteorological data 24
5.2 ETlook.saturated_vapour_pressure_average 26
5.2.1 Reference Evapotranspiration 29
5.3 Soil Moisture 33
5.3.1 Air Density 33

5.3.2 Atmospheric emissivity 36
5.3.3 Bare soil maximum temperature 37
5.3.4 Full canopy maximum temperature 42
5.3.5 Soil Moisture 46
5.4 Interception 49
5.4.1 Interception 50
5.5 Transpiration........ 51
5.5.1 Leaf Area Index........ 51
5.5.2 Surface Roughness 53
5.5.3 Stress factors 56
5.5.4 Net radiation 59
5.5.5 Transpiration 63
5.6 Evaporation........ 71
5.6.1 Soil Heat Flux........ 71
5.6.2 Evaporation 75
5.7 ETLook functions (ETLook API REFERENCE) 81
5.7.1 Instantaneous Radiation (ETLook.clear_sky_radiation) 82
5.7.2 Evapotranspiration (ETLook.evapotranspiration) 87
5.7.3 Vegetation Cover (ETLook.leaf)........ 88
5.7.4 Meteorology (ETLook.meteo) 88
5.7.5 Net Available Energy (ETLook.radiation) 101
5.7.6 Roughness (ETLook.roughness)........ 101
5.7.7 Solar radiation (ETLook.solar_radiation) 102
5.7.8 Plant stress (ETLook.stress)........ 107
5.7.9 Canopy and Soil Resistance (ETLook.resistance)........ 108
5.7.10 Neutral Atmosphere (ETLook.neutral) 108
5.7.11 Unstable Atmosphere (ETLook.unstable) 109
5.7.12 Soil Moisture (ETLook.soil_moisture)........ 114
6 Function parameters........ 118
References 120

Abbreviations

AET	Actual Evapotranspiration
ET	Evapotranspiration
E	Evaporation
I	Interception
LST	Land Surface Temperature
MODIS	MODerate-resolution Imaging Spectroradiometer
MOS	Maximum of Season
NDVI	Normalised Difference Vegetation Index
NIR	Near Infrared
NRT	Near Real Time
RET	Reference Evapotranspiration
SLC	Scan Line Corrector (on Landsat 7)
SMC	Soil Moisture Content
T	Transpiration
TIR	Thermal Infrared
VNIR	Visible and Near Infrared

1 Introduction

This document provides a detailed description of the processing chain applied for the production of the Evapotranspiration data components distributed through the WaPOR portal. Whereas the level-specific WaPOR methodology documents set out the theory that underlies the applied methodology, this document provides details on the input data sources used at all levels and sets out the processing chain for the production of the evapotranspiration data components Evaporation (E), Transpiration (T), Interception (I) and Reference Evapotranspiration (RET).

The WaPOR methodology for producing evapotranspiration data can therefore be found in the WaPOR methodology document, which should be used in conjunction with this data manual. To avoid confusion, in this data manual document we use the same terminology used in the methodology documents to refer to the different data components and dataset levels. Therefore 'Level 1' denotes the continental dataset at 250 m resolution, 'Level 2' denotes the national dataset at 100m resolution and 'Level 3' denotes the sub-national dataset at 30 m resolution. Furthermore, 'intermediate data components' refer to datasets that are created during pre-processing steps and which are used as input to the final processing of the evapotranspiration data components. Whereas data components (such as E, T, I and RET) are available for download from WaPOR, intermediate data components are not available on WaPOR.

The WaPOR data components are mainly derived from freely available remote sensing satellite and other data sources and can be produced using open source software and tools. The WaPOR database manual specifies the input data sources and sets out how the processing is done.

Section 2 provides an overview of the processing chain for the evapotranspiration data components, with detail on the different input data sources used at the 3 different levels given in section 3. The pre-processing steps applied for the production of intermediate data components that are used as input to the final processing is provided in section 4. Finally, section 5 gives a detailed description of the processing chain and functions used for the production of the evapotranspiration data components.

2 Overview of the processing chain

2.1 Actual E, T and I

The evapotranspiration data components (E, T, I) are produced using the same processing chain at all resolution levels. The data are delivered at a dekadal basis at all levels, and are additionally available on WaPOR at seasonal level for levels 2 and 3 and at annual basis for level 1.

The full processing chain starts with input data sources which can be either used directly as input to the processing chain, or during the pre-processing phase as input to produce intermediate data components that are in turn used as inputs to the processing chain. Figure 1 presents a flow chart that shows the different input datasets, both static external data and intermediate data components that are needed to produce the evapotranspiration data components (E, T, and I). Details of the input data sources can be found in section 3 and the pre-processing steps for producing intermediate data components are given in section 4. The final processing is described in detail in section 5.

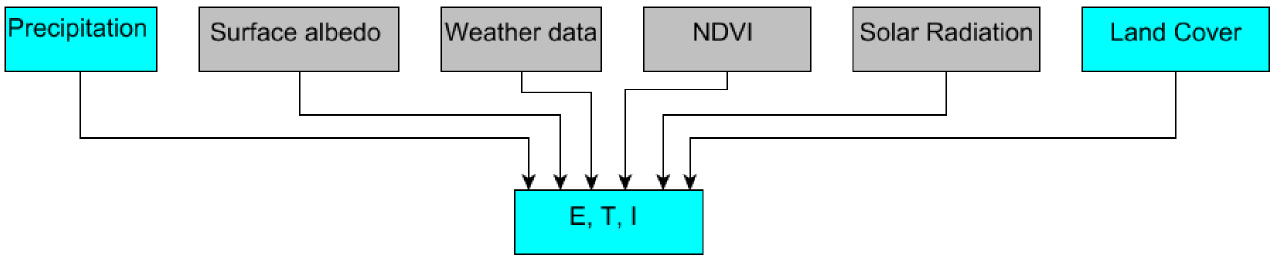


Figure 1: (Intermediate) data components that are used as input data for the production of the three evapotranspiration data components. The grey boxes represent intermediate data components that convert external data into standardised input. Blue boxes represent data components that are distributed through WaPOR.

2.2 RET

The reference evapotranspiration data component (RET) is produced at a spatial resolution of 20 km. The data are produced on a daily basis. Figure 2 shows the input data components for the production of reference evapotranspiration (RET). Details for the input data sources can be found in section 3 and the pre-processing steps for producing intermediate data components are given in section 4. The final processing is described in detail in section 5.

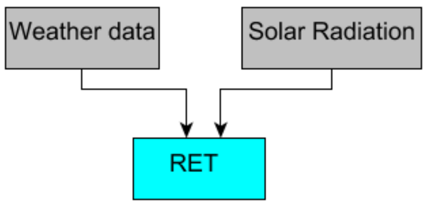


Figure 2: (Intermediate) data components that are used as input data for the production of reference evapotranspiration data component. Grey boxes represent intermediate data components that convert external data into standardised input. The blue box represents the RET data component that is distributed through WaPOR.

3 Input data

This section sets out the input data required to calculate evapotranspiration at different levels. These are required for calculating a number of variables (see Table 1) that are needed to solve the equations for the production of evapotranspiration data components.

Table 1: Variables required to calculate the evapotranspiration data components

Term	Variable	Unit	Range	Description	Temporal resolution
α_0	Surface albedo	-	0-1	Used to calculate net radiation	Dekadal
NDVI	Normalized Difference Vegetation Index	-	-1-1	Characterises vegetation condition to differentiate between evaporation and transpiration	Dekadal
R_s	Solar radiation	Wm^{-2}	0-500	Incoming solar radiation	Daily
T_s	Land Surface Temperature	K	270-330	Used to calculate relative soil moisture content	Daily
T_a	Air Temperature	K	270-320	For calculation of long wave radiation	Daily
ϕ	Specific Humidity	%	0-100	For calculation of vapour pressure	Daily
u_{obs}	Wind speed	ms^{-1}		For calculation of aerodynamic resistance	Daily
P	Precipitation	mm	0-100	For calculation of interception	Daily
$\Delta T_{a,year}$	Yearly air temperature amplitude	C	0-30	For calculation of soil heat flux	Static
$T_{opt,year}$	Yearly air temperature optimum	C	0-30	For calculation of soil heat flux	Static
$r_{soil,min}$	Minimum soil resistance	sm^{-1}		Soil type specific variable to calculate soil surface resistance	Static
$r_{canopy,min}$	Minimum stomatal resistance	sm^{-1}		Land cover specific variable to calculate canopy surface resistance	Static
z_{obs}	Observation height	M		Land cover specific variable to calculate surface roughness	Static
φ, λ	Latitude, Longitude	rad		Used for radiation calculations	Static
z	Elevation	m		Used to correct for meteorological effects in mountainous areas, and to retrieve surface characteristics	Static
Δ	Slope	rad	0-π	Used to calculate local solar radiation	Static
α	Aspect	rad	0-2π	Used to calculate local solar radiation	Static
lw_{slope}	Longwave radiation slope			Variable in FAO-56. Globally calibrated parameter (see ETLook.radiation.longwave_radiation_fao in section 6.4.4)	Static
lw_{offset}	Longwave radiation offset			Variable in FAO-56. Globally calibrated parameter (see ETLook.radiation.longwave_radiation_fao in section 6.4.4)	Static

The variables listed in Table 1 are derived from various types of input data sources. Satellite-based sensor data is discussed in detail in section 3.1, whilst model data (e.g. weather data) is set out in section 3.2 and static data (e.g. elevation) is discussed in section 3.3. Table 2 to 4 specify the input data sources used at levels 1, 2 and 3 respectively. The table also indicates whether the input data source is obtained from satellite sensor, model or static data sources.

Table 2: Input data sources for the production of evapotranspiration data components (E, T, and I) and Reference ET at Level 1

Input data components	Type of input	Sensor	Data product	Comment
Precipitation	Model		CHIRPS v2, CHIRP	
Surface albedo	Sensor	MODIS	MOD09GA, MOD09GQ	
Weather data (temp, specific humidity, wind speed, air pressure, aerosol optical depth)	Model		MERRA/GEOS-5	MERRA used prior to start of GEOS-5 (21-2-2014)
NDVI	Sensor	MODIS	MOD09GQ	
Land Surface Temperature	Sensor	MODIS	MOD11A1, MYD11A1	Used to derive Soil Moisture Stress
Elevation, slope and aspect	Static		SRTM	Elevation, slope and aspect are derived from the DEM
Transmissivity	Model	MSG		Transmissivity is derived from MSG shortwave radiation products
Land Cover	Static		WaPOR L1 Land Cover Classification	If L1 LC was not yet available, preliminary LC obtained from ESA GlobCover.

Table 3: Input data sources for the production of evapotranspiration data components (E, T, and I) at Level 2

Input data components	Type of input	Sensor	Data product	Comment
Precipitation	Model		CHIRPS v2, CHIRP	
Surface albedo	Sensor	Proba-V		PROBA-V data are available from March 2014, for earlier dates the Level 1 Surface albedo based on MODIS MOD09GQ, MOD09GA is resampled to 100m
Weather data (temp, specific humidity, wind speed, air pressure, aerosol optical depth)	Model		MERRA/GEOS-5	MERRA used prior to start of GEOS-5 (21-2-2014)
NDVI	Sensor	Proba-V		PROBA-V data are available from March 2014, data for earlier dates uses MODIS MOD09GQ, resampled to 100m
Land Surface Temperature	Sensor	MODIS	MOD11A1, MYD11A1	Used to derive Soil Moisture Stress

Elevation, slope and aspect	Static		SRTM	Elevation, slope and aspect are derived from the DEM
Transmissivity	Model	MSG		Transmissivity is derived from MSG shortwave radiation products
Land Cover	Static		WaPOR L2 Land Cover Classification	If L2 LC was not yet available, preliminary LC obtained from ESA GlobCover.

Table 4: Input data sources for the production of evapotranspiration data components (E, T, and I) at Level 3

Input data components	Type of input	Sensor	Data product	Comment
Precipitation	Model		CHIRPS v2, CHIRP	
Surface albedo	Sensor	Landsat 5 TM Landsat 7 ETM+ Landsat 8 OLI	L1TP	
Weather data (temp, specific humidity, wind speed, air pressure, aerosol optical depth)	Model		MERRA/GEOS-5	MERRA used prior to start of GEOS-5 (21-2-2014)
NDVI	Sensor	Landsat 5 TM Landsat 7 ETM+ Landsat 8 OLI	L1TP	
Land Surface Temperature	Sensor	Landsat 5 TM Landsat 7 ETM+ Landsat 8 OLI	L1TP	Used to derived Soil Moisture Stress
Elevation, slope and aspect	Static		SRTM	Elevation, slope and aspect are derived from the DEM
Transmissivity	Model	MSG		Transmissivity is derived from MSG shortwave radiation products
Land Cover	Static		WaPOR L3 Land Cover Classification	If L3 LC was not yet available, preliminary LC obtained from ESA CCI 20m LC (for 2016).

3.1 Sensor input data

3.1.1 MODIS TERRA and AQUA

NASA’s Moderate-resolution Imaging Spectroradiometer (MODIS) sensor on board of the TERRA and AQUA platforms is the primary remote sensing data source for optical data at 250m resolution used at Level 1. It covers the entire Earth on a near-daily basis, recording in 36 bands:

- At 250m (0.00223^{o}) resolution: 2 bands (Red/NIR).
- At 500m (0.00446^{o}) resolution: 5 bands in the shortwave range.
- At 1km (0.00892^{o}) resolution: 29 bands in the full spectrum (shortwave to TIR).

Having been active since early 2000, MODIS has generated a large archive with historical remote sensing data covering the whole project period (from 2009 onward). The system is mounted on two satellites (TERRA and AQUA) which are exact copies of each other and can therefore achieve double coverages for large parts of Africa. This makes it a stable and consistent data source for Level 1 data components.

Purpose of data

MODIS TERRA and AQUA data is used for the production of the following data components:

- NDVI composites for Level 1
- Surface albedo for Level 1
- Soil moisture stress for Level 1 and 2 (based on the Thermal infrared bands at 1km)

Acquired data

A host of data products are generated by NASA on the basis of MODIS observations, including surface albedo, NDVI, fAPAR and NPP. However these derived products do not meet the required temporal and spatial resolution. Therefore basic data is acquired to generate these products with the specifications needed for the WaPOR database. Calculating these data products within the project also allows for greater consistency across the three different levels which, in many cases, use different input datasets (e.g. MODIS, Proba-V and Landsat). Specifics on the MODIS data acquired for the production of evapotranspiration data at Level 1 is as follows:

- **MxD11A1:** Land Surface Temperature and Emissivity (LST/E) products provide per-pixel temperature and emissivity values on a daily basis at 1 km spatial resolution. It also includes a quality rating, used for automated cloud-masking. This is a processing-level product comprising calibrated and atmospherically corrected images, requiring no additional atmospheric and radiometric corrections. The data is provided in the form of “tiles” of roughly 10°x10° mapped to the equal-area, sinusoidal projection. The files are in EOS-HDF5 format.

- **M*x*D09GQ**: Daily composites of TOC-reflectances in Red and NIR, plus some ancillary information, all at 250m resolution
- **M*x*D09GA**: These files contain two types of information:
  - At 1km resolution: The observation angles and the status information (clear, cloud, snow, land/sea, errors, etc.) used for quality control.
  - At 500m resolution: The reflectances in the 7 shortwave bands and some other information. These data are used for the calculation of the surface albedo, after resampling to 250m. Data is resampled by splitting each 500m pixel into four (2x2) identical 250m pixels.

Table 5: Overview of the 7 MODIS bands

MODIS Terra Band	Bandwidth (nm)	Resolution (m)
1	620 – 670	250
2	841 – 876	250
3	459 – 479	500
4	545 – 565	500
5	1230 – 1250	500
6	1628 – 1652	500
7	2105 - 2155	500

Source of the data

All MODIS data are freely available from the Land Processes Distributed Active Archive Center (LP-DAAC), located at USGS in Sioux Falls, South Dakota (https://lpdaac.usgs.gov/).

Challenges

Since the MODIS sensors on both Terra (1999) and Aqua (2002) satellites have been operational for more than 15 years there is a risk that the MODIS sensors will stop working unexpectedly.

Alternative sensors

In case of failure of the MODIS Terra or Aqua sensor, the following back-ups could be used, listed in order of importance:

- Sentinel-3, launched in 2016, with its on-board sensors OLCI and SLSTR, can provide information on land reflectance and land surface temperature in resolutions of 300 and 1000m respectively.
- Visible Infrared Imaging Radiometer Suite (VIIRS), launched in 2011 as a successor to AVHRR and MODIS, with a spatial resolution of 375m for spectral and thermal infrared bands. The spatial resolution of the thermal infrared bands suitable to create Land Surface Temperature is a major advantage of this sensor.

3.1.2 Proba-V

The Proba-V sensor is used as the primary source for the 100m resolution optical data used at Level 2. It collects global imagery in four spectral bands (Blue, Red, NIR, SWIR) and in three resolutions:

- 1km (0.00892^o) resolution with near-daily revisit time
- 300m (0.002976 ^{o}) resolution with near-daily revisit time
- 100m (0.0000992^o) resolution with a revisit time of approximately 5 days.

Proba-V data at 100m resolution is used as primary source for the derivation of the different data components at Level 2. The system was launched in May 2013, but the 100m data is only available from March 2014.

Purpose of data

Proba-V data is used for the production of the following data components:

- NDVI composites for Level 2
- Surface albedo for Level 2

Acquired data

Proba-V S5-TOC products are used. These 5-daily composites are calibrated, atmospherically-corrected and mapped to Lon/Lat at resolution of 0.000992°. The product consists of a number of data layers: 4 reflectances, 4 angles (sun/view zenith/azimuth), status map (indication of deviate observations: cloud, snow, water, error).
The format is either HDF4/5 or GeoTIFF, distributed in 10° x 10° tiles.

Source of the data

For the WaPOR processing, VITO's subgroup CVB is responsible for the pre-processing (calibration, atmospheric correction, mapping to Geographic Lon/Lat) and the distribution of all the data via the portal http://www.vito-eodata.be.

Challenges

Proba-V only started delivering 100m resolution data in March 2014. Prior to this date no free 100m resolution data is available. The gap in 100m resolution data for the historical data has been filled by resampling MODIS data. This affects the data quality as more spatial variation is visible at 100m resolution than at 250m resolution.

Alternative sensors

No historical data is available at 100m resolution prior to March 2014 and no other alternative source is available at 100m resolution. Resampling from lower (MODIS) resolution is the only alternative for the period prior to March 2014. Resampling from a higher resolution is an alternative from 2017 onwards with both Sentinel-2A and 2B as data sources providing a suitable revisit time.

3.1.3 Sentinel-2

The Sentinel-2 satellites are the secondary source for 100m optical data. As the Proba-V satellite will be decommissioned in April 2020, the 100m data products will be produced based on Sentinel 2. The switch from Proba-V to Sentinel-2 input data will start from the first dekad of 2020.

Sentinel-2 is a European wide-swath, high-resolution, multi-spectral imaging mission. Currently, two satellites are in orbit (Sentinel-2A and Sentinel-2B), flying in the same orbit but with a 180 degrees phase difference. The MSI sensor samples 13 spectral bands, with four bands at 10m, six bands at 20m and three bands at 60m spatial resolution. Sentinel-2A has been in orbit since 23 June 2015, and Sentinel-2B launched on 7 March 2017, together providing a five-daily global coverage.

Purpose of data

Sentinel-2 data is used for the production of the following data components:

- NDVI/fAPAR composites for Level 2
- Surface albedo for Level 2

Acquired data

The Sentinel-2 data used is level-2A, representing Bottom-Of-Atmosphere reflectances in cartographic geometry. These images are thus already atmospherically corrected with the Sen2Cor atmospheric correction methodology. These data are provided in 100x100km^2 tiles in UTM/WGS84 projection.

Source of the data

The European Commission has deployed cloud-based platforms to provide centralised access to the Sentinel-2 data, as well as processing tools, i.e. the DIAS, or Data and Information Access Services. The current pre-processing and download-workflow have been deployed on the SOBLOO DIAS, but can easily be transferred to other cloud-based platforms in case issues with the SOBLOO DIAS occur.

Challenges

The operational use of Sentinel-2 at a continental scale comes with a few challenges. One is the large amount of data. A total of 1776 tiles need to be processed to cover the Level 2 area of WaPOR. With each tile approximately 800 MB large, and with approximetaly 2 images are available per dekad, a total of 1.42 TB of input data is generated per dekad. These data sizes will result in longer processing time, with a possible later delivery time compared to the Proba-V workflow.

In addition, many changes are still ongoing in the data availability and data delivery, both by the Sentinel hubs and the DIAS platforms. This is currently complicating the development of operational and NRT workflows for the Sentinel data. For example, at the time of writing this document, no official release terms were available, stating how soon acquired images would be available at either Level-1C or Level-2A (atmospherically corrected or not). This could severely impact the NRT data availability of Level 2 data in WaPOR.

Alternative sources

The Sentinel-2 satellites are the best currently available at the sub-100m resolution. No alternatives are currently foreseen. As Sentinel-2 is itself an alternative for the Proba-V sensor, we refer to the Proba-V alternatives for more information.

3.1.4 Landsat

The Landsat satellites are the primary source for 30m optical data used at Level 3. The Landsat program started in 1972 with the Landsat 1 satellite. Currently the 8th Landsat satellite (launched in 2013) is orbiting the Earth, carrying the Operational Land Imager (OLI) sensor. Together with Landsat 7 (launched in 2009, ETM+), it forms the input of recent Level 3 data. To cover the whole project period, data from Landsat 5 TM (1984-2011) is also used.

Purpose of data

Landsat data is used for the production of the following data components:

- Surface albedo at Level 3
- NDVI composites for Level 3
- Land surface temperature at Level 3[2]
- Land Cover and Crop Classification at Level 3

Acquired data

All Landsat optical and thermal bands except the panchromatic band are used.

Source of the data

The data can be acquired from the National Satellite Land Remote Sensing Data Archive at the USGS EROS Centre (http://landsat.usgs.gov). Thanks to the Landsat Global Archive Consolidation[3] (LGAC) effort (since 2010),. this centre holds the most geographically and temporally rich collection of Landsat data.

Challenges

Several problems exist with the Landsat data archive. Firstly, not all data is accessible, this varies by location of the Level 3 areas. Secondly, Landsat 7 developed a problem with its scan-line corrector (SLC), leading to

[2] For Landsat 5 and 7, the method described in Jiménez-Muñoz et al. (2009) is used and for Landsat 8, the method described in Rozenstein, et al. (2014) is used.

[3] https://www.usgs.gov/land-resources/nli/landsat/landsat-global-archive-consolidation?qt-science_support_page_related_con=3#qt-science_support_page_related_con

reduced data quality in the form of data gaps, affecting Landsat 7 data from June 2003 onwards. Thirdly, a malfunction in the Landsat 5 TM sensor prompted reactivation of the MSS sensor, which made only limited acquisitions, affecting the period Nov 2011 – Jan 2013.

Alternative sensors

Alternative sources of freely available, moderate resolution multi-spectral (in VNIR) satellite sensors exist that could be used as alternatives if significant data gaps in the Landsat archives occur for a Level 3 ROI. These are as follows:

- Sentinel-2A and 2B data are available from 2015 and 2017 respectively. Its VNIR bands are at 10m resolution and compare well with those of Landsat 8. The Sentinel-2 constellation can serve as back up for recent data acquisitions. Unfortunately Sentinel-2 satellites do not have thermal bands.
- ASTER has VNIR data at 15 m resolution and thermal infrared bands at 90m resolution. It has a temporal extent from 2000 to present but data has only been acquired for areas that have been requested in the past. Spectrally, ASTER's Green, Red and NIR bands compare well with those of Landsat 7 ETM.
- CBERS (China Brazil Resource Satellite) has VNIR and thermal sensors (CCD, PANMUX, IRMSS) which are spectrally similar to Landsat. , CBERS-2B and CBERS 4 form potential alternative data sources within the period of interest. Limited data is freely available from INPE for Africa.
- The ALI sensor on board EO-1 was decommissioned on 30 March 2017, and could therefore be a possible additional source of historic data. The sensor mimics the Landsat ETM spatial and spectral resolution. Note that data does not exist in the archives for all the Level 3 areas.
- Archived SPOT data older than 5 years has been made freely available for research purposes by CNES. This data can also be used to fill in significant data gaps in historic datasets.

3.2 Model input data

3.2.1 MSG

Atmospheric transmissivity is a measure of the fraction of solar radiation that passes through the atmosphere and reaches the Earth's surface. Cloud cover information is used to quantify the transmissivity of the atmosphere for shortwave solar radiation. The Meteosat Second Generation (MSG) geostationary satellite measures cloud cover in time steps of 15 minutes. Cleaned-up and geo-corrected cloud cover data is provided as part of the Cloud Physical Properties (CPP) algorithm development by the Koninklijk Nederlands Meteorologisch Instituut (KNMI) in the Netherlands. The down-welling surface fluxes provided by the algorithm are reliable inputs for calculating daily transmissivity.

Purpose of data

The MSG shortwave radiation product is used to calculate daily transmissivity which, in turn, is used to produce the solar radiation data component for all levels.

Acquired data

KNMI produces several products within the Cloud Physical Properties algorithm based on the MSG Spinning Enhanced Visible and InfraRed Imager (SEVIRI) sensor. One of these products is a shortwave radiation product in the form of 15-minute surface downwelling solar radiation ("sds"), in total 48 files per day. This product has a resolution of 4 km.

Acquisition of the data

Data can be obtained from EUMETSAT's Climate Monitoring Satellite Application Facility (CM-SAF).

Challenges
The coarse resolution of the data affects the estimation of solar radiation at higher resolution levels. At the moment no higher resolution data is available for the project area.

Alternative sources
The GEOS-5 data assimilation system can deliver the same information but at a coarser resolution.

3.2.2 GEOS-5
The Goddard Earth Observing System Model, Version 5 (GEOS-5) uses the Earth System Modeling Framework (ESMF). The GEOS-5 Data Assimilation System (GEOS-5 DAS) integrates the GEOS-5 Atmospheric Global Climate Model (GEOS-5 AGCM) with the Gridpoint Statistical Interpolation (GSI) atmospheric analysis developed jointly with NOAA/NCEP/EMC.

Purpose of data
Air temperature, relative humidity and wind speed are derived from GEOS-5. These data are used to produce the actual and reference ET, as well as relative soil moisture content at all 3 levels.

Acquired data
Temperature at 2m height, specific humidity and wind speed in the east-west and north-south direction.

Acquisition of the data
The GEOS-5 systems are developed by the Global Modeling and Assimilation Office (GMAO) to support NASA's earth science research. The data are available for download from the NASA ftp-site: ftp://gmao_ops@ftp.nccs.nasa.gov/fp/das/. Data is freely available with a time-step of 6 hours. The data is also available as OPeNDAP data on the http://opendap.nccs.nasa.gov/ server.

Challenges
The coarse resolution of the data makes it impossible to obtain location-specific meteo data. In mountainous regions corrections are made for elevation using a digital elevation model (see section 4.3.1).

Alternative sources
These data can be obtained from MERRA for the period spanning 1979 to February 2016.

3.2.3 CHIRPS
The Climate Hazards Group InfraRed Precipitation with Station data (CHIRPS) is a 30+ year quasi-global rainfall dataset that covers all longitudes between the latitudes 50°S-50°N. Data is available from 1981 to near-present at a 0.05° resolution. The data combines satellite imagery with in-situ station data to create gridded rainfall time series for trend analysis and seasonal drought monitoring.

Purpose of data
Rainfall estimates are delivered to WaPOR, and is also used as input for estimating Interception at Levels 1, 2 and 3.

Alternative sources
In case of the highly unlikely event that the CHIRPS dataset will no longer be available, several other rainfall estimate datasets are available, including:

- RFE data, freely provided by USGS. The main characteristics include:

  - Temporal: daily and dekadal frequency (S10), products available since June 1995, new data delivered after 1 or 2 days (in near-real time).
  - Spatial: The images cover Africa and the Near East and are expressed in WGS84 - Albers Conical Equal-Area projection at 8 km resolution.

- TAMSAT:
  - Spatial: Africa in Geographical Lon/Lat at 0.0375° resolution (ca. 4km).
  - Temporal: Coverage: 1983 – present; Frequencies: daily, dekadal, monthly, seasonal
  - Note that TAMSAT data does not cover the Near East.

3.3 Static input data

3.3.1 Elevation, slope and aspect (topography)

Topography is an important land surface characteristic. Elevation, slope and aspect have a large impact on weather conditions and incoming solar radiation. The Shuttle Radar Topography Mission (SRTM) created a digital elevation model (DEM) on a near-global scale at a resolution of 90m. This DEM is used to generate information on surface characteristics required for the algorithms.

Purpose of data

Elevation is an input to calculate the air temperature and solar radiation parameters. The latter also uses slope and aspect to calculate local levels of solar radiation.

Source

The SRTM digital elevation model dataset is available from the USGS. It can be downloaded at http://dds.cr.usgs.gov/srtm/.

3.3.2 Land Cover

Another important source of information on land surface characteristics is land cover. Land cover information is required because not all information on vegetation can be derived from NDVI alone. Land cover is one of the data components available on WaPOR for Levels 1, 2 and 3. The legends distinguish different levels of detail, with seasonal agricultural land mapped at Level 2 and main crop types mapped for 2 seasons per year at Level 3 (see the WaPOR Methodology Documents for detail on the Land cover map legends). The L1 and L2 land cover maps are derived from the Copernicus Global Land Service land cover map (100m) for 2015.

Purpose of data

Static Land Cover information is used to calculate the Actual ET. Maps showing seasonal crop areas and/or specific main crop types per season are used as input to calculate E, T and I.

Source

Land cover maps are produced at all 3 levels for WaPOR.

Alternative sources

Several alternative data sources exist. For example ESA CCI Land Cover (20m, for 2016) and GlobCover (300m, for 2009).

4 Pre-processing steps

Pre-processing is the phase in the production process where the input data undergoes one of two different actions:

- The input data is pre-processed to the correct format to be used directly in the processing chain, for example gridding of weather data or resampling of a lower resolution raster input data to a higher resolution, or
- The input data is used in several pre-processing steps to produce intermediate data components that will be used to calculate the final evapotranspiration data components. An example is the production of NDVI and surface albedo described in Sections 4.1 and 4.2

Figure 1 shown in section 2 shows the input data components needed for the production of evapotranspiration (E, T, and I). Since Precipitation and Land Cover are data components that are available on WaPOR, the methods for their production are described separately in the WaPOR methodology documents.

The processing steps needed for the production of the different intermediate data components that are needed as input data during the processing chain for evapotranspiration are discussed in separate subsections below.

4.1 NDVI

The required input data for the derivation of the NDVI is the daily composites of TOC-reflectances in Red and NIR for all three levels, regardless of which sensor/platform is used. For Sentinel-2, a separate pre-processing step is included, to convert the original 10m data, projected in UTM, to the Proba-V 100m grid in Latitude/longitude projection. Reprojection and resampling are combined in one step, and performed directly on the required product (i.e. NDVI, albedo).

For Level 1 and 2, daily reflectance products are converted to dekadal NDVI images, using a constrained Max-NDVI compositing rule. The viewing angle is limited to be smaller than 35 degrees. This S10 (dekadal image) comprises the "best" observation extracted from the available S1 (daily) scenes. "Constrained" means that the flagged observations (e.g. clouds, snow) are not included in the selection.

The dekadal series of NDVI are still perturbed by undetected noise and missing values. Dekadal time series are smoothed based on Swets et al. (1999)[4], where first the unreliable observations (mostly local minima) are detected and then all missing or unreliable values are replaced by means of interpolation. The resulting images are completely filled with valid data. In the NDVI quality layer, an indication of the length of the filled gap is denoted, showing the reliability of the filled value.

NDVI at Level 3 is based on Landsat data. The Landsat data is first pre-processed in a number of steps shown in Figure 3.

[4]Swets, D.L, Reed, B.C., Rowland, J.D., Marko, S.E., 1999. A weighted least-squares approach to temporal NDVI smoothing. In: Proceedings of the 1999 ASPRS Annual Conference, Portland, Oregon, pp. 526-536.

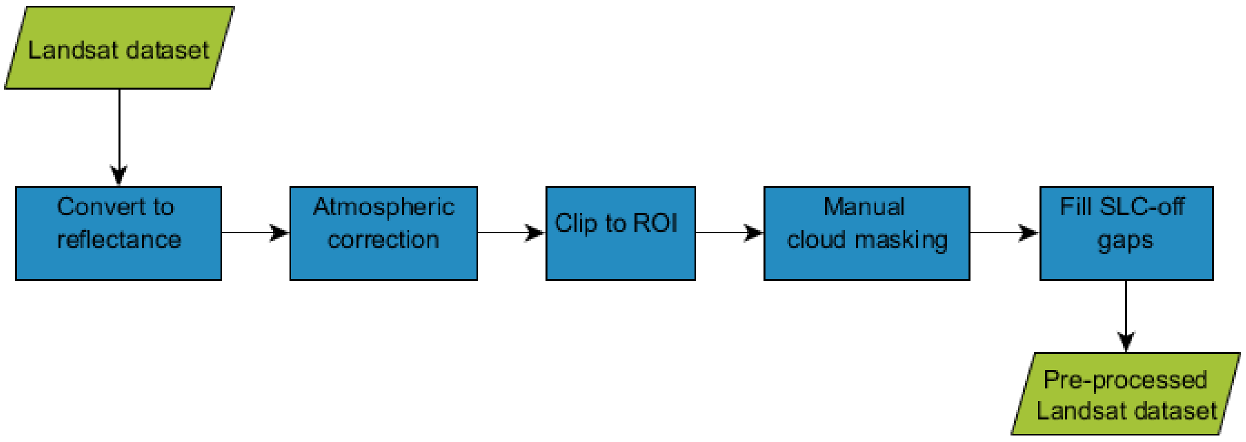


Figure 3: Flow chart showing the pre-processing steps that are applied to Landsat data at Level 3 for producing NDVI and Surface Albedo. Green parallelograms represent datasets and blue rectangles represent processing steps.

The pre-processing steps are as follows:

- Acquire Landsat data from the USGS as level 1 standard terrain corrected product.
- First, Landsat 5 and 7 data that are obtained in Digital Numbers are converted to radiance using the methodology prescribed by USGS[5]. This step is necessary in order to apply the atmospheric correction for Landsat 5 and 7, but is not necessary for Landsat 8.
- In order to remove the effects of varying atmospheric conditions across all Landsat images in the time series, atmospheric correction is done using SMAC (Simplified Model for Atmospheric Correction) after H. Rahman & G. Dedieu 1994[6].
- The data is then clipped to the extent of the Level 3 ROI plus a buffer area. This ensures that the manual cloud masking step does not take unnecessarily long.
- Since there are no automatic cloud masking procedures available for Landsat data that consistently produce accurate results, cloud masking is carried out manually using the QGIS interface.
- All gaps in Landsat 7 data that are affected by the scan line corrector (SLC) error are filled using the method proposed by Chen et al (2011)[7].
- Once the Landsat input data is pre-processed, the NDVI ratio is calculated using the Red and NIR bands of the Landsat data. The missing data in the NDVI time series (resulting from cloud masking) is then modelled using the gap-filling approach developed by Weiss, Daniel J., et al. (2014)[8]. This approach uses both spatial and temporal information within the Landsat-based NDVI time-series to fill the missing pixels. The noise in the filled NDVI time series is later smoothed out by applying the Savitzky–Golay filter method developed by Chen, Jin, et al. (2004)[10]. As a final step, dekadal NDVI

[5] https://landsat.usgs.gov/landsat-7-data-users-handbook-section-5

[6] Rahman, H. and G. Dedieu, "SMAC: a simplified method for the atmospheric correction of satellite measurements in the solar spectrum", International Journal of Remote Sensing 15(1):123-143 (1994). doi.org/10.1080/01431169408954055.

[7] Chen, J., et al., "A simple and effective method for filling gaps in Landsat ETM+ SLC-off images" Remote Sensing of Environment 115:1053-1064 (2011).

[8] Weiss, D.J., Atkinson, P.M., Bhatt, S., Mappin, B., Hay, S.I. and Gething, P.W., "An effective approach for gap-filling continental scale remotely sensed time-series" ISPRS Journal of Photogrammetry and Remote Sensing 98:106-118 (2014). doi.org/10.1016/j.isprsjprs.2014.10.001.

composites are created from the smoothed NDVI inputs using a simple average method: a dekadal NDVI is calculated as the mean of the NDVI data falling within this dekad.

4.2 Surface Albedo

For the calculation of the albedo a specific weight w_i is assigned to each available spectral band *i*. These weights compensate for the uneven distribution of the incoming solar radiation over the spectrum. The final albedo is thus computed as $r_0 = \Sigma w_i . r_i$ (summation over the *i* bands), with *ri* and *wi* the spectral reflectance and weight of the *i*-th band. NB: $\Sigma wi = 1$. For the level 1 bands, please refer to

Table 5 with the MODIS band specification. Similar to NDVI, albedo dekadal series are still perturbed by noise due to missing values and data errors, therefore the data are smoothed based on Swets et al (1999)[9] to remove noise and missing values.

At Level 3 surface albedo is based on Landsat data. The Landsat data is first pre-processed in a number of steps shown in Figure 3 (these are the same as for the calculation of the NDVI at Level 3). Surface Albedo is calculated taking into account solar exoatmospheric spectral irradiances (ESUN) provided by the USGS[10], using the algorithm given below. Note that the ESUN values vary per Landsat sensor (see Table 6) and all bands except the panchromatic and thermal bands are used.

- Level 1: Surface Albedo = 0.215*B1 + 0.215*B2 + 0.242*B3 + 0.18*B4 + 0.112*B6 + 0.036*B7
- Level 2 (Proba-V): Surface Albedo = 0.429*BLUE + 0.333*RED + 0.133*NIR + 0.105SWIR
- Level 2 (Sentinel-2): Surface Albedo = (blue_refl * blue_ESUN_value + green_refl * green_ ESUN_value + red_refl * red_ ESUN_value + nir_refl*nir_ ESUN_value) / sum(all_ ESUN_value)
- Level 3: Surface Albedo = (blue_refl * blue_ESUN_value + green_refl * green_ ESUN_value + red_refl * red_ ESUN_value + nir_refl*nir_ ESUN_value …) / sum(all_ ESUN_value)

Table 6: ESUN values (in W/m²/µm) for Landsat

Bands	Landsat 7 ETM+	Landsat 5	Landsat 8	Sentinel-2
Blue (Band 1 for LS5,7; Band 2 for LS 8)	1970	1958	1991	1941.63
Green (Band 2 for LS5,7; Band 3 for LS 8)	1842	1827	1812	1822.61
Red (Band 3 for LS5,7; Band 4 for LS 8)	1547	1551	1549	1512.79
NIR (Band 4 for LS5,7; Band 5 for LS 8)	1044	1033	972.6	1036.39
SWIR-1 (Band 5 for LS5,7; Band 6 for LS 8)	225.7	214.7	245.0	
SWIR-2 (Band 7 for LS5,7; Band 7 for LS 8)	82.06	80.7	79.72	

The last step involves applying a Savitzky-Golay filter (Chen et al, 2004)[11] to smooth surface albedo inputs and fill gaps that exist due to cloud cover.

4.3 Weather data

Weather data, i.e. temperature, wind speed and specific humidity, are available on an hourly basis. Weather data used as input to the evapotranspiration calculations are aggregated to an average daily value and instantaneous weather data used as input for soil moisture is derived by linear interpolation of the hourly input data. Daily minimum and maximum values are extracted from the time series where necessary. The coarse spatial resolution of the weather data is resampled to the level resolution and extent. The temperature data are corrected for elevation using a Digital Elevation Model (DEM). The difference in

[9] Swets, D.L, Reed, B.C., Rowland, J.D., Marko, S.E., 1999. A weighted least-squares approach to temporal NDVI smoothing. In: Proceedings of the 1999 ASPRS Annual Conference, Portland, Oregon, pp. 526-536.
[10] https://landsat.usgs.gov/esun

[11] Chen, J., et al. "A simple method for reconstructing a high-quality NDVI time-series data set based on the Savitzky-Golay filter" Remote Sensing of Environment, 91, 332-344 (2004).

elevation between a smoothed coarse scale DEM (>10 km) and a high resolution DEM (250 m) is used to upscale the coarse scale air temperature to a higher resolution. A default lapse rate of -6 °C km^{-1} is used to account for elevation differences. Wind speed and specific humidity are resampled to the level resolution by means of bilinear interpolation.

The yearly amplitude of temperature can be calculated using any time series of temperature data.

4.4 Solar radiation

Solar radiation is derived using MSG and the DEM. The MSG shortwave radiation product (see 3.2.1) is used to calculate daily transmissivity. The reason for not using the MSG shortwave radiation data directly is because of the limited resolution (> 3km) and of the lack of surface relief effects. The slope and aspect maps derived from the DEM are used to calculate the solar radiation for inclined surfaces, based on Tasumi, et al. (2006)[12]. The transmissivity is used as a proxy to distinguish between direct and diffuse radiation. At lower transmissivity values a larger percentage of the solar radiation is considered diffuse. Diffuse solar radiation is estimated by ignoring the aspect and slope of the underlying terrain. At high transmissivity values the effects of the aspect and slope of the terrain are much more visible.

4.5 Land Surface Temperature

As discussed in section 3.1, Land Surface Temperature (LST) at Levels 1 and 2 is obtained from MODIS (MOD11A1, MYD11A1). A daily composite LST dataset is produced by combining the MOD11A1 and MYD11A1 datasets. Where a pixel has multiple valid data points from both MOD11A1 and MYD11A1, the data value with its observation angle closest to nadir is selected. Daily LST is used to calculate daily relative soil moisture content. The actual daily relative soil moisture content is combined with the soil moisture composite calculated for the previous day. Each soil moisture estimate has a concomitant weight which is based on the LST observation angle (angles close to nadir carry higher weights). Where daily relative soil moisture content is missing due to a missing LST observation, the previous day's soil moisture estimate is used and the concomitant weight is reduced by 10%. Daily relative soil moisture content is used to create dekadal soil moisture composites. The 1000m LST data is resampled to the Level 1 and Level 2 resolutions using bilinear resampling.

LST data at Level 3 is obtained from Landsat which has fewer observations than Level 1 and 2 over the timespan of the WaPOR dataset. Rather than make a composite LST dataset, instantaneous soil moisture is calculated for the dates that LST observations are available and the relative soil moisture content data is then temporally interpolated with a Savitzky–Golay filter (Chen et al, 2004)[13]. Large temporal data gaps due to cloud cover are filled with Level 1 relative soil moisture content data.

[12] Tasumi, M. Allen, R. G, and R. Trezza. 2006. DEM based solar radiation estimation model for hydrological studies. Hydrological Science and Technology 22:197-208.

[13] Chen, J., et al. "A simple method for reconstructing a high-quality NDVI time-series data set based on the Savitzky-Golay filter" Remote Sensing of Environment, 91, 332-344 (2004).

5 Description of evapotranspiration data component functions

This chapter contains the documentation for the algorithms used to implement the ETLook model (Bastiaanssen et al., 2012)[14] as it is applied for the computation of the evapotranspiration data components available in WaPOR, i.e. interception, transpiration, evaporation and reference evapotranspiration. The connection between the different functions is shown. The computation has been split up into different groups of functions, each of which calculates one or more intermediate and final results. The calculation of soil moisture is also described. The documentation starts with a description of the final outputs. From these outputs the user may drill down into the different calculations.

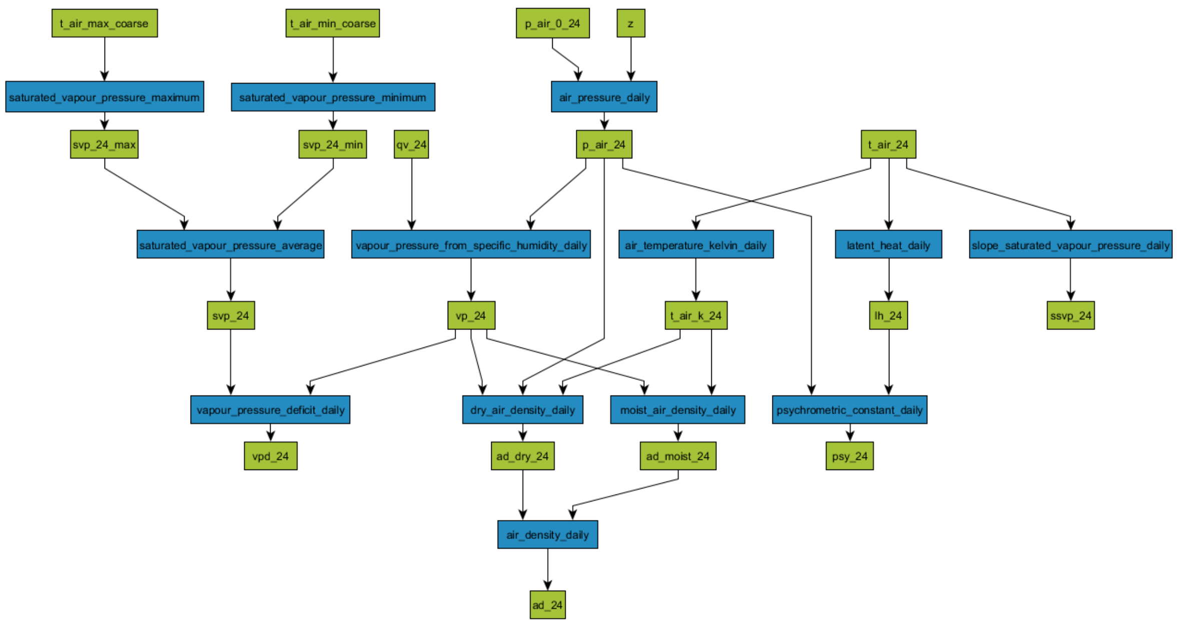
Note that all calculations (including intermediate calculations) are done on a daily basis and are aggregated to dekadal outputs for E, T and I.

5.1 Reference Evapotranspiration

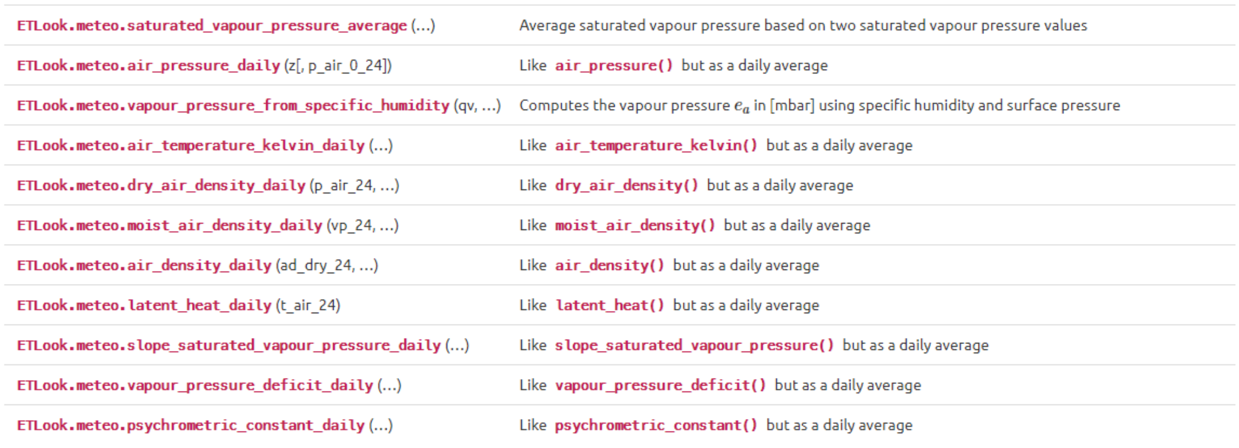
The calculation procedure for the reference evapotranspiration is presented below. Because the calculation graph for the whole algorithm is quite large, this graph is subdivided into smaller graphs. The FAO-56 procedure for the calculation of reference evapotranspiration has been split into two steps; one for obtaining the meteorological data, the other for calculating the reference evapotranspiration.

5.1.1 Meteorological data

The graph depicted in the *meteo network* below shows the calculation graph for the daily meteorological data needed for the reference evapotranspiration procedure. The meteorological data is taken from GEOS-5 and Merra.


t_air_max_coarse
t_air_min_coarse
p_air_0_24
z
saturated_vapour_pressure_maximum
saturated_vapour_pressure_minimum
air_pressure_daily
svp_24_max
svp_24_min
qv_24
p_air_24
t_air_24
saturated_vapour_pressure_average
vapour_pressure_from_specific_humidity_daily
air_temperature_kelvin_daily
latent_heat_daily
slope_saturated_vapour_pressure_daily
svp_24
vp_24
t_air_k_24
lh_24
ssvp_24
vapour_pressure_deficit_daily
dry_air_density_daily
moist_air_density_daily
psychrometric_constant_daily
vpd_24
ad_dry_24
ad_moist_24
psy_24
air_density_daily
ad_24

[14] Bastiaanssen, W.G.M., Cheema, M.J.M., Immerzeel, W. W., Miltenburg, I.J. and Pelgrum, H., (2012). Surface energy balance and actual evapotranspiration of the transboundary Indus Basin estimated from satellite measurements and the ETLook model. Water Resources Research, 48(11).



`ETLook.meteo.saturated_vapour_pressure_average` (...)	Average saturated vapour pressure based on two saturated vapour pressure values
`ETLook.meteo.air_pressure_daily` (z[, p_air_0_24])	Like `air_pressure()` but as a daily average
`ETLook.meteo.vapour_pressure_from_specific_humidity` (qv, ...)	Computes the vapour pressure e_a in [mbar] using specific humidity and surface pressure
`ETLook.meteo.air_temperature_kelvin_daily` (...)	Like `air_temperature_kelvin()` but as a daily average
`ETLook.meteo.dry_air_density_daily` (p_air_24, ...)	Like `dry_air_density()` but as a daily average
`ETLook.meteo.moist_air_density_daily` (vp_24, ...)	Like `moist_air_density()` but as a daily average
`ETLook.meteo.air_density_daily` (ad_dry_24, ...)	Like `air_density()` but as a daily average
`ETLook.meteo.latent_heat_daily` (t_air_24)	Like `latent_heat()` but as a daily average
`ETLook.meteo.slope_saturated_vapour_pressure_daily` (...)	Like `slope_saturated_vapour_pressure()` but as a daily average
`ETLook.meteo.vapour_pressure_deficit_daily` (...)	Like `vapour_pressure_deficit()` but as a daily average
`ETLook.meteo.psychrometric_constant_daily` (...)	Like `psychrometric_constant()` but as a daily average

5.2 ETlook.saturated_vapour_pressure_average

ETlook.meteo.**saturated_vapour_pressure_average**(*svp_24_max*, *p_24_min*)

Average saturated vapour pressure based on two saturated vapour pressure values calculated using minimum and maximum air temperature respectively. This is preferable to calculating saturated vapour pressure using the average air temperature, because of the strong non-linear relationship between saturated vapour pressure and air temperature

$$e_{s,24} = \frac{\left(e_{s,max} + e_{s,min}\right)}{2}$$

Parameters

svp_24_max [float] daily saturated vapour pressure based on maximum air temperature $e_{s,max}$ [mbar]

svp_24_min [float] daily saturated vapour pressure based on minimum air temperature $e_{s,min}$ [mbar]

Returns

svp_24 [float] daily saturated vapour pressure $e_{s,24}$ [mbar]

ETLook.meteo.air_pressure_daily

ETLook.meteo.**air_pressure_daily**(*z*, *p_air_0_24=1013.25*)

Like *air_pressure()* but as a daily average

Parameters

z [float] elevation z [m]

p_air_0_24 [float] daily air pressure at sea level $P_{0,24}$ [mbar]

Returns

p_air_24 [float] daily air pressure P_{24} [mbar]

ETLook.meteo.vapour_pressure_from_specific_humidity

ETLook.meteo.**vapour_pressure_from_specific_humidity**(*qv*, *p_air*)

Computes the vapour pressure e_a in [mbar] using specific humidity and surface pressure

$$e_a = \frac{q_v P}{\varepsilon}$$

where the following constant is used

- ε = ratio of molecular weight of water to dry air = 0.622 [-]

Parameters

qv [float] specific humidity q_v [kg/kg]

p_air [float] air pressure P [mbar]

Returns

vp [float] vapour pressure e_a [mbar]

ETLook.meteo.air_temperature_kelvin_daily

ETLook.meteo.**air_temperature_kelvin_daily**(t_air_24)
Like *air_temperature_kelvin()* but as a daily average

Parameters

t_air_24 [float] daily air temperature $T_{a,24}$ [C]

Returns

t_air_k_24 [float] daily air temperature $T_{a,24}$ [K]

ETLook.meteo.dry_air_density_daily

ETLook.meteo.**dry_air_density_daily**(*p_air_24*, *vp_24*, *t_air_k_24*)

Like *dry_air_density()* but as a daily average

Parameters

p_air_24 [float] daily air pressure P_{24} [mbar]

vp_24 [float] daily vapour pressure $e_{a,24}$ [mbar]

t_air_k_24 [float] daily air temperature $T_{a,24}$ [K]

Returns

ad_dry_24 [float] daily dry air density $\rho_{d,24}$ [kg m-3]

ETLook.meteo.moist_air_density_daily

ETLook.meteo.**moist_air_density_daily**(vp_24, t_air_k_24)

Like *moist_air_density()* but as a daily average

Parameters

vp_24 [float] daily vapour pressure $e_{a,24}$ [mbar]

t_air_k_24 [float] daily air temperature $T_{a,K,24}$ [K]

Returns

ad_moist_24 [float] daily moist air density $\rho_{s,24}$ [kg m-3]

ETLook.meteo.air_density_daily

ETLook.meteo.**air_density_daily**(*ad_dry_24*, *ad_moist_24*)

Like *air_density()* but as a daily average

Parameters

ad_dry_24 [float] daily dry air density $\rho_{d,24}$ [kg m-3]

ad_moist_24 [float] daily moist air density $\rho_{s,24}$ [kg m-3]

Returns

ad_24 [float] daily air density ρ_{24} [kg m-3]

ETLook.meteo.saturated_vapour_pressure_daily

ETLook.meteo.**saturated_vapour_pressure_daily**(t_air_24)

Like *saturated_vapour_pressure()* but as a daily average

Parameters

t_air_24 [float] daily air temperature $T_{a,24}$ [C]

Returns

svp_24 [float] daily saturated vapour pressure $e_{s,24}$ mbar]

ETLook.meteo.latent_heat_daily

ETLook.meteo.**latent_heat_daily**(t_air_24)

Like *latent_heat()* but as a daily average

Parameters

t_air_24 [float] daily air temperature $T_{a,24}$ [C]

Returns

lh_24 [float] daily latent heat of evaporation λ_{24} [J/kg]

ETLook.meteo.slope_saturated_vapour_pressure_daily

ETLook.meteo.**slope_saturated_vapour_pressure_daily**(t_air_24)

Like *slope_saturated_vapour_pressure()* but as a daily average

Parameters

t_air_24 [float] daily air temperature $T_{a,24}$ [C]

Returns

ssvp_24 [float] daily slope of saturated vapour pressure curve Δ_{24} [mbar K-1]

ETLook.meteo.vapour_pressure_deficit_daily

ETLook.meteo.**vapour_pressure_deficit_daily**(svp_24, vp_24)

Like *vapour_pressure_deficit()* but as a daily average

Parameters

svp_24 [float] daily saturated vapour pressure $e_{s,24}$

[mbar]

vp_24 [float] daily actual vapour pressure $e_{a,24}$ [mbar]

Returns

vpd_24 [float] daily vapour pressure deficit $\Delta_{e,24}$ [mbar]

ETLook.meteo.psychrometric_constant_daily

ETLook.meteo.**psychrometric_constant_daily**(p_air_24, lh_24)

Like *psychrometric_constant()* but as a daily average

Parameters

p_air_24 [float] daily air pressure P_{24} [mbar]

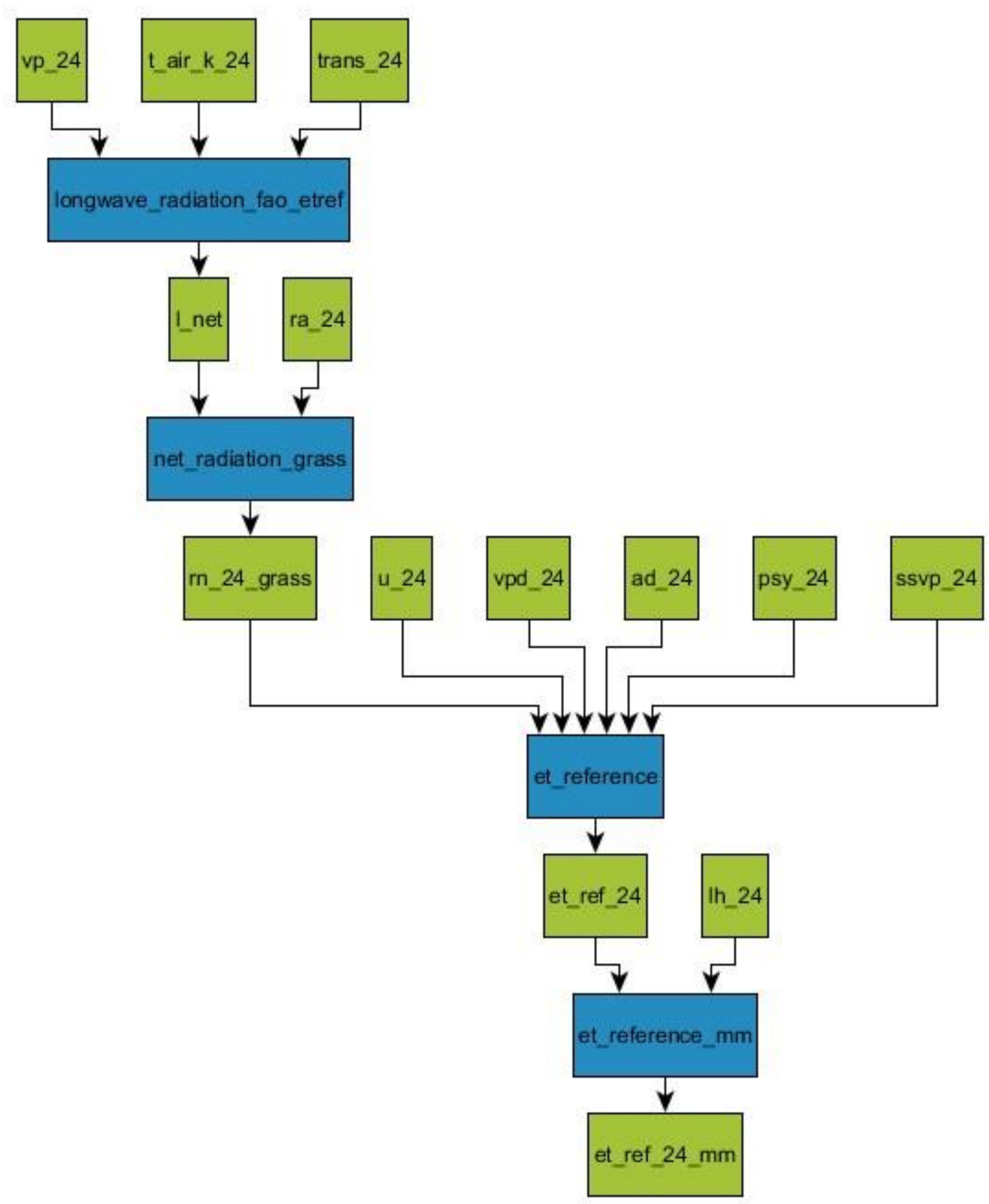
lh_24 [float] daily latent heat of evaporation λ_{24} [J/kg]

Returns

psy_24 [float] daily psychrometric constant γ_{24} [mbar K-1]

5.2.1 Reference Evapotranspiration

The graph depicted in the *reference evapotranspiration network* below shows the calculation procedure for the reference evapotranspiration according to the FAO methodology. In this methodology the evapotranspiration of a well-watered field of grass is calculated under the current atmospheric conditions. The albedo for grass is set to 0.23.



`ETLook.radiation.longwave_radiation_fao_etref` (…)	Computes the net longwave radiation according to the FAO 56 manual.
`ETLook.radiation.net_radiation_grass` (ra_24, …)	Computes the net radiation for reference grass
`ETLook.evapotranspiration.et_reference` (…)	Computes the reference evapotranspiration.
`ETLook.evapotranspiration.et_reference_mm` (…)	Computes the reference evapotranspiration.

ETLook.radiation.longwave_radiation_fao_etref

ETLook.radiation.**longwave_radiation_fao_etref**(t_air_k_24, vp_24, trans_24)

Computes the net longwave radiation according to the FAO 56 manual. For the reference ET calculation the values for vp_slope, vp_offset, lw_slope and lw_offset are being provided as defaults

$$L^* = \sigma \left(T_{a,K}\right)^4 \left(vp_off - vp_slp\sqrt{0.1e_a}\right)\left(lw_slp\frac{\tau}{0.75} + lw_off\right)$$

where the following constant is used

- σ = Stefan Boltzmann constant = 5.67 e-8 J s-1 m-2 K-4

Parameters

t_air_k_24 : float

daily air temperature in Kelvin $T_{a,K}$ [-]

vp_24 : float

daily vapour pressure e_a [mbar]

trans_24 : float

daily atmospheric transmissivity τ [-]

Returns

l_net : float

daily net longwave radiation L^* [Wm-2]

Examples

```
>>> import ETLook.radiation as rad
>>> rad.longwave_radiation_fao_etref(t_air_k=302.5, vp=10.3, trans_24=0.6)
68.594182173686306
```

ETLook.radiation.net_radiation_grass

ETLook.radiation.**net_radiation_grass**(*ra_24*, *l_net*, *r0_grass=0.23*)

Computes the net radiation for reference grass

$$Q^* = \left[(1 - \alpha_{0,grass})S^{\downarrow} - L^* - I\right]$$

Parameters ra_24 : float

daily solar radiation $S^{\downarrow}$ [Wm-2]

l_net : float

daily net longwave radiation L^* [wm-2]

r0_grass : float

albedo for reference grass $\alpha_{0,grass}$ [-]

Returns rn_24_grass : float

daily net radiation for reference grass Q^* [Wm-2]

Examples

```
>>> import ETLook.radiation as rad
>>> rad.net_radiation_grass(ra_24=123., l_net=24.)
70.7
```

ETLook.evapotranspiration.et_reference

ETLook.evapotranspiration.**et_reference**(*rn_24_grass*, *ad_24*, *psy_24*, *vpd_24*, *ssvp_24*, *u_24*)

Computes the reference evapotranspiration. The reference evapotranspiration ET_{ref} is an important concept in irrigation science. The reference evapotranspiration can be inferred from routine meteorological measurements. The reference evapotranspiration is the evapotranspiration of grass under well watered conditions. First the aerodynamical resistance for grass $r_{a,grass}$ [sm $^{-1}$] is calculated

$$r_{a,grass} = \frac{208}{u_{obs}}$$

Then the reference evapotranspiration ET_{ref} [W m $^{-2}$] can be calculated as follows, with taking the default value for the grass surface resistance r_{grass} = 70 sm $^{-1}$

$$ET_{ref} = \frac{\Delta\left(Q^*_{grass}\right) + \rho c_p \frac{\Delta_e}{r_{a,grass}}}{\Delta + \gamma\left(1 + \frac{r_{grass}}{r_{a,grass}}\right)}$$

The soil heat flux is assumed to be zero or close to zero on a daily basis.

Parameters

rn_24_grass [float] net radiation for reference grass surface Q^*_{grass} [Wm-2]

u_24 [float] daily wind speed at observation height u_{obs} [m/s]

ad_24 [float] daily air density ρ_{24} [kg m-3]

psy_24 [float] daily psychrometric constant γ_{24} [mbar K-1]

vpd_24 [float] daily vapour pressure deficit $\Delta_{e,24}$ [mbar]

ssvp_24 [float] daily slope of saturated vapour pressure curve Δ_{24} [mbar K-1]

Returns

et_ref_24 [float] reference evapotranspiration (well watered grass) energy equivalent ET_{ref} [W m-2]

ETLook.evapotranspiration.et_reference_mm

ETLook.evapotranspiration.**et_reference_mm**(et_ref_24, lh_24)

Computes the reference evapotranspiration.

$$ET_{ref} = ET_{ref} d_{sec} \lambda_{24}$$

where the following constants are used

- d_{sec} seconds in the day = 86400 [s]

Parameters

et_ref_24 [float] daily reference evapotranspiration energy equivalent ET_{ref}[W m-2]

lh_24 [float] daily latent heat of evaporation λ_{24} [J/kg]

Returns

et_ref_24_mm [float] reference evapotranspiration (well watered grass) ET_{ref}[mm d-1]

5.3 Soil Moisture

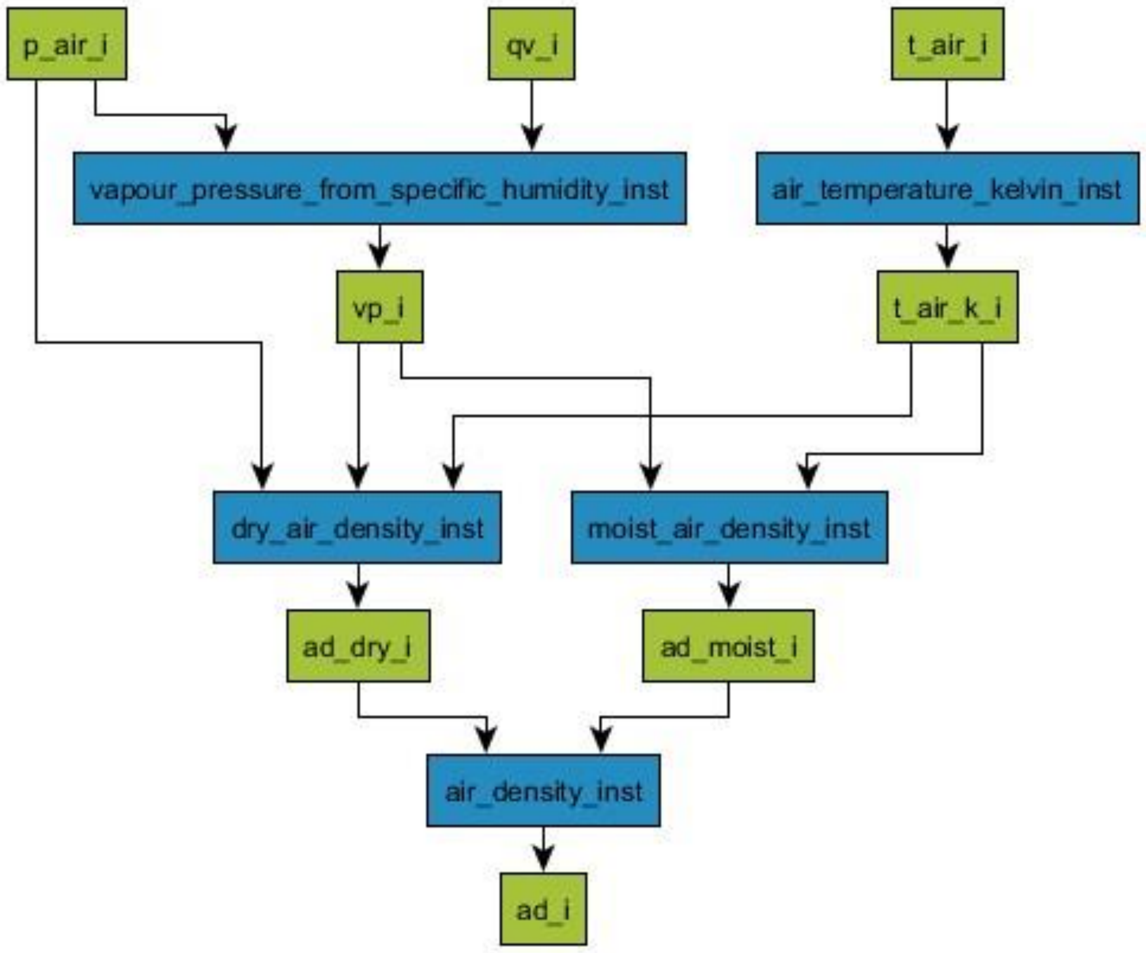
An overview of the soil moisture algorithm. The soil moisture algorithm follows the procedure outlined by Yang et al. (2015)[15]. It calculates the four cornerpoints of the vegetation cover - land surface temperature (LST) trapezoid. The actual vegetation cover and LST value are then used to estimate the relative soil moisture content.

Because the calculation graph for the whole algorithm is quite large, this graph is subdivided into smaller graphs.

5.3.1 Air Density

The first graph depicted in the *air density network* below shows the calculation graph for the air density calculation. The air density is used in other graphs below.

[15] Yang, Y., H. Guan, D. Long, B. Liu, G. Qin, J, Qin, and O. Batelaan, Estimation of Surface SoilMoisture from Thermal Infrared Remote Sensing Using an Improved Trapezoid Method, Remote Sens. 2015, 7, 8250-8270; doi:10.3390/rs70708250



`ETLook.meteo.air_temperature_kelvin_inst` (t_air_i)	Like `air_temperature_kelvin()` but as an instantaneous value
`ETLook.meteo.vapour_pressure_from_specific_humidity_inst` (…)	Like `vapour_pressure_from_specific_humidity()` but as an instantaneous value
`ETLook.meteo.moist_air_density_inst` (vp_i, …)	Like `moist_air_density()` but as an instantaneous value
`ETLook.meteo.dry_air_density_inst` (p_air_i, …)	Like `dry_air_density()` but as an instantaneous value
`ETLook.meteo.air_density_inst` (ad_dry_i, …)	Like `air_density()` but as a instantaneous value

ETLook.meteo.air_temperature_kelvin_inst

ETLook.meteo.**air_temperature_kelvin_inst**(t_air_i)

Like *air_temperature_kelvin()* but as an instantaneous value

Parameters

t_air_i [float] instantaneous air temperature $T_{a,i}$ [C]

Returns

t_air_k_i [float] instantaneous air temperature $T_{a,i}$ [K]

ETLook.meteo.vapour_pressure_from_specific_humidity_inst

ETLook.meteo.**vapour_pressure_from_specific_humidity_inst**(qv_i, p_air_i)

Like *vapour_pressure_from_specific_humidity()* but as an instantaneous value

Parameters

qv_i [float] instantaneous specific humidity $q_{v,i}$ [kg/kg]

p_air_i [float] instantaneous air pressure P_i [mbar]

Returns

vp_i [float] instantaneous vapour pressure $e_{a,i}$

[mbar]

ETLook.meteo.moist_air_density_inst

ETLook.meteo.**moist_air_density_inst**(vp_i, t_air_k_i)

Like *moist_air_density()* but as an instantaneous value

Parameters

vp_i [float] instantaneous vapour pressure $e_{a,i}$ [mbar]

t_air_k_i [float] instantaneous air temperature $T_{a,K,i}$ [K]

Returns

ad_moist_i [float] instantaneous moist air density $\rho_{s,i}$ [kg m-3]

ETLook.meteo.dry_air_density_inst

ETLook.meteo.**dry_air_density_inst**(p_air_i, vp_i, t_air_k_i)

Like *dry_air_density()* but as an instantaneous value

Parameters

p_air_i [float] instantaneous air pressure P_i [mbar]

vp_i [float] instantaneous vapour pressure $e_{a,i}$ [mbar]

t_air_k_i [float] instantaneous air temperature $T_{a,i}$ [K]

Returns

ad_dry_i [float] instantaneous dry air density $\rho_{d,i}$ [kg m-3]

ETLook.meteo.air_density_inst

ETLook.meteo.**air_density_inst**(*ad_dry_i*, *ad_moist_i*)

Like *air_density()* but as a instantaneous value

Parameters

ad_dry_i [float] instantaneous dry air density $\rho_{d,i}$ [kg m-3]

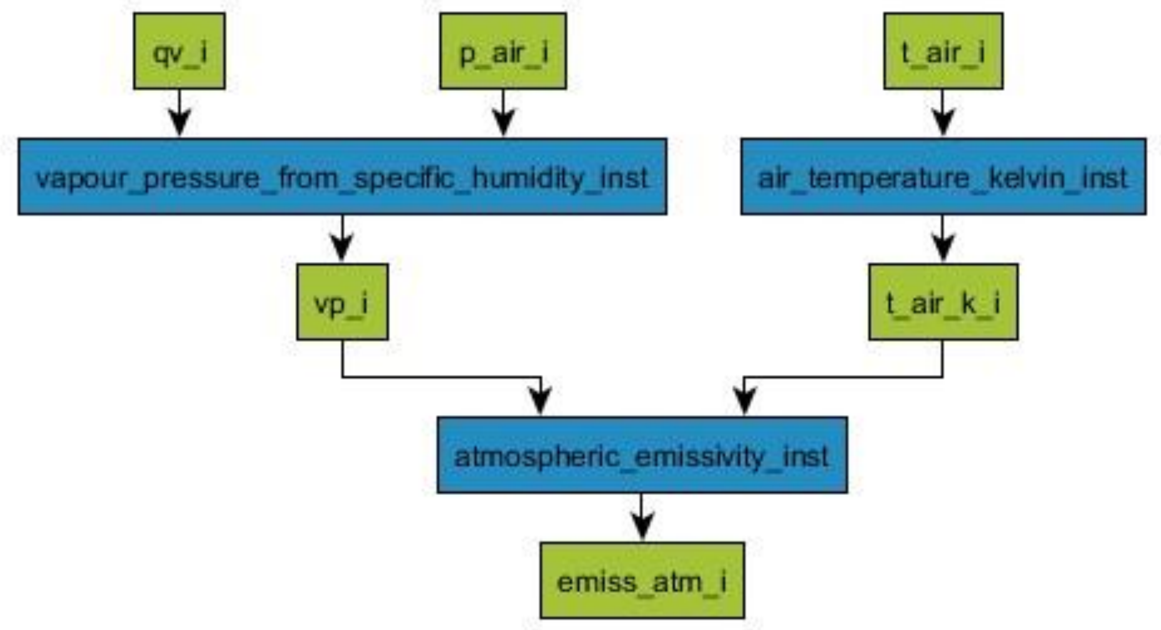
ad_moist_i [float] instantaneous moist air density $\rho_{s,i}$ [kg m-3]

Returns

ad_i [float] instantaneous air density ρ_i [kg m-3]

5.3.2 Atmospheric emissivity

The second graph depicted in the *atmospheric network* below shows the calculation graph for the atmospheric emissivity calculation. The atmospheric emissivity is used in other graphs below.


qv_i
p_air_i
t_air_i
vapour_pressure_from_specific_humidity_inst
air_temperature_kelvin_inst
vp_i
t_air_k_i
atmospheric_emissivity_inst
emiss_atm_i

ETLook.meteo.vapour_pressure_from_specific_humidity_inst (...)	Like vapour_pressure_from_specific_humidity() but as an instantaneous value
ETLook.meteo.air_temperature_kelvin_inst (t_air_i)	Like air_temperature_kelvin() but as an instantaneous value
ETLook.soil_moisture.atmospheric_emissivity_inst (...)	Computes the atmospheric emissivity according to Brutsaert [R1cef826be7fa-1].

ETLook.soil_moisture.atmospheric_emissivity_inst

ETLook.soil_moisture.**atmospheric_emissivity_inst**(*vp_i*, *t_air_k_i*) Computes the atmospheric emissivity according to Brutsaert[16].

$$\varepsilon_a = a\left(\frac{e_a}{T_a}\right)^b$$

where the following constants are used

- a = 1.24
- b = 1/7

Parameters

vp_i [float] instantaneous vapour pressure e_a [mbar]

t_air_k_i [float] instantaneous air temperature T_a [K]

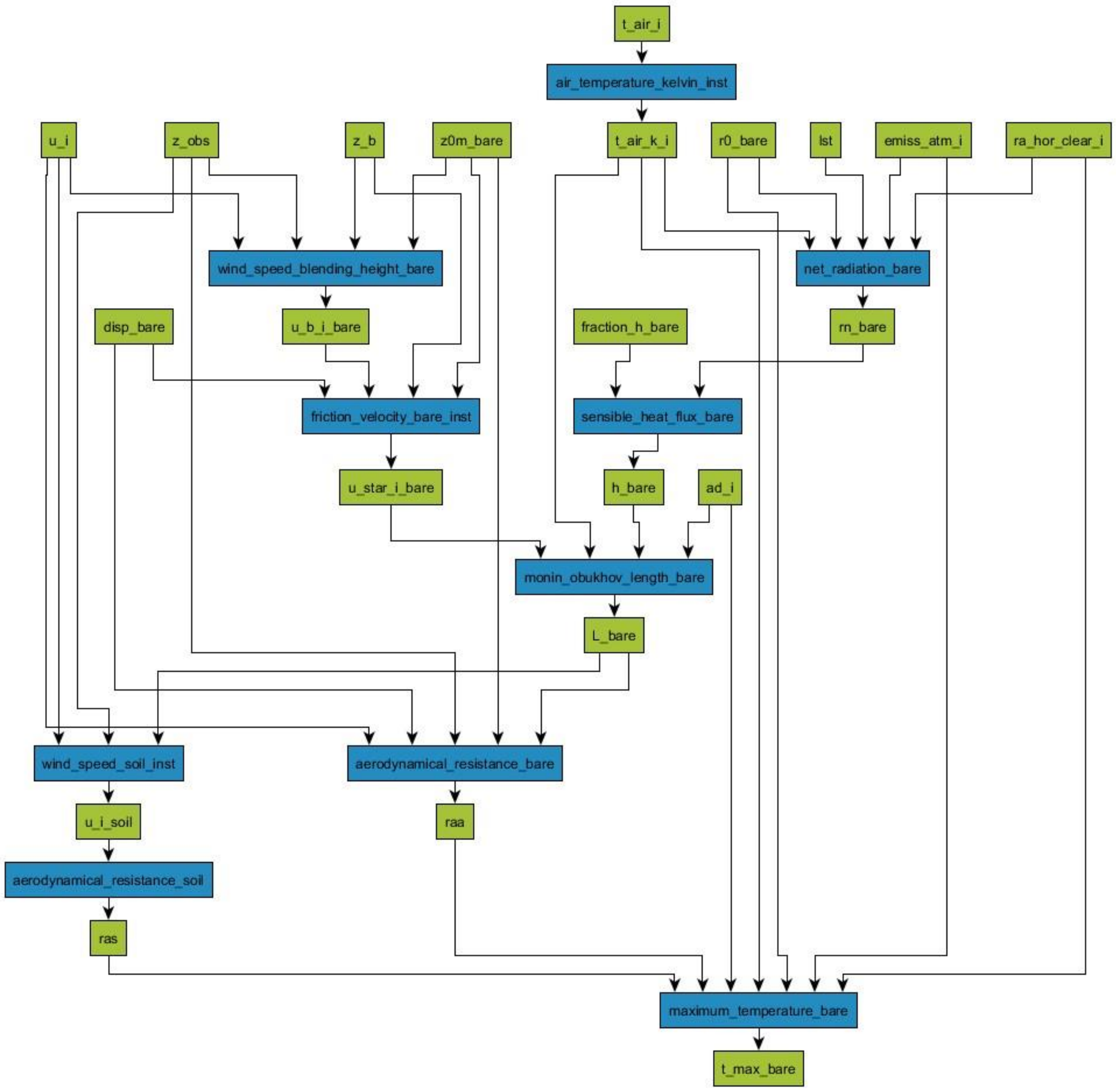
Returns

[16] Brutsaert, W., On a derivable formula for long-wave radiation from clear skies, Water Resour. Res, 1975, 11, 742-744.

emiss_atm_i [float] instantaneous atmospheric emissivity ε_a [-
]

5.3.3 Bare soil maximum temperature

The third graph depicted in the *bare soil maximum temperature network* below shows the calculation graph for the calculation of the theoretical bare soil maximum temperature. This defines one of the cornerpoints of the temperature - vegetation cover trapezoid.



ETLook.meteo.air_temperature_kelvin_inst (t_air_i)	Like air_temperature_kelvin() but as an instantaneous value
ETLook.soil_moisture.wind_speed_blending_height_bare (u_i)	Computes the wind speed at blending height u_b [m/s] using the logarithmic wind profile
ETLook.soil_moisture.net_radiation_bare (...)	Computes the net radiation for the bare soil with zero evaporation
ETLook.soil_moisture.friction_velocity_bare_inst (...)	Like initial_friction_velocity_inst() but with bare soil parameters
ETLook.soil_moisture.sensible_heat_flux_bare (rn_bare)	Computes the bare soil sensible heat flux
ETLook.soil_moisture.monin_obukhov_length_bare (...)	Like unstable.monin_obukhov_length() but with bare soil parameters
ETLook.soil_moisture.wind_speed_soil_inst (...)	Computes the instantaneous wind speed at soil surface
ETLook.soil_moisture.aerodynamical_resistance_soil (...)	Computes the aerodynamical resistance of the soil
ETLook.soil_moisture.maximum_temperature_bare (...)	Computes the maximum temperature under dry bare soil conditions
ETLook.soil_moisture.aerodynamical_resistance_bare (...)	Computes the aerodynamical resistance for a dry bare soil.

ETLook.soil_moisture.wind_speed_blending_height_bare

ETLook.soil_moisture.**wind_speed_blending_height_bare**(*u_i*, *z0m_bare=0.001*, *z_obs=10*, *z_b=100*)

Computes the wind speed at blending height u_b [m/s] using the logarithmic wind profile

$$u_b = \frac{u_{obs}\ln\left(\frac{z_b}{z_{0,m}}\right)}{\ln\left(\frac{z_{obs}}{z_{0,m}}\right)}$$

Parameters

u_i [float] instantaneous wind speed at observation height u_{obs}

[m/s]

z_obs [float] observation height of wind speed z_{obs} [m]

z_b [float] blending height z_b [m]

z0m_bare [float] surface roughness bare soil $z_{0,m}$ m

Returns

u_b_i_bare [float] instantaneous wind speed at blending height for bare soil $u_{b,i,bare}$

[m/s]

ETLook.soil_moisture.net_radiation_bare

ETLook.soil_moisture.**net_radiation_bare**(*ra_hor_clear_i*, *emiss_atm_i*, *t_air_k_i*, *lst*, *r0_bare=0.38*)

Computes the net radiation for the bare soil with zero evaporation

$$Q^*_{bare} = (1-\alpha_{0,bare})\,S_d + \varepsilon_s\varepsilon_a\sigma T_a^4 - \varepsilon_s\sigma T_s^4$$

Parameters

ra_hor_clear_i [float] Total clear-sky irradiance on a horizontal surface S_d

[W/m2]

emiss_atm_i [float] instantaneous atmospheric emissivity ε_a [-]

t_air_k_i [float] instantaneous air temperature T_a [K]

lst [float] surface temperature T_0 [K] r0_bare [float] dry bare soil surface albedo

$\alpha_{0,bare}$ [-]

Returns

rn_bare [float] net radiation bare soil Q^*_{bare} [Wm-2]

ETLook.soil_moisture.friction_velocity_bare_inst

ETLook.soil_moisture.**friction_velocity_bare_inst**(*u_b_i_bare*, *z0m_bare=0.001*, *disp_bare=0.0*, *z_b=100*)

Like *initial_friction_velocity_inst()* but with bare soil parameters

Parameters

u_b_i_bare [float] instantaneous wind speed blending height bare soil $u_{b,d}$ [W m-2]

z0m_bare [float] surface roughness bare soil $z_{0,m,b}$ [m]

disp_bare [float] displacement height bare soil d^b [m]

z_b [float] blending height z_b [m]

Returns

u_star_i_bare [float] instantaneous friction velocity bare soil u^*_b [m s-1]

ETLook.soil_moisture.sensible_heat_flux_bare

ETLook.soil_moisture.**sensible_heat_flux_bare**(*rn_bare*, *fraction_h_bare=0.65*)

Computes the bare soil sensible heat flux

$$H_{bare} = H_{f,bare} Q_{*bare}$$

Parameters

rn_bare [float] net radiation bare soil Q^*_{bare} [Wm-2]

fraction_h_bare [float] fraction of H of net radiation bare soil $H_{f,bare}$ [-]

Returns

h_bare [float] sensible heat flux bare soil H_{bare} [Wm-2]

ETLook.soil_moisture.monin_obukhov_length_bare

ETLook.soil_moisture.**monin_obukhov_length_bare**(h_bare, ad_i, u_star_i_bare, *t_air_k_i*)

Like unstable.monin_obukhov_length() but with bare soil parameters

Parameters

h_bare [float] sensible heat flux for dry bare soil $H_{b,d}$ [W m-2]

ad_i [float] instantaneous air density ρ [k g m-3]

u_star_i_bare [float] instantaneous friction velocity bare soil

u_b^* [m s-1]

t_air_k_i [float] instantaneous air temperature T_a [K]

Returns

L_bare [float] monin obukhov length dry vegetation $L_{b,d}$ [m]

ETLook.soil_moisture.wind_speed_soil_inst

ETLook.soil_moisture.**wind_speed_soil_inst**(u_i, L_bare, z_obs=10)

Computes the instantaneous wind speed at soil surface

$$u_{i,s} = u_{obs} \frac{\ln\left(\frac{z_{obs}}{z_0}\right)}{\ln\left(\frac{z_{obs}}{z_{0,s}}\right) - \psi_m\left(\frac{-z_0}{L}\right)}$$

Parameters

u_i [float] wind speed at observation height u_{obs} [m/s]

z_obs [float] observation height of wind speed z_{obs} [m]

L_bare [float] monin obukhov length L [m]

Returns

u_i_soil [float] instantaneous wind speed just above soil surface $u_{i,s}$ [ms-1]

ETLook.soil_moisture.aerodynamical_resistance_soil

ETLook.soil_moisture.**aerodynamical_resistance_soil**(u_i_soil)

Computes the aerodynamical resistance of the soil

$$r_{a,s} = \frac{1}{\left(0.0025T_{dif}^{\frac{1}{3}} + 0.012u_{i,s}\right)}$$

Parameters

u_i_soil [float] instantaneous wind speed just above soil surface $u_{i,s}$ [m s-1]

Returns

ras [float] aerodynamical resistance $r_{a,s}$ [sm-1]

ETLook.soil_moisture.maximum_temperature_bare

ETLook.soil_moisture.**maximum_temperature_bare**(*ra_hor_clear_i*, *emiss_atm_i*, *t_air_k_i*, *ad_i*, *raa*, *ras*, *r0_bare=0.38*) Computes the maximum temperature under dry bare soil conditions

$$T_{s,max} = \frac{(1-\alpha_s)\,S_d + \varepsilon_s\varepsilon_a\sigma T_a^4 - \varepsilon_s\sigma T_a^4}{4\varepsilon_s\sigma T_a^3 + \rho C_p/\left[(r_{a,a}+r_{a,s})\,(1-G/R_{n,s})\right]} + T_a$$

Parameters

ra_hor_clear_i [float] Total clear-sky irradiance on a horizontal surface

$ra_h or_c lear_i$ [W/m2]

emiss_atm_i [float] instantaneous atmospheric emissivity P [-]

t_air_k_i [float] instantaneous air temperature T_a [K]

ad_i [float] instantaneous air density ρ [kg m-3]

raa [float] aerodynamical resistance $r_{a,a}$ [sm-1]

ras [float] aerodynamical resistance $r_{a,s}$ [sm-1]

r0_bare [float] dry bare soil surface albedo $\alpha_{0,bare}$ [-]

Returns

t_max_bare [float] maximum temperature at bare soil Tc,max [K]

ETLook.soil_moisture.aerodynamical_resistance_bare

ETLook.soil_moisture.**aerodynamical_resistance_bare**(*u_i*, *L_bare*, *z0m_bare=0.001*, *disp_bare=0.0*, *z_obs=10*) Computes the aerodynamical resistance for a dry bare soil.

$$z_1 = \frac{z_{obs}-d}{z_{0,b,m}}$$

$$z_2 = \frac{z_{obs}-d}{L_b}$$

$$r_{a,a} = \frac{(\ln(z_1)-\phi_m(-z_2))(\ln(z_1)-\phi_h(-z_2))}{k^2 u}$$

Parameters

u_i [float] instantaneous wind speed at observation height u_{obs}

[m/s]

z_obs [float] observation height of wind speed z_{obs} [m]

disp_bare [float] displacement height d [m]

z0m_bare [float] surface roughness $z_{0,b,m}$ [m]

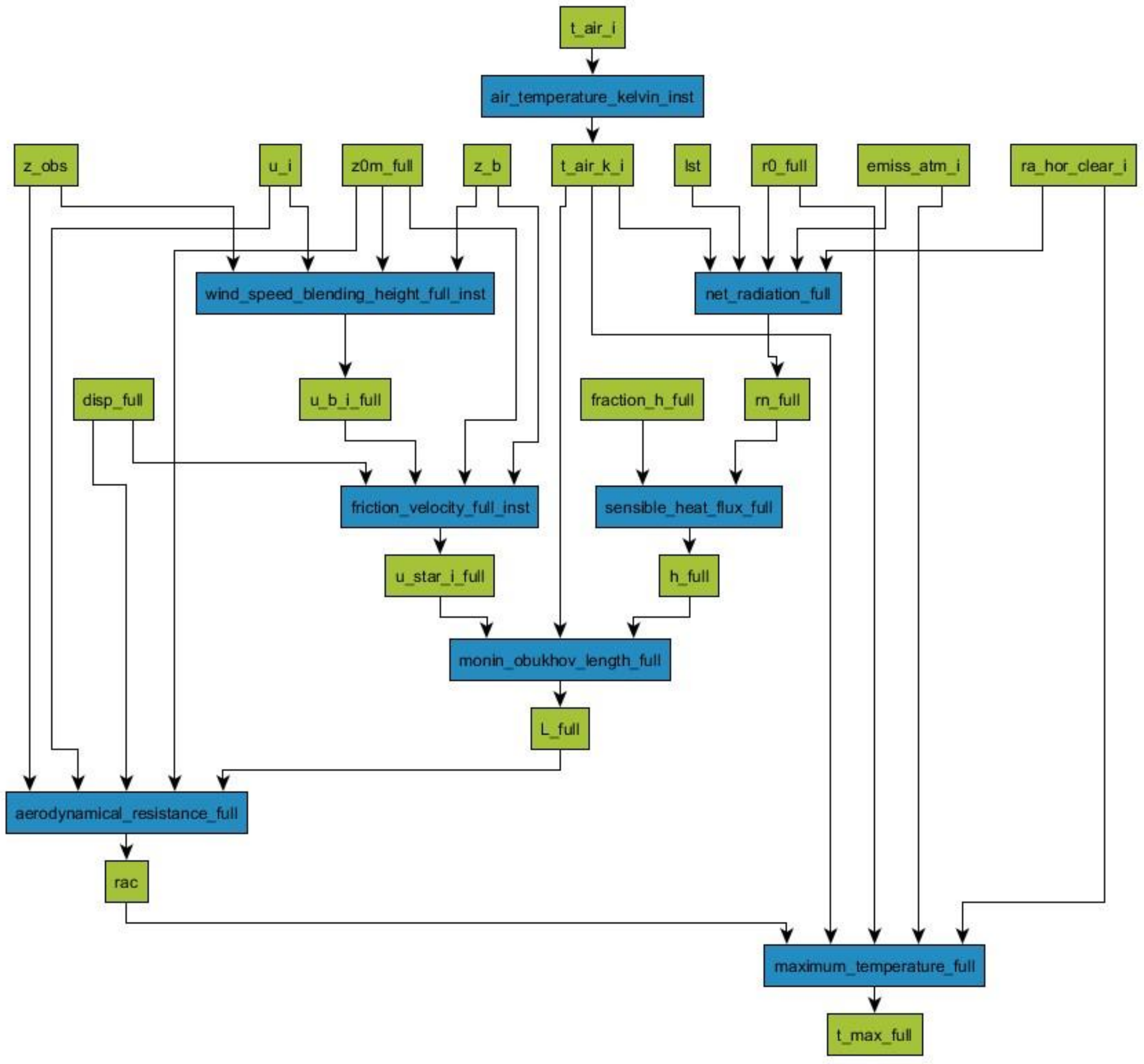
L_bare [float] monin obukhov length L_b [m]

Returns

raa [float] aerodynamical resistance dry surface $r_{a,a}$ [sm-1]

5.3.4 Full canopy maximum temperature

The fourth graph depicted in the *full canopy maximum temperature network* below shows the calculation graph for the calculation of the theoretical full canopy maximum temperature. This defines one of the cornerpoints of the temperature - vegetation cover trapezoid.



`ETLook.meteo.air_temperature_kelvin_inst` (t_air_i)	Like `air_temperature_kelvin()` but as an instantaneous value
`ETLook.soil_moisture.wind_speed_blending_height_full_inst` (u_i)	Computes the wind speed at blending height u_b [m/s] using the logarithmic wind profile
`ETLook.soil_moisture.net_radiation_full` (…)	Computes the net radiation at full canopy with zero evaporation
`ETLook.soil_moisture.sensible_heat_flux_full` (rn_full)	Computes the full canopy sensible heat flux
`ETLook.soil_moisture.friction_velocity_full_inst` (…)	Like `initial_friction_velocity_inst()` but with full vegetation parameters
`ETLook.soil_moisture.monin_obukhov_length_full` (…)	Like `unstable.monin_obukhov_length()` but with full canopy parameters
`ETLook.soil_moisture.aerodynamical_resistance_full` (…)	Computes the aerodynamical resistance for a full canopy.
`ETLook.soil_moisture.maximum_temperature_full` (…)	Computes the maximum temperature under fully vegetated conditions

ETLook.soil_moisture.wind_speed_blending_height_full_inst

ETLook.soil_moisture.**wind_speed_blending_height_full_inst**(u_i, z0m_full=0.1, *z_obs=10*, *z_b=100*)

Computes the wind speed at blending height u_b [m/s] using the logarithmic wind profile

$$u_b = \frac{u_{obs} \ln\left(\frac{z_b}{z_{0,m}}\right)}{\ln\left(\frac{z_{obs}}{z_{0,m}}\right)}$$

Parameters

u_i [float] instantaneous wind speed at observation height u_{obs}

[m/s]

z_obs [float] observation height of wind speed z_{obs} [m]

z_b [float] blending height z_b [m]

z0m_full [float] surface roughness vegetation $z_{0,m}$ [m]

Returns

u_b_i_full [float] instantaneous wind speed at blending height for full vegetation $u_{b,i,full}$ [m s-1]

ETLook.soil_moisture.net_radiation_full

ETLook.soil_moisture.**net_radiation_full**(*ra_hor_clear_i*, *emiss_atm_i*, *t_air_k_i*, *lst*, *r0_full=0.18*)

Computes the net radiation at full canopy with zero evaporation

$$Q^*_{full} = \left(1 - \alpha_{0,full}\right) S_d + \varepsilon_c \varepsilon_a \sigma T_a^4 - \varepsilon_c \sigma T_s^4$$

Parameters

ra_hor_clear_i [float] Total clear-sky irradiance on a horizontal surface

$ra_h or_c lear_i$ [W/m2]

emiss_atm_i [float] instantaneous atmospheric emissivity P [-]

t_air_k_i [float] instantaneous air temperature T_a [K]

lst [float] surface temperature T_0 [K]

r0_full [float] surface albedo full vegetation $\alpha_{0,full}$ [-]

Returns

rn_full [float] net radiation full vegetation Q^*_{full} [Wm-2]

ETLook.soil_moisture.sensible_heat_flux_full

ETLook.soil_moisture.**sensible_heat_flux_full**(*rn_full*, *fraction_h_full=0.95*)

Computes the full canopy sensible heat flux

$$H_{full} = H_{f,full} Q^*_{full}$$

Parameters

rn_full [float] net radiation full vegetation Q^*_{full} [Wm-2]

fraction_h_full [float] fraction of H of net radiation full vegetation $H_{f,full}$ [-]

Returns

h_full [float] sensible heat flux full vegetation Hfull [Wm-2]

ETLook.soil_moisture.friction_velocity_full_inst

ETLook.soil_moisture.**friction_velocity_full_inst**(*u_b_i_full*, *z0m_full=0.1*, *disp_full=0.667*, *z_b=100*)

Like *initial_friction_velocity_inst()* but with full vegetation parameters

Parameters

u_b_i_full [float] instantaneous wind speed blending height for full vegetation $u_{b,d}$ [m s-1]

z0m_full [float] surface roughness vegetation $z_{0,m,b}$ [m]

disp_full [float] displacement height vegetation d^b [m]

z_b [float] blending height z_b [m]

Returns

u_star_i_full [float] instantaneous friction velocity vegetation u^*_f [m s-1]

ETLook.soil_moisture.monin_obukhov_length_full

ETLook.soil_moisture.**monin_obukhov_length_full**(h_full, ad_i, u_star_i_full, *t_air_k_i*)

Like unstable.monin_obukhov_length() but with full canopy parameters

Parameters

h_full [float] sensible heat flux for dry full vegetation $H_{f,d}$ [W m-2]

ad_i [float] instantaneous air density ρ [k g m-3]

u_star_i_full [float] instantaneous friction velocity vegetation u^*_b [m s-1]

t_air_k_i [float] instantaneous air temperature T_a [K]

Returns

L_full [float] monin obukhov length dry vegetation $L_{f,d}$ [m]

ETLook.soil_moisture.aerodynamical_resistance_full

ETLook.soil_moisture.**aerodynamical_resistance_full**(*u_i*, *L_full*, *z0m_full=0.1*, *disp_full=0.667*, *z_obs=10*) Computes the aerodynamical resistance for a full canopy.

$$z_1 = \frac{z_{obs} - d}{z_{0,m}}$$
$$z_2 = \frac{z_{obs} - d}{L}$$
$$z_3 = \frac{z_{0,m}}{L}$$
$$z_4 = \frac{z_{obs} - d}{\frac{z_{0,m}}{7}}$$
$$z_5 = \frac{\frac{z_{0,m}}{7}}{L}$$
$$r_{a,c} = \frac{(\ln(z_1) - \phi_m(-z_2) + \phi_m(-z_3))(\ln(z_4) - \phi_h(-z_2) + \phi_h(-z_5))}{k^2 u}$$

Parameters

u_i [float] instantaneous wind speed at observation height u_{obs}

[m/s]

z_obs [float] observation height of wind speed z_{obs} [m]

disp_full [float] displacement height d [m]

z0m_full [float] surface roughness $z_{0,m}$ [m]

L_full [float] monin obukhov length L [m]

Returns

rac [float] aerodynamical resistance canopy $r_{a,c}$ [sm-1]

ETLook.soil_moisture.maximum_temperature_full

ETLook.soil_moisture.**maximum_temperature_full**(*ra_hor_clear_i*, *emiss_atm_i*, *t_air_k_i*, *ad_i*, *rac*, *r0_full=0.18*)

Computes the maximum temperature under fully vegetated conditions

$$T_{c,max} = \frac{(1 - \alpha_c) S_d + \varepsilon_c \varepsilon_a \sigma T_a^4 - \varepsilon_c \sigma T_a^4}{4 \varepsilon_s \sigma T_a^3 + \rho C_p / r_{a,c}} + T_a$$

Parameters

ra_hor_clear_i [float] Total clear-sky irradiance on a horizontal surface $ra_hor_clear_i$ [W/m2]

emiss_atm_i [float] instantaneous atmospheric emissivity P [-]

t_air_k_i [float] instantaneous air temperature T_a [K]

rac [float] aerodynamic resistance canopy $r_{a,c}$ [sm-1]

ad_i [float] instantaneous air density ρ [kg m-3]

r0_full [float] surface albedo full vegetation cover $\alpha_{0,full}$ [-]

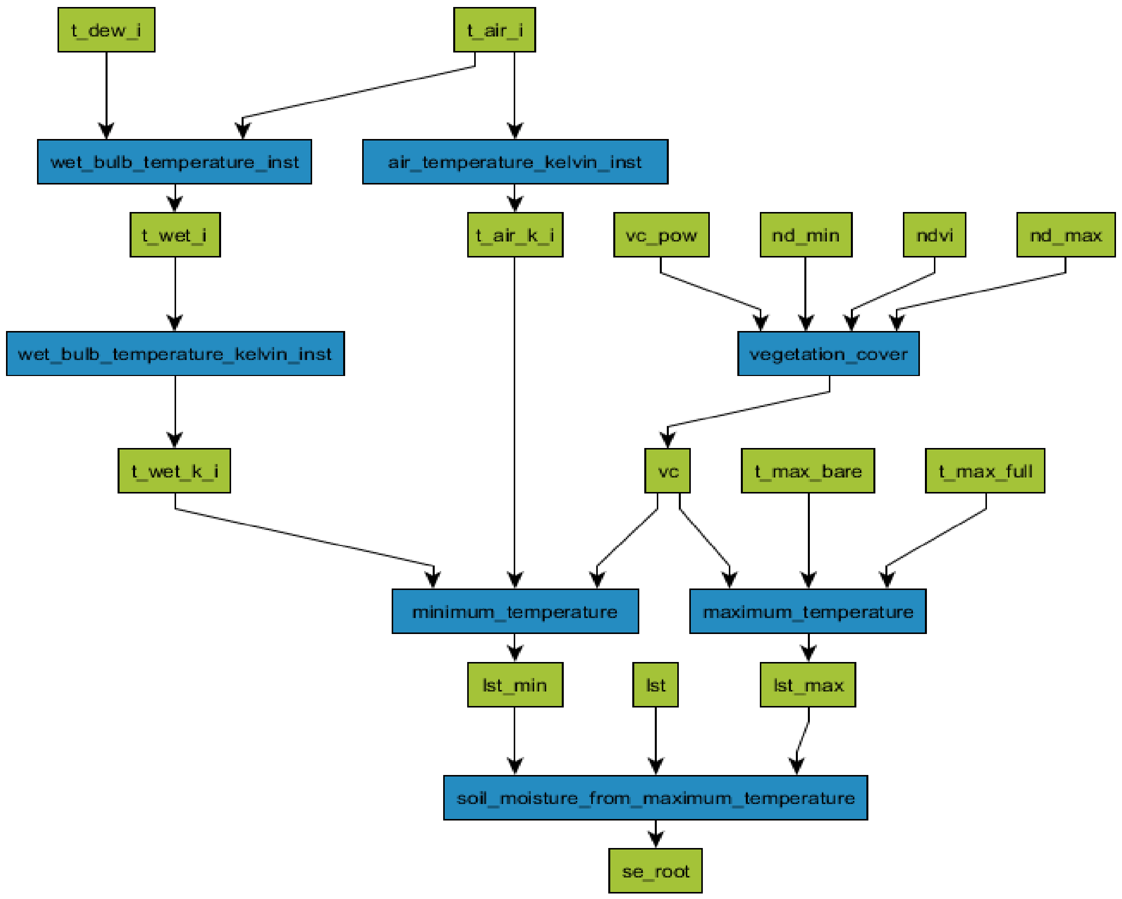
Returns

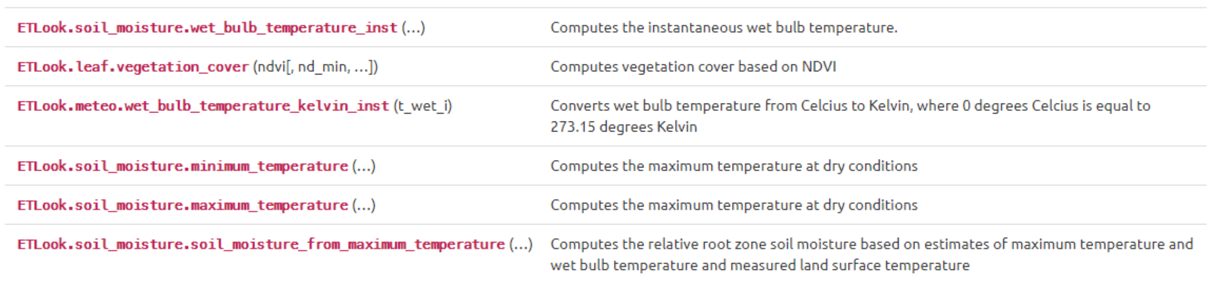
t_max_full [float] maximum temperature at full vegetation cover $T_{c,max}$

[K]

5.3.5 Soil Moisture

The fifth and final graph depicts the *soil moisture network*. This graph shows the calculation graph for the calculation of the soil moisture based on the temperature - vegetation cover trapezoid.





ETLook.soil_moisture.wet_bulb_temperature_inst (…)	Computes the instantaneous wet bulb temperature.
ETLook.leaf.vegetation_cover (ndvi[, nd_min, …])	Computes vegetation cover based on NDVI
ETLook.meteo.wet_bulb_temperature_kelvin_inst (t_wet_i)	Converts wet bulb temperature from Celcius to Kelvin, where 0 degrees Celcius is equal to 273.15 degrees Kelvin
ETLook.soil_moisture.minimum_temperature (…)	Computes the maximum temperature at dry conditions
ETLook.soil_moisture.maximum_temperature (…)	Computes the maximum temperature at dry conditions
ETLook.soil_moisture.soil_moisture_from_maximum_temperature (…)	Computes the relative root zone soil moisture based on estimates of maximum temperature and wet bulb temperature and measured land surface temperature

ETLook.soil_moisture.wet_bulb_temperature_inst

ETLook.soil_moisture.**wet_bulb_temperature_inst**(*t_air_i*, *t_dew_i*)

Computes the instantaneous wet bulb temperature.

Parameters

t_air_i [float] instantaneous air temperature T_a [C]

t_dew_i [float] instantaneous dew point temperature Td_a

[C]

Returns

t_wet_i [float] instantaneous wet bulb temperature Tw_a

[C]

ETLook.leaf.vegetation_cover

ETLook.leaf.**vegetation_cover**(*ndvi*, *nd_min=0.125*, *nd_max=0.8*, *vc_pow=0.7*)

Computes vegetation cover based on NDVI

$$c_{veg} = \begin{cases} 0 & I_{NDVI} \leq I_{NDVI,min} \\ 1 - \left(\frac{I_{NDVI,max} - I_{NDVI}}{I_{NDVI,max} - I_{NDVI,min}}\right)^a & I_{NDVI,min} < I_{NDVI} < I_{NDVI,max} \\ 1 & I_{NDVI} \geq I_{NDVI,max} \end{cases}$$

Parameters

ndvi [float] Normalized Difference Vegetation Index I_{NDVI} [-]

nd_min [float] NDVI value where vegetation cover is 0 $I_{NDVI,min}$ [-]

nd_max [float] NDVI value where vegetation cover is 1 $I_{NDVI,max}$ [-]

vc_pow [float] Exponential power used in vegetation cover function a [-]

Returns

vc [float] vegetation cover c_{veg} [-]

Examples

```
>>> from ETLook import leaf
>>> leaf.vegetation_cover(0.1, nd_min=0.2)
0
>>> leaf.vegetation_cover(0.5)
0.4331446663885373
>>> leaf.vegetation_cover(0.85)
1
```

ETLook.meteo.wet_bulb_temperature_kelvin_inst

ETLook.meteo.**wet_bulb_temperature_kelvin_inst**(t_wet_i)

Converts wet bulb temperature from Celcius to Kelvin, where 0 degrees Celcius is equal to 273.15 degrees Kelvin

Parameters

t_wet_i [float] instantaneous wet bulb temperature $T_{w,i}$ [C]

Returns

t_wet_k_i [float] instantaneous wet bulb temperature $T_{w,i}$ [K]

ETLook.soil_moisture.maximum_temperature

ETLook.soil_moisture.**maximum_temperature**(t_max_bare, t_max_full, vc)
Computes the maximum temperature at dry conditions

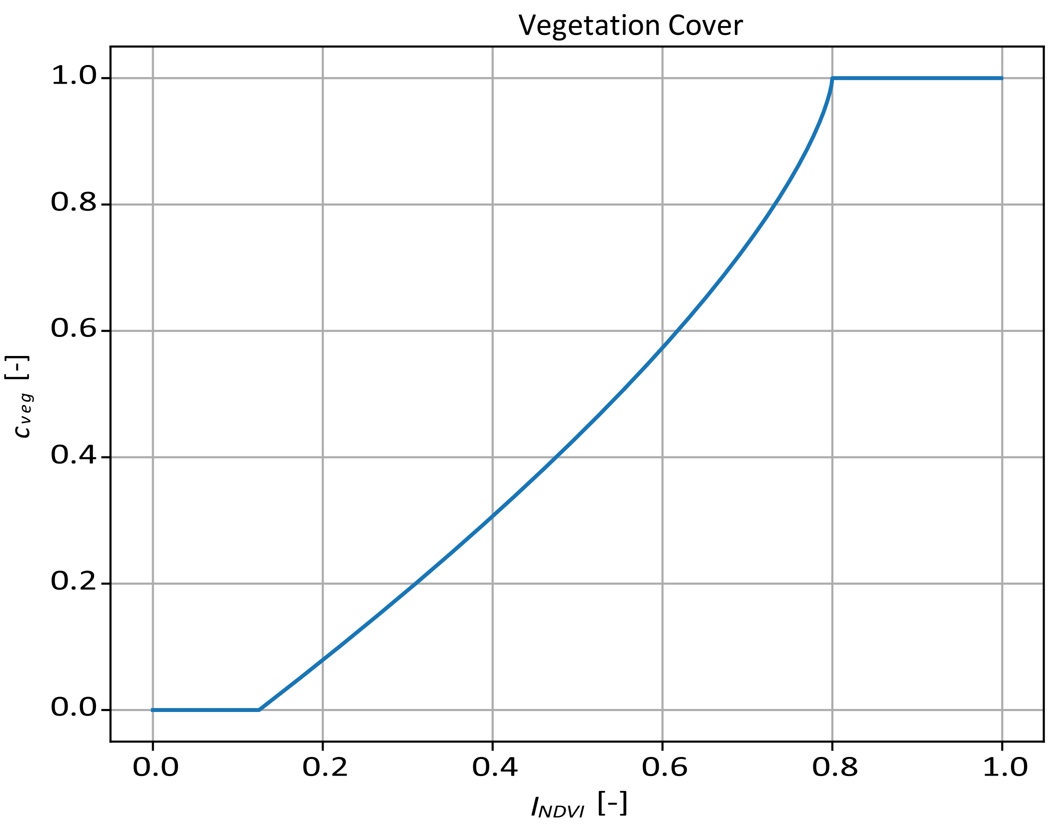
$$T_{0,max} = c_{veg}\left(T_{c,max} - T_{s,max}\right) + T_{s,max}$$

Parameters

t_max_bare [float] maximum temperature at bare soil $T_{s,max}$ [K]

t_max_full [float] maximum temperature at full dry vegetation $T_{c,max}$ [K]

vc [float] vegetation cover c_{veg} [-]



Returns

lst_max [float] maximum temperature at dry conditions $T_{0,max}$ [K]

ETLook..soil_moisture.minimum_temperature

ETLook.soil_moisture.**minimum_temperature**(*t_wet_k_i*, *t_air_k_i*, *vc*)

Computes the maximum temperature at dry conditions

$$T_{0,min} = c_{veg}(T_{a,i} - T_w) + T_w$$

Parameters

t_wet_k_i [float] minimum temperature at bare soil $T_{s,max}$ [K]

t_air_k_i [float] minimum temperature at full vegetation $T_{c,max}$ [K]

vc [float] vegetation cover c_{veg} [-]

Returns

lst_min [float] minimum temperature at wet conditions $T_{0,min}$ [K]

ETLook.soil_moisture.soil_moisture_from_maximum_temperature

ETLook.soil_moisture.**soil_moisture_from_maximum_temperature**(lst_max,lst,**Lst_min*)***

Computes the relative root zone soil moisture based on estimates of maximum temperature and wet bulb temperature and measured land surface temperature

$$\Theta = \frac{T_0 - T_{0,min}}{T_{0,max} - T_{0,min}}$$

Parameters

lst [float] land surface temperature T_0 [K]

lst_max [float] maximum temperature at dry conditions $T_{0,max}$ [K]

lst_min [float] minimum temperature at wet conditions $T_{0,min}$ [K]

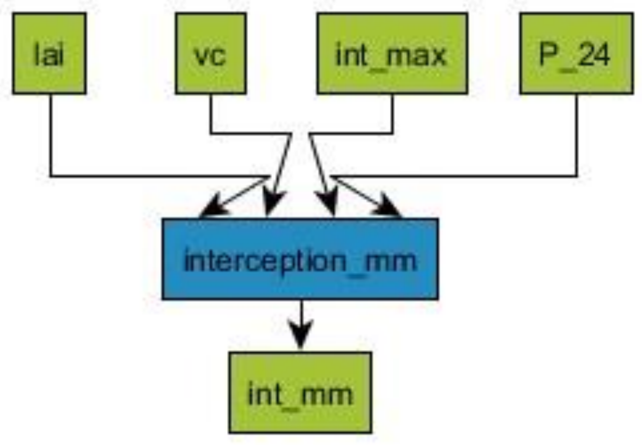
Returns

se_root [float] relative root zone soil moisture (based on LST) Θ [%]

5.4 Interception

The interception calculation procedure is presented below. The calculation graph is shown in the *figure* below. A list of functions is provided with links to their description, followed by a list of input variables.

5.4.1 Interception



ETLook.evapotranspiration.interception_mm (...) Computes the daily interception.

ETLook.evapotranspiration.interception_mm

ETLook.evapotranspiration.**interception_mm**(P_24, vc, lai, int_max=0.2)
Computes the daily interception. The daily interception of a vegetated area is calculated according to von Hoyningen-Hüne (1983)[17] and Braden(1985)[18].

$$I^* = I_{max} * I_{lai} * \left(1 - \left(\frac{1}{1 + \frac{c_{veg} P24}{I_{max} I_{lai}}}\right)\right)$$

Parameters

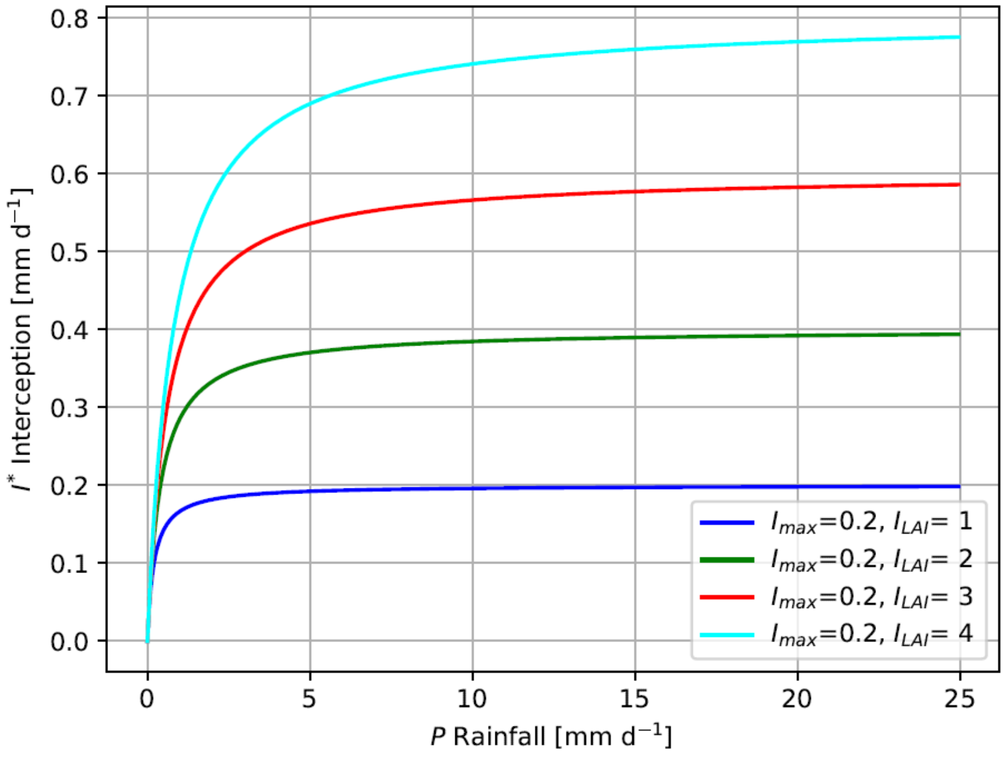
- P_24 [float] daily rainfall P [mm day $^{-1}$]
- vc [float] vegetation cover c_{veg} [-]
- lai [float] leaf area index I_{lai} [-]
- int_max [float] maximum interception per leaf I_{max} [mm day $^{-1}$]

Returns

int_mm [float] interception I^* [mm day $^{-1}$]

[17] von Hoyningen-Hüne, J., Die Interception des Niederschlags in landwirtschaftlichen Beständen. Schriftenreihe des DVWK, 1983, 57, 1-53
[18] Braden, H., Energiehaushalts- und Verdunstungsmodell für Wasser- und Stoffhaushalts-untersuchungen landwirtschaftlich genutzter Einzugsgebiete. Mitteilungen der Deutschen Bodenkundlichen Gesellschaft, (1985), 42, 254-299

Examples

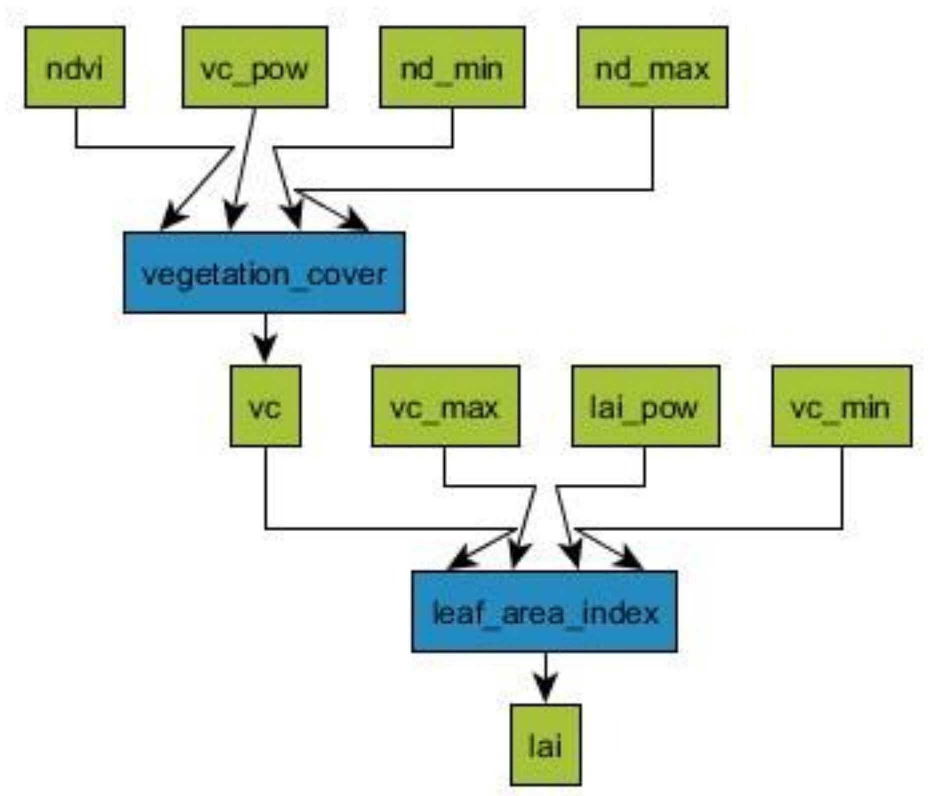


5.5 Transpiration

The transpiration calculation procedure is presented below. Because the calculation graph for the whole algorithm is quite large, this graph is subdivided into smaller graphs.

5.5.1 Leaf Area Index

The first graph depicted in the *leaf area index network* below shows the calculation graph for the leaf area index. The leaf area index is used in other graphs below.



`ETLook.leaf.leaf_area_index` (vc[, vc_min, …])	Computes leaf area index based on vegetation cover.
`ETLook.leaf.vegetation_cover` (ndvi[, nd_min, …])	Computes vegetation cover based on NDVI

ETLook.leaf.leaf_area_index

ETLook.leaf.**leaf_area_index**(*vc*, *vc_min=0.0*, *vc_max=0.9677324224821418*, *lai_pow=-0.45*)

Computes leaf area index based on vegetation cover. It is based on the Kustas formulation of LAI vs NDVI.

$$I_{lai} = \begin{cases} 0 & c_{veg} \leq c_{veg,min} \\ \frac{\ln(-(c_{veg}-1))}{b} & c_{veg,min} < c_{veg} \leq c_{veg,max} \\ \frac{\ln(-(c_{veg,max}-1))}{b} & c_{veg} > c_{veg,max} \end{cases}$$

Parameters

vc [float] vegetation cover c_{veg} [-]

vc_min [float] vegetation cover where LAI is 0 $c_{veg,min}$ [-]

vc_max [float] vegetation cover at maximum LAI $c_{veg,max}$ [-]

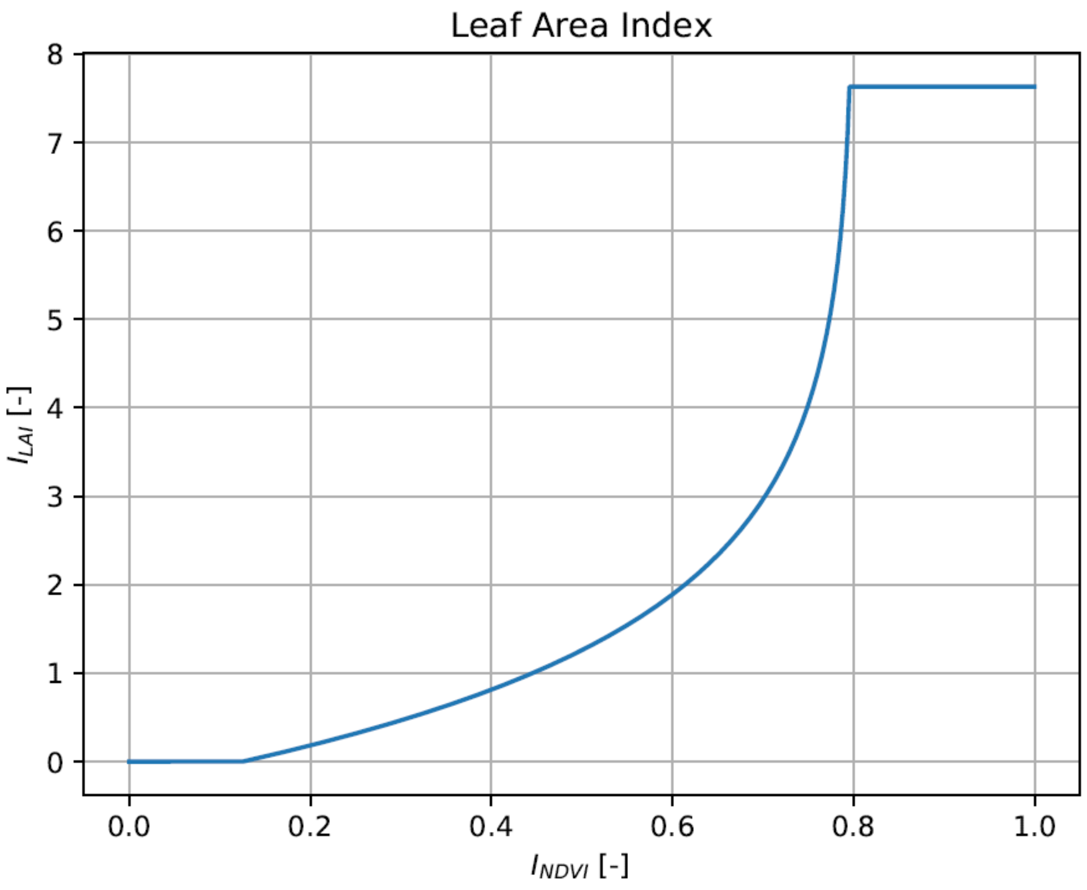
lai_pow [float] exponential factor used in LAI function b [-]

Returns

lai [float] leaf area index I_{lai} [-]

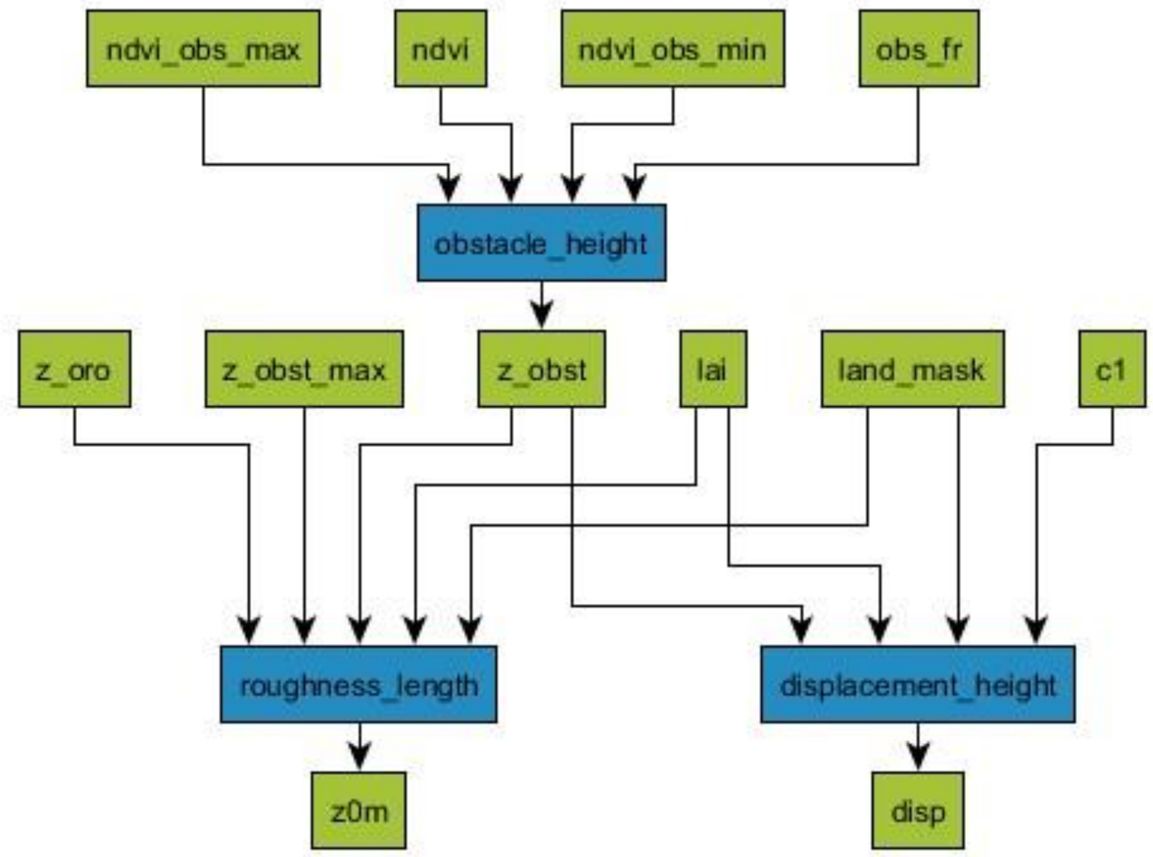
Examples

```
>>> from ETLook import leaf
>>> leaf.leaf_area_index(0.0)
0
>>> leaf.leaf_area_index(0.5)
1.5403270679109895
>>> leaf.leaf_area_index(1.0)
7.6304274331264414
```



5.5.2 Surface Roughness

The second graph depicted in the *surface roughness network* below shows the calculation graph for the surface roughness. The surface roughness is used in other graphs below.



`ETLook.roughness.displacement_height` (lai, z_obst)	Computes the displacement height.
`ETLook.roughness.obstacle_height` (ndvi, …)	Computes the obstacle height.
`ETLook.roughness.roughness_length` (lai, …)	Computes the surface roughness length.

ETLook.roughness.displacement_height

ETLook.roughness.**displacement_height**(*lai*, *z_obst*, *land_mask=1*, *c1=1*)

Computes the displacement height. The lai is used to limit the displacement height. It is defined differently for different types of landuse.

Land use is classified as follows:

0. no data
1. land
2. water
3. urban

$$z_{disp} = \begin{cases} 0 & l = 0 \\ z_{obst}\left(1 - \frac{1-\exp\left(-\sqrt{c_1 I_{lai}}\right)}{\sqrt{c_1 I_{lai}}}\right) & l = 1 \\ 0 & l = 2 \\ \frac{2}{3} z_{obst} & l = 3 \end{cases}$$

Parameters

lai [float] leaf area index I_{lai} [-] z_obst [float] obstacle height

z_{obst} [m] land_mask [int] land use classification l [-]

c1 [float] exponential growth rate displacement height

function c_1 [-]

Returns

disp [float] displacement height $disp$ [m]

Examples

```
>>> import ETLook.roughness as roughness >>> roughness.displacement_height(0.4, 2.0)
0.51779495
```

ETLook.roughness.obstacle_height

ETLook.roughness.**obstacle_height**(*ndvi*, *z_obst_max*, *ndvi_obs_min=0.25*, *ndvi_obs_max=0.75*, *obs_fr=0.25*)

Computes the obstacle height. The ndvi is used to limit the obstacle height.

$$z_{obst} = \begin{cases} f_{obs} z_{obst,max} & I_{ndvi} \leq I_{ndvi,obs,min} \\ z_{obst,max}\left(f_{obs} + (1 - f_{obs})\left(\frac{I_{ndvi} - I_{ndvi,obs,min}}{I_{ndvi,obs,max} - I_{ndvi,obs,min}}\right)\right) & I_{ndvi} > I_{ndvi,obs,min} \& I_{ndvi} < I_{ndvi,obs,max} \\ z_{obst,max} & I_{ndvi} \geq I_{ndvi,obs,max} \end{cases}$$

Parameters

ndvi [float] normalized difference vegetation index I_{ndvi} [-]

ndvi_obs_min [float] normalized difference vegetation

index @ min obstacle height $I_{ndvi,obs,min}$ [-]

ndvi_obs_max [float] normalized difference vegetation

index @ max obstacle height $I_{ndvi,obs,max}$ [-]

obs_fr [float] ratio of minimum and maximum obstacle

height f_{obs} [-]

z_obst_max [float] maximum obstacle height

:math‘z_{obst,max}‘ [m]

Returns

z_obst [float] obstacle height z_{obst} [m]

Examples

```
>>> import ETLook.roughness as roughness >>> roughness.obstacle_height(0.4, 2.0)
0.95
```

ETLook.roughness.roughness_length

ETLook.roughness.**roughness_length**(*lai*, *z_oro*, *z_obst*, *z_obst_max*, *land_mask=1*)

Computes the surface roughness length. The roughness length is related to the roughness characteristics.

For the logarithmic wind-profile the surface roughness length is the height at which the wind speed is zero.

The roughness length is calculated differently for different types of land use

Land use is classified as follows:

0. no data
1. land
2. water
3. urban

$$z_{0,m} = \begin{cases} 0 & l = 0 \\ z_{0,m} & l = 1 \\ 0.0001 & l = 2 \\ \frac{1}{7} z_{obst,max} + z_{oro} & l = 3 \end{cases}$$

Parameters

lai [float] leaf area index I_{lai} [-]

z_oro [float] orographic roughness z_{oro} [m]

z_obst [float] obstacle height z_{obst} [m]

z_obst_max [float] maximum obstacle height $z_{obst,max}$ [m]

land_mask [int] land use classification l [-]

Returns

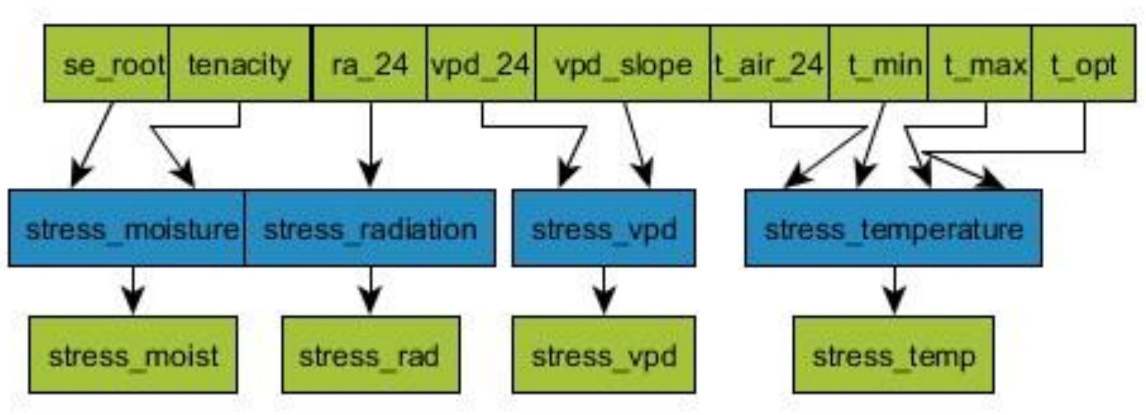
z0m [float] roughness length $z_{0,m}$ [m]

Examples

```
>>> import ETLook.roughness as roughness >>> roughness.roughness_length(0.4)
0.34179999999999999
```

5.5.3 Stress factors

The third graph depicted in the *stress factors network* below shows the calculation graph for the different stress factors. These stress factors limit the transpiration.



`ETLook.stress.stress_moisture` (se_root[, ...])	Computes the stress for plants when there is not sufficient soil moisture in the root zone
`ETLook.stress.stress_radiation` (ra_24)	Computes the stress for plants when there is not sufficient radiation
`ETLook.stress.stress_temperature` (t_air_24[, ...])	Computes the stress for plants when it is too cold or hot
`ETLook.stress.stress_vpd` (vpd_24[, vpd_slope])	Computes the stress for plants if the vpd increases too much.

ETLook.stress.stress_moisture

ETLook.stress.**stress_moisture**(*se_root*, *tenacity=1.5*)

Computes the stress for plants when there is not sufficient soil moisture in the root zone

$$S_m = K_{sf} S_{e,root} - \frac{\sin(2\pi S_{e,root})}{2\pi}$$

The tenacity factor K_{sf}ranges from 1 for sensitive plants to 1.5 for moderately sensitive plants to 3 for insensitive (tenacious plants).

Parameters

se_root [float] effective saturation root zone moisture *S_{e,root}*

[-]

tenacity [float] tenacity factor *K_{sf}* [-]

Returns

stress_moist [float] stress factor for root zone moisture S_m [-

]

Examples

```
>>> import ETLook.stress as stress >>> stress.stress_moisture(0.5)
0.75
>>> stress.stress_moisture(0.5, tenacity = 1)
0.5
>>> stress.stress_moisture(0.5, tenacity = 3)
1.0
```

ETLook.stress.stress_radiation

ETLook.stress.**stress_radiation**(ra_24)

Computes the stress for plants when there is not sufficient radiation

$$S_r = \frac{S^{\downarrow}}{(S^{\downarrow} + 60.)}\left(1 + \frac{60}{500}\right)$$

Parameters

ra_24 [float] daily solar radiation $S^{\downarrow}$ [Wm-2]

Returns

stress_rad [float] stress factor for radiation S_r [-]

Examples

```
>>> import ETLook.stress as stress >>> stress.stress_radiation()
0.0
>>> stress.stress_radiation(500)
1.0
>>> stress.stress_radiation(700)
1.0
>>> stress.stress_radiation(250)
0.90322580645161288
```

ETLook.stress.stress_temperature

ETLook.stress.**stress_temperature**(*t_air_24*, *t_opt=25.0*, *t_min=0.0*, *t_max=50.0*)

Computes the stress for plants when it is too cold or hot

$$f = \frac{T_{max} - T_{opt}}{T_{opt} - T_{min}}$$

$$s_T = \frac{(T_a - T_{min})(T_{max} - T_a)^f}{(T_{opt} - T_{min})(T_{max} - T_{opt})^f}$$

Parameters

t_air_24 [float] daily air temperature T_a [C]

t_opt [float] optimum air temperature for plant growth T_{opt} [C]

t_min [float] minimum air temperature for plant growth T_{min} [C]

t_max [float] maximum air temperature for plant growth T_{max}

[C]

Returns

stress_temp [float] stress factor for air temperature S_T [-

]

Examples

```
>>> import ETLook.stress as stress >>> stress.stress_temperature(15)
0.8399999999999997
>>> stress.stress_temperature(15, t_opt =20)
0.9451080185178129
>>> stress.stress_temperature(15, t_opt =20, t_min=10)
0.79398148148148151
>>> stress.stress_temperature(15, t_opt =20, t_min=10, t_max=30)
0.75
```

ETLook.stress.stress_vpd

ETLook.stress.**stress_vpd**(*vpd_24*, *vpd_slope=-0.3*)

Computes the stress for plants if the vpd increases too much. With lower slopes the stress increases faster.

The slope of the curve is between -0.3 and -0.7

$$S_v = m \ln(0.1\Delta_e + \frac{1}{2}) + 1$$

Parameters

vpd_24 [float] daily vapour pressure deficit Δ_e [mbar]

vpd_slope [float] vapour pressure stress curve slope m [mbar-1]

Returns

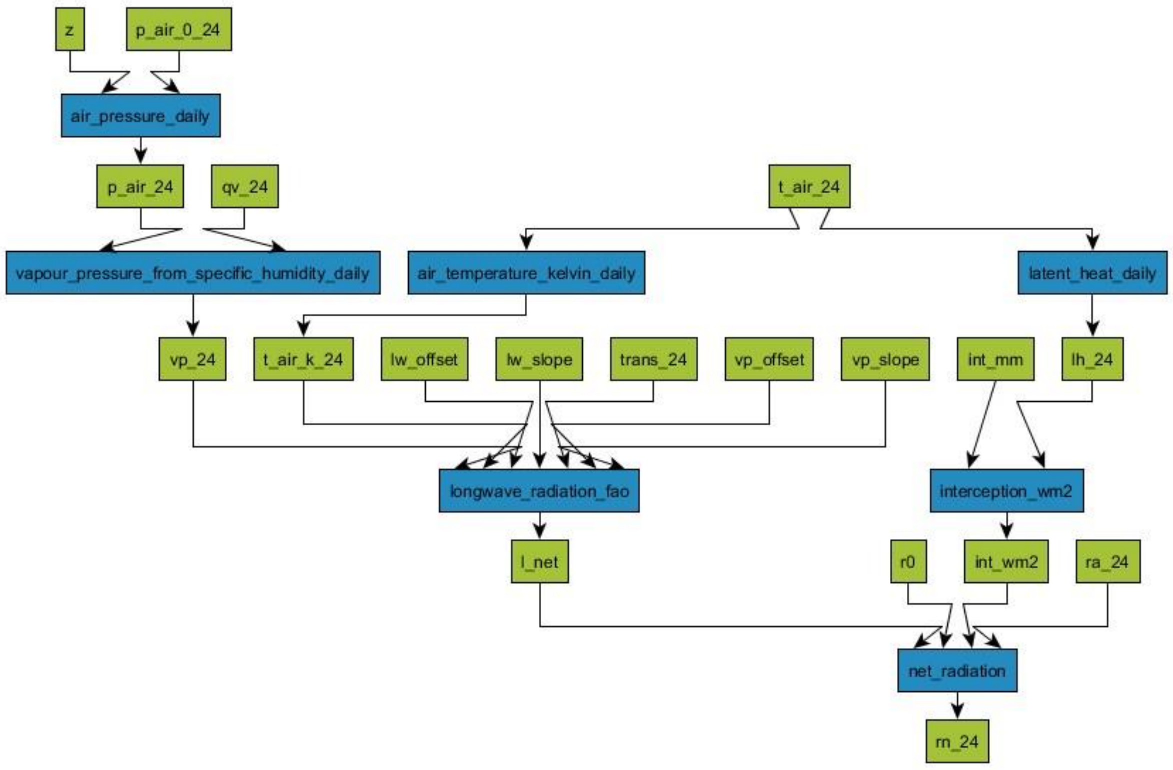
stress_vpd [float] stress factor for vapour pressure deficit S_v [-]

Examples

```
>>> import ETLook.stress as stress
>>> stress.stress_vpd(15)
0.79205584583201638
>>> stress.stress_vpd(15, vpd_slope=-0.7)
0.51479697360803833
>>> stress.stress_vpd(15, vpd_slope=-0.3)
0.79205584583201638
```

5.5.4 Net radiation

The fourth graph depicted in the *net radiation network* below shows the calculation graph for the net radiation. The net radiation is one of the components of the radiation balance.



`ETLook.meteo.latent_heat_daily` (t_air_24)	Like `latent_heat()` but as a daily average
`ETLook.meteo.air_pressure_daily` (z[, p_air_0_24])	Like `air_pressure()` but as a daily average
`ETLook.meteo.air_temperature_kelvin_daily` (…)	Like `air_temperature_kelvin()` but as a daily average
`ETLook.meteo.vapour_pressure_from_specific_humidity_daily` (…)	Like `vapour_pressure_from_specific_humidity()` but as a daily average
`ETLook.radiation.interception_wm2` (int_mm, lh_24)	Computes the energy equivalent for the interception in Wm-2 if it is provide in mm/day
`ETLook.radiation.longwave_radiation_fao` (…)	Computes the net longwave radiation according to the FAO 56 manual.
`ETLook.radiation.net_radiation` (r0, ra_24, …)	Computes the net radiation

ETLook.meteo.vapour_pressure_from_specific_humidity_daily

ETLook.meteo.**vapour_pressure_from_specific_humidity_daily**(qv_24, p_air_24)

Like *vapour_pressure_from_specific_humidity()* but as a daily average

Parameters

qv_24 [float] daily specific humidity $q_{v,24}$ [kg/kg]

p_air_24 [float] daily air pressure P_{24} [mbar]

Returns

vp_24 [float] daily vapour pressure $e_{a,24}$ [mbar]

ETLook.radiation.interception_wm2

ETLook.radiation.**interception_wm2**(int_mm, lh_24)

Computes the energy equivalent for the interception in Wm-2 if it is provide in mm/day

$$I = \frac{\lambda I^*}{86400}$$

Parameters int_mm : float

interception I^* [mm day-1]

lh_24 : float

daily latent heat for evaporation λ [J kg-1]

Returns int_wm2 : float

interception I [W m-2]

Examples

```
>>> import ETLook.radiation as rad
>>> import ETLook.meteo as meteo
>>> lh = meteo.latent_heat_daily(20.0)
>>> rad.interception_wm2(1.0, lh)
28.40023148148148
```

ETLook.radiation.longwave_radiation_fao

ETLook.radiation.**longwave_radiation_fao**(*t_air_k_24*, *vp_24*, *trans_24*, *vp_slope=0.14*, *vp_offset=0.34*, *lw_slope=1.35*, *lw_offset=-0.35*)

Computes the net longwave radiation according to the FAO 56 manual.

$$L^* = \sigma \left(T_{a,K}\right)^4 \left(vp_{off} - vp_{slope}\sqrt{0.1e_a}\right) \left(lw_{slope}\frac{\tau}{0.75} + lw_{off}\right)$$

where the following constant is used

- σ = Stefan Boltzmann constant = 5.67 e-8 J s-1 m-2 K-4

Parameters t_air_k_24 : float

daily air temperature in Kelvin $T_{a,K}$ [-]

vp_24 : float

daily vapour pressure e_a [mbar]

trans_24 : float

daily atmospheric transmissivity

τ [-]

vp_slope : float

slope of the vp-term in the

FAO-56 longwave radiation

relationship vp_{slope} [-]

vp_offset : float

offset of the vp-term in the FAO-56 longwave radiation relationship vp_{off}

[-]

lw_slope : float

slope of the tau-term in the FAO-56 longwave radiation relationship

lw_{slope} [-]

lw_offset : float

offset of the tau-term in the FAO-56 longwave radiation relationship lw_{off}

[-]

Returns l_net : float

daily net longwave radiation L^* [Wm-2]

Examples

```
>>> import ETLook.radiation as rad >>> rad.longwave_radiation_fao(t_air_k=302.5, vp=10.3, trans_24=0.6)
68.594182173686306
```

ETLook.radiation.net_radiation

ETLook.radiation.**net_radiation**(*r0*, *ra_24*, *l_net*, *int_wm2*)

Computes the net radiation

$$Q^* = \left[(1 - \alpha_0)S^{\downarrow} - L^* - I\right]$$

Parameters r0 : float

albedo α_0 [-]

ra_24 : float

daily solar radiation $S^{\downarrow}$ [Wm-2]

l_net : float

daily net longwave radiation L^* [wm-2]

int_wm2 : float

interception I [Wm-2]

Returns rn_24 : float

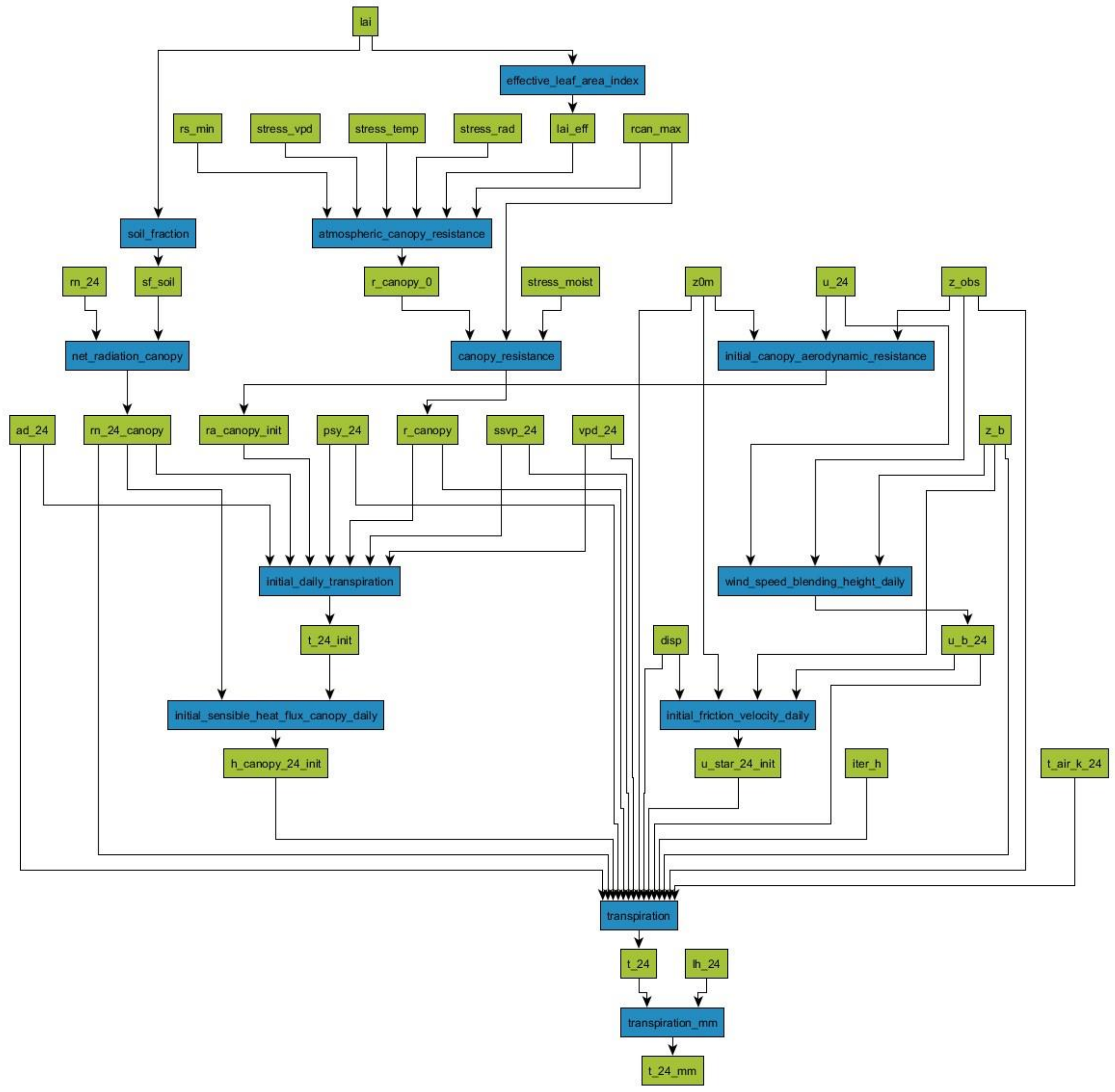
daily net radiation Q^* [Wm-2]

Examples

```
>>> import ETLook.radiation as rad >>> rad.net_radiation(r0=0.10, ra_24=123., l_net=24., int_wm2=0)
86.7
```

5.5.5 Transpiration

The final graph depicts the *transpiration network* below. A list of functions is provided with links to their description. Some inputs are defined in the graphs above.



`ETLook.leaf.effective_leaf_area_index` (lai)	Computes effective leaf area index, this describes the leaf area which actively participates in transpiration.
`ETLook.radiation.soil_fraction` (lai)	Computes the effect of the vegetation has in separating the net radiation into a soil and canopy component.
`ETLook.resistance.atmospheric_canopy_resistance` (…)	Computes canopy resistance excluding soil moisture stress
`ETLook.resistance.canopy_resistance` (…[, …])	Computes canopy resistance
`ETLook.neutral.initial_canopy_aerodynamic_resistance` (…)	Computes the aerodynamic resistance for a canopy soil without stability corrections $r^0_{a,}$.
`ETLook.radiation.net_radiation_canopy` (rn_24, …)	Computes the net radiation for the canopy
`ETLook.meteo.wind_speed_blending_height_daily` (u_24)	Like `wind_speed_blending_height()` but as a daily average
`ETLook.neutral.initial_daily_transpiration` (…)	Computes the soil evaporation based on the Penman Monteith equation adapted for soil.
`ETLook.unstable.initial_friction_velocity_daily` (…)	Computes the initial friction velocity without using stability corrections.
`ETLook.unstable.initial_sensible_heat_flux_canopy_daily` (…)	Computes the initial sensible heat flux before the iteration which solves the stability corrections.
`ETLook.unstable.transpiration` (rn_24_canopy, …)	Computes the transpiration using an iterative approach.
`ETLook.unstable.transpiration_mm` (t_24, lh_24)	Computes the canopy transpiration based on the Penman Monteith equation adapted for canopy.

ETLook.leaf.effective_leaf_area_index

ETLook.leaf.**effective_leaf_area_index**(lai)

Computes effective leaf area index, this describes the leaf area which actively participates in transpiration. It is based on the actual leaf area index and an extinction function. So with a higher leaf area index the effective leaf area index is a smaller percentage of the total leaf area index.

$$I_{lai,eff} = \frac{I_{lai}}{0.3I_{lai} + 1.2}$$

Parameters

lai [float] Leaf area index I_{lai} [-]

Returns

lai_eff [float] effective leaf area index $I_{lai,eff}$ [-]

Examples

```
>>> from ETLook import leaf
>>> leaf.effective_leaf_area_index(3.0)
1.4285714285714288
>>> leaf.effective_leaf_area_index(5.0)
1.8518518518518516
```

ETLook.radiation.soil_fraction

ETLook.radiation.**soil_fraction**(lai)

Computes the effect of the vegetation has in separating the net radiation into a soil and canopy component. If the canopy has a full cover almost no radiation reaches the soil.

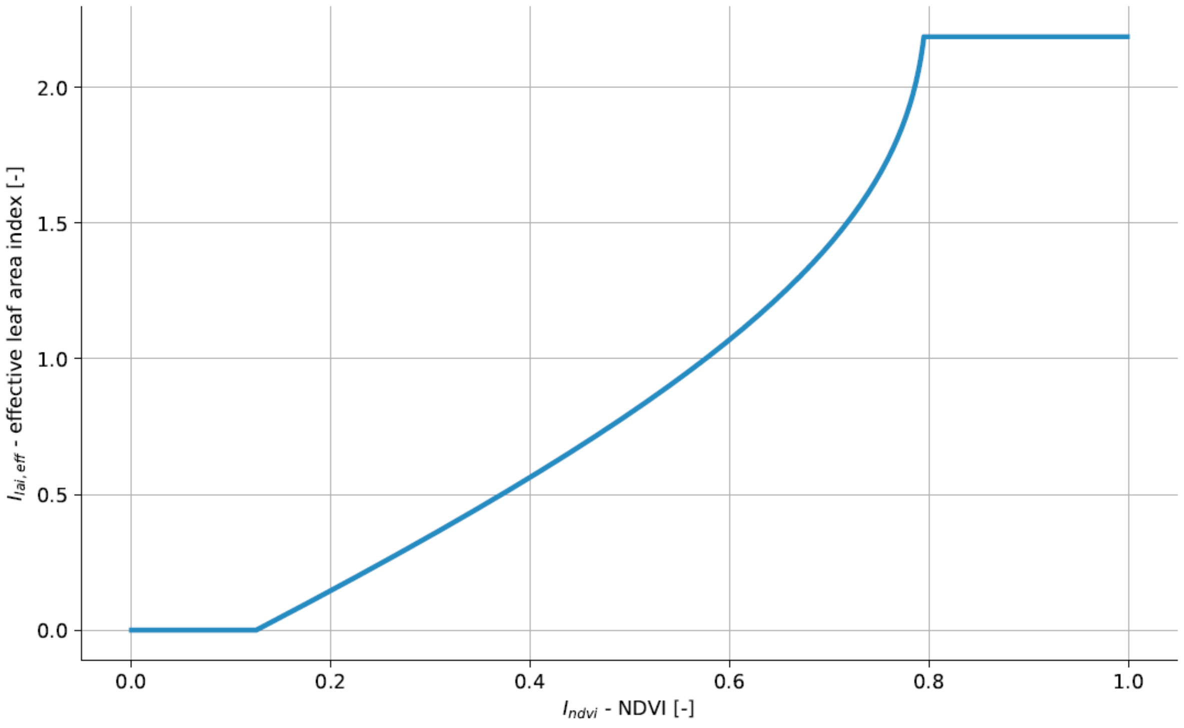
$$s_f = \exp(-0.6 * I_{lai})$$

Parameters lai : float

leaf area index I_{lai} [-]

Returns sf_soil : float

soil fraction s_f [-]



Examples

```
>>> import ETLook.radiation as rad
>>> rad.soil_fraction(3.0)
0.16529888822158656
```

ETLook.resistance.atmospheric_canopy_resistance

ETLook.resistance.**atmospheric_canopy_resistance**(lai_eff, stress_rad, stress_vpd, *stress_temp*, *rs_min=70*, *rcan_max=1000000.0*)

Computes canopy resistance excluding soil moisture stress

$$r_{canopy,0} = \left(\frac{r_{s,min}}{I_{lai,eff}}\right)\left(\frac{1}{S_T S_V S_r}\right)$$

Parameters

lai_eff [float] effective leaf area index $I_{lai,eff}$ [-]

stress_temp [float] stress factor for air temperature S_t [-]

stress_vpd [float] stress factor for vapour pressure deficit S_v [-

]

stress_rad [float] stress factor for radiation S_r [-]

rs_min [float] Minimal stomatal resistance r_{smin} [sm-1]

rcan_max [float] Maximum stomatal resistance r_{canmax} [sm-1]

Returns

r_canopy_0 [float] atmospheric canopy resistance $r_{canopy,0}$ [sm-1]

Examples

```
>>> import ETLook.resistance as res >>> res.atmospheric_canopy_resistance(0.9, 0.4, 0.9, 0.94)
229.839768846861
```

ETLook.resistance.canopy_resistance

ETLook.resistance.**canopy_resistance**(*r_canopy_0*, *stress_moist*, *rcan_max=1000000.0*)

Computes canopy resistance

$$r_{canopy} = \frac{r_{canopy,0}}{S_m}$$

Parameters

r_canopy_0 [float] Atmospheric canopy resistance $r_{canopy0}$ [sm-1]

stress_moist [float] stress factor for root zone soil moisture S_m [-]

rcan_max [float] Maximum stomatal resistance r_{canmax} [sm-1]

Returns

r_canopy [float] canopy resistance r_{canopy} [sm-1]

Examples

```
>>> import ETLook.resistance as res >>> res.canopy_resistance(218, 0.8)
272.5
```

ETLook.neutral.initial_canopy_aerodynamic_resistance

ETLook.neutral.**initial_canopy_aerodynamic_resistance**(u_24, z0m, z_obs=2)

Computes the aerodynamic resistance for a canopy soil without stability corrections $r_{a,}{}^{0}$.

$$r_{a,canopy}^{0} = \frac{\ln\left(\frac{z_{obs}}{z_{0,m}}\right)\ln\left(\frac{z_{obs}}{0.1z_{0,m}}\right)}{k^2 u_{obs}}$$

where the following constants are used

- k = karman constant = 0.41 [-]

The factor 0.1 is the ratio between the surface roughness for momentum and heat.

Parameters

u_24 [float] daily wind speed at observation height u_obs

[m/s]

z0m [float] roughness length $z_{0,m}$ [m]

z_obs [float] observation height z_{obs} [m]

Returns

ra_canopy_init [float] canopy resistance without stability corrections $r_{a,canopy}{}^0$

[s/m]

ETLook.radiation.net_radiation_canopy

ETLook.radiation.**net_radiation_canopy**(rn_24, sf_soil)

Computes the net radiation for the canopy

$$Q^*_{canopy} = (1 - s_f)\, Q^*$$

Parameters

rn_24 : float

net radiation Q^* [Wm-2]

sf_soil : float

soil fraction s_f [-]

Returns

rn_24_canopy : float

net radiation for the canopy Q^*_{canopy} [Wm-2]

Examples

```
>>> import ETLook.radiation as rad >>> rad.net_radiation_canopy(rn_24=200, sf_soil=0.4)
120.0
```

ETLook.meteo.wind_speed_blending_height_daily

ETLook.meteo.**wind_speed_blending_height_daily**(u_24, z_obs=2, z_b=100)

Like *wind_speed_blending_height()* but as a daily average

Parameters

u_24 [float] daily wind speed at observation height $u_{obs,24}$

[m/s]

z_obs [float] observation height of wind speed z_{obs} [m]

z_b [float] blending height z_b [m]

Returns

u_b_24 [float] daily wind speed at blending height $u_{b,24}$

[m/s]

ETLook.neutral.initial_daily_transpiration

ETLook.neutral.**initial_daily_transpiration**(*rn_24_canopy*, *ssvp_24*, *ad_24*, *vpd_24*, *psy_24*, *r_canopy*, *ra_canopy_init*) Computes the soil evaporation based on the Penman Monteith equation adapted for soil.

$$T_0 = \frac{\Delta\left(Q^*_{canopy}\right) + \rho c_p \frac{\Delta_e}{r_{a,canopy}}}{\Delta + \gamma\left(1 + \frac{r_{canopy}}{r_{a,canopy}}\right)}$$

where the following constants are used

- c_p specific heat for dry air = 1004 [J kg-1 K-1]
- k = karman constant = 0.41 [-]

Parameters

rn_24_canopy [float] daily net radiation for the canopyQ^*_{soil} [W m-2]

ssvp_24 [float] daily slope of saturated vapour pressure curve Δ [mbar K-1]

ad_24 [float] daily air density ρ [kg m-3]

vpd_24 [float] daily vapour pressure deficit Δ_e [mbar]

psy_24 [float] daily psychrometric constant γ [mbar K-1]

r_canopy [float] canopy resistance r_{canopy} [m s-1]

ra_canopy_init [float] initial canopy aerodynamic resistance $r_{a,canopy}$ [m s-1]

Returns

t_24_init [float] initial estimate radiation equivalent daily transpiration T^0 [W m-2]

ETLook.unstable.initial_friction_velocity_daily

ETLook.unstable.**initial_friction_velocity_daily**(u_b_24, z0m, disp, z_b=100)

Computes the initial friction velocity without using stability corrections.

$$u_* = \frac{k u_b}{ln\left(\frac{z_b - d}{z_{0,m}}\right)}$$

Parameters

u_b_24 : float

daily wind speed at blending heigt u_b [m s-1]

z0m : float

surface roughness $z_{0,m}$ [m]

disp : float

displacement height d [m]

z_b : float

blending height z_b [m]

Returns u_star_24_init : float

initial estimate of the daily friction velocity u_* [m s-1]

ETLook.unstable.initial_sensible_heat_flux_canopy_daily

ETLook.unstable.**initial_sensible_heat_flux_canopy_daily**(rn_24_canopy, *t_24_init*)

Computes the initial sensible heat flux before the iteration which solves the stability corrections. The first estimation of transpiration is used to estimate the initial sensible heat flux.

$$H_{canopy} = Q^*_{canopy} - T$$

Parameters rn_24_canopy : float

daily net radiation for the canopy Q^*_{canopy}[W m-2]

t_24_init : float

initial estimate of daily transpiration T [W m-

2]

Returns h_canopy_24_init : float

initial estimate of the sensible heat flux H_{canopy} [W m-2]

ETLook.unstable.transpiration

ETLook.unstable.**transpiration**(*rn_24_canopy*, *ssvp_24*, *ad_24*, *vpd_24*, *psy_24*, *r_canopy*, *h_canopy_24_init*, *t_air_k_24*, *u_star_24_init*, *z0m*, *disp*, *u_b_24*, *z_obs=2*, *z_b=100*, *iter_h=3*)

Computes the transpiration using an iterative approach. The iteration is needed to compute the aerodynamical resistance.Iteration stops either after five iterations orif the difference between two subsequent estimations is less than 0.01. The iteration is started with an estimate on H using the initial guess without stability corrections. Subsequent iterations use the guess with stability corrections.

$$T = \frac{\Delta\left(Q^*_{canopy}\right) + \rho c_p\ frac\Delta_e r_{a,canopy}}{\Delta + \gamma\left(1 + \frac{r_{canopy}}{r_{a,canopy}}\right)}$$

Parameters rn_24_canopy : float

net radiation for the canopy Q^*_{canopy} [Wm-2]

ssvp_24 : float

daily slope of saturated vapour pressure curve

Δ_{24} [mbar K-1]

ad_24 : float

daily air density ρ_{24} [kg m-3]

vpd_24 : float

daily vapour pressure deficit $\Delta_{e,24}$ [mbar]

psy_24 : float

daily psychrometric constant γ_{24} [mbar K-1]

r_canopy : float

canopy resistance r_{canopy} [sm-1]

h_canopy_24_init : float

initial estimate of the sensible heat flux H^{canopy}

[W m-2]

t_air_k_24 : float

daily air tempererature in kelvin T_a [K]

u_star_24_init : float

initial estimate of the daily friction velocity u_* [m

s-1] z0m : float roughness length $z_{0,m}$

[m]

disp : float

displacement height d [m]

u_b_24 : float

daily windspeed at blending height u_b [m] z_b :

float blending height z_b [m]

z_obs : float

observation height z_{obs} [m]

iter_h : integer

number of iterations for sensible heat flux n_h [-]

Returns t_24 : float

daily transpiration energy equivalent T_{24} [W m-

2]

ETLook.unstable.transpiration_mm

ETLook.unstable.**transpiration_mm**(t_24, lh_24)

Computes the canopy transpiration based on the Penman Monteith equation adapted for canopy.

$$T = Td_{sec}\lambda_{24}$$

where the following constants are used

- d_{sec} seconds in the day = 86400 [s]

Parameters t_24 : float

daily transpiration energy equivalent E^0 [W m-2]

lh_24 : float

daily latent heat of evaporation λ_{24} [J/kg]

Returns t_24_mm : float

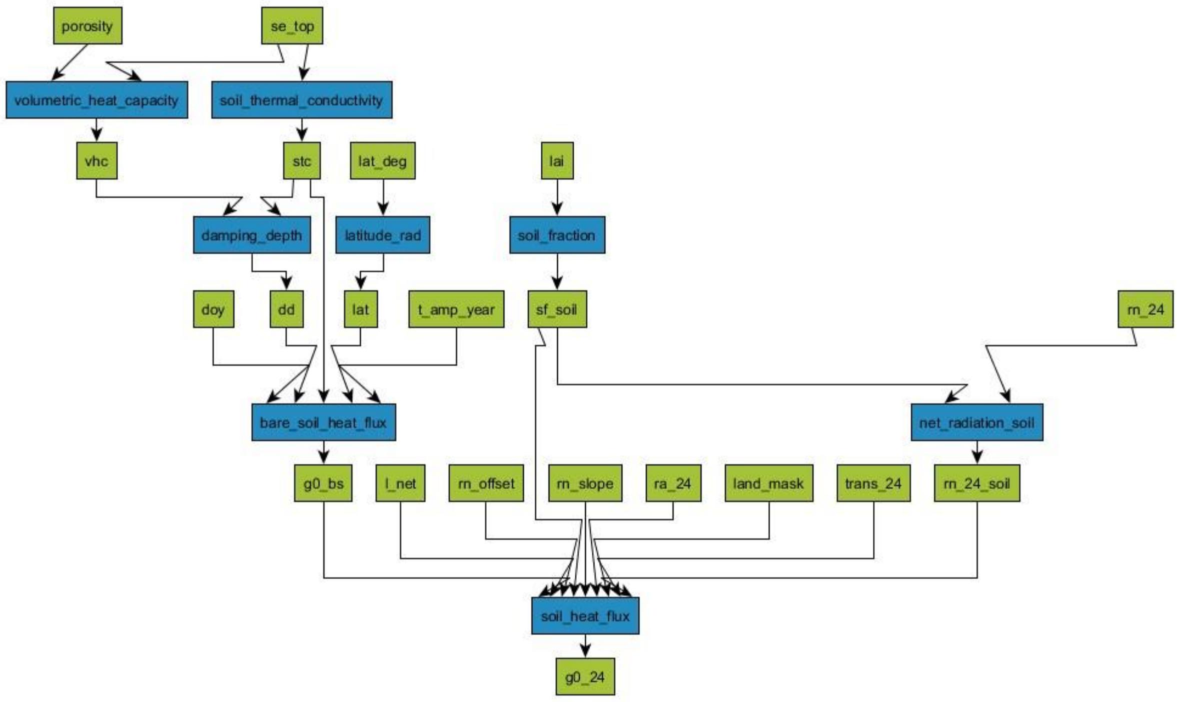
daily transpiration in mm T [mm d-1]

5.6 Evaporation

The evaporation calculation procedure is presented below.

5.6.1 Soil Heat Flux

First the soil heat flux is calculated separately. This calculation is not needed for the transpiration. The calculation network is shown in the *soil heat flux network* below. A list of functions is provided with links to their description.



`ETLook.radiation.bare_soil_heat_flux` (doy, …)	Computes the bare soil heat flux
`ETLook.radiation.damping_depth` (stc, vhc)	Computes the damping depth
`ETLook.radiation.soil_thermal_conductivity` (se_top)	Computes the soil thermal conductivity
`ETLook.radiation.volumetric_heat_capacity` ([…])	Computes the volumetric heat capacity of the soil
`ETLook.solar_radiation.latitude_rad` (lat_deg)	Converts latitude from degrees to radians.
`ETLook.radiation.soil_fraction` (lai)	Computes the effect of the vegetation has in separating the net radiation into a soil and canopy component.
`ETLook.radiation.net_radiation_soil` (rn_24, …)	Computes the net radiation for the soil
`ETLook.radiation.soil_heat_flux` (g0_bs, …)	Computes the soil heat flux

ETLook.radiation.bare_soil_heat_flux

ETLook.radiation.**bare_soil_heat_flux**(*doy*, *dd*, *stc*, *t_amp_year*, *lat*) Computes the bare soil heat flux

$$G_0 = \frac{\sqrt{2}A_{t,year}k\sin\left(\frac{2\pi J}{P} - \frac{\pi}{4}\right)}{z_d}$$

where the following constant is used

- P period (seconds within a year)

The term $-\frac{\pi}{4}$ is a phase shift for northern latitudes. For southern latitudes the phase shift will be $-\frac{\pi}{4} + \pi$

Parameters stc : float

soil thermal conductivity k [W m-1 K-1]

dd : float

damping depth z_d [m]

t_amp_year : float

yearly air temperature amplitude $A_{t,year}$ [m]

doy : float

julian day of the year J [-]

lat : float

latitude λ [rad]

Returns g0_bs : float

bare soil heat flux G_0 [m]

Examples

```
>>> import ETLook.radiation as rad
>>> stc = rad.soil_thermal_conductivity(se_top=1.0)
>>> vhc = rad.volumetric_heat_capacity(se_top=1.0)
>>> dd = damping_depth(stc,vhc)
>>> rad.bare_soil_heat_flux(126, dd, stc, t_amp_year=13.4, lat=40*(math.pi/180.
 ↪0))
 array([ 45.82350561])
```

ETLook.radiation.damping_depth

ETLook.radiation.**damping_depth**(*stc*, *vhc*)

Computes the damping depth

$$z_d = \sqrt{\frac{2kP}{2\pi\rho c_p}}$$

with the following constant

- P period (seconds within a year)

Parameters stc : float

soil thermal conductivity k [W m-1 K-1]

vhc : float volumetric heat capacity ρc_p [J m-3 K-1]

Returns dd : float

damping depth z_d [m]

Examples

```
>>> import ETLook.radiation as rad >>> rad.damping_depth(stc=0.9, vhc=volumetric_heat_capacity())
0.54514600029013294
```

ETLook.radiation.soil_thermal_conductivity

ETLook.radiation.**soil_thermal_conductivity**(se_top)

Computes the soil thermal conductivity

$$k = 0.15 + 18.5S_{e,top}$$

Parameters se_top : float

effective saturation of the topsoil $S_{e,top}$ [-]

Returns stc : float

soil thermal conductivity k [W m-1 K-1]

Examples

```
>>> import ETLook.radiation as rad >>> rad.soil_thermal_conductivity(se_top=0.4)
0.8900000000000001
```

ETLook.radiation.volumetric_heat_capacity

ETLook.radiation.**volumetric_heat_capacity**(se_top=1.0, porosity=0.4)

Computes the volumetric heat capacity of the soil

$$\rho c_p = 10e^6 \left[(1-\varphi)^2 + 2.5\varphi + 4.2\varphi S_{e,top}\right]$$

Parameters se_top : float

effective saturation of the topsoil $S_{e,top}$ [-]

porosity : float

porosity of the soil φ [-]

Returns vhc : float

volumetric heat capacity ρc_p [J m-3 K-1]

Examples

```
>>> import ETLook.radiation as rad >>> rad.volumetric_heat_capacity(se_top=0.4, porosity = 0.5)
23400000.0
```

ETLook.solar_radiation.latitude_rad

ETLook.solar_radiation.**latitude_rad**(lat_deg)

Converts latitude from degrees to radians.

Parameters lat_deg : float

latitude in degrees λ [deg]

Returns lat : float

latitude λ [rad]

ETLook.radiation.net_radiation_soil

ETLook.radiation.**net_radiation_soil**(rn_24, sf_soil)

Computes the net radiation for the soil

$$Q^*_{soil} = s_f Q^*$$

Parameters rn_24 : float

net radiation Q^* [Wm-2]

sf_soil : float

soil fraction s_f[-]

Returns rn_24_soil : float

net radiation for the soil Q^*_{soil} [Wm-2]

Examples

```
>>> import ETLook.radiation as rad >>> rad.net_radiation_soil(rn_24=200, sf_soil=0.4)
80.0
```

ETLook.radiation.soil_heat_flux

ETLook.radiation.**soil_heat_flux**(*g0_bs*, *sf_soil*, *land_mask*, *rn_24_soil*, *trans_24*, *ra_24*, *l_net*, *rn_slope=0.92*, *rn_offset=-61.0*)

Computes the soil heat flux

$$G = s_f G_0$$

Parameters g0_bs : float

bare soil heat flux G_0 [W m-2]

sf_soil : float

soil fraction s_f [-]

land_mask : int

land use classification l [-]

rn_24_soil : float

net radiation for the soil Q^*_{soil} [Wm-2]

trans_24 : float

daily atmospheric transmissivity τ [-]

n_slope : float

slope rn/g0 relation water lws [-]

rn_offset : float

offset rn/g0 relation water lwo [-]

ra_24 : float

daily solar radiation $S^{\downarrow}$ [Wm-2]

l_net : float

daily net longwave radiation L^* [wm-2]

Returns g0_24 : float

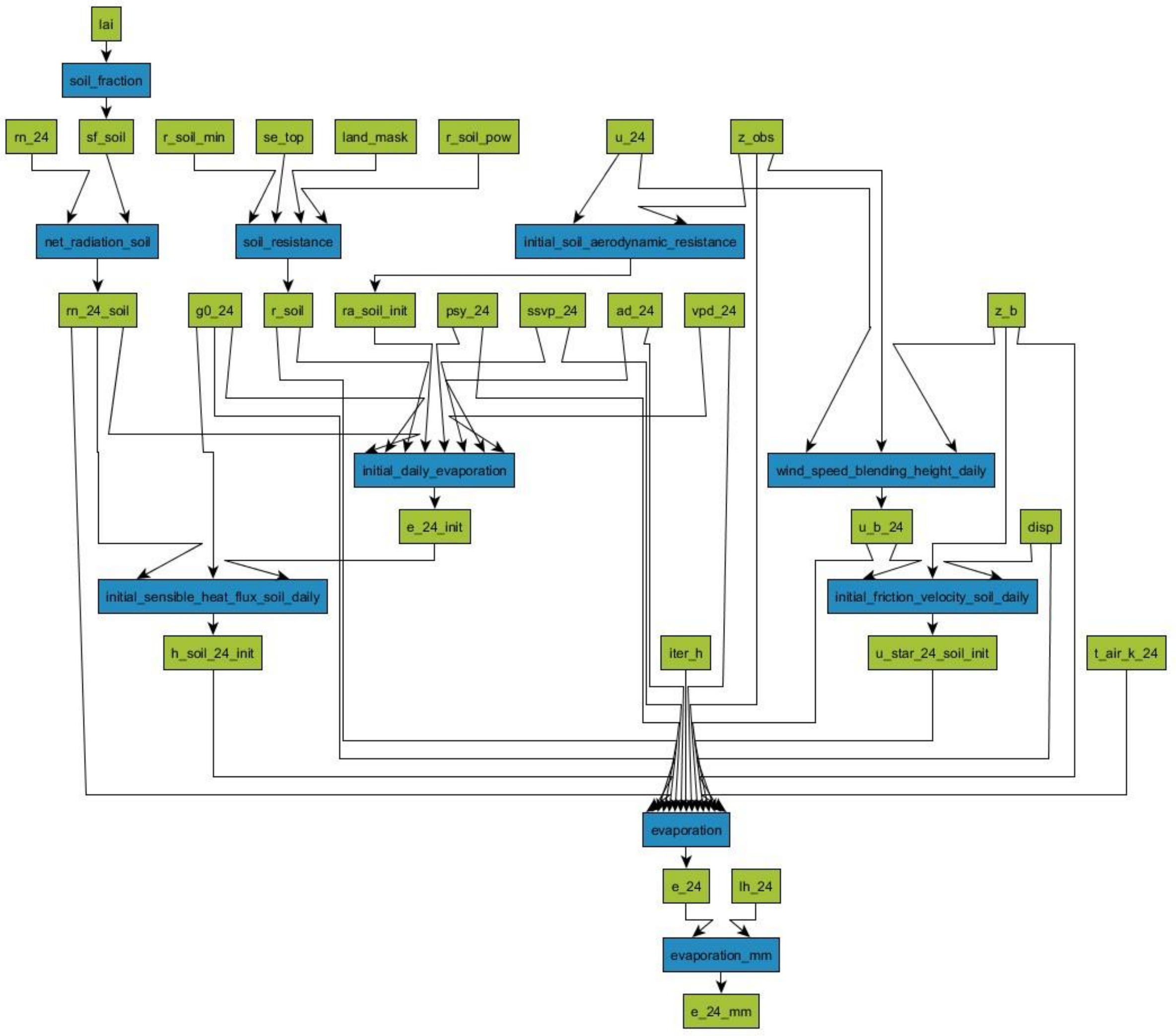
daily soil heat flux G [W m-2]

Examples

```
>>> import ETLook.radiation as rad >>> rad.soil_heat_flux(g0_bs=12.4, sf_soil=0.4)
4.960000000000001
```

5.6.2 Evaporation

The calculation network for the evaporation is shown in the *evaporation figure* below. A list of functions is provided with links to their description.



`ETLook.radiation.soil_fraction` (lai)	Computes the effect of the vegetation has in separating the net radiation into a soil and canopy component.
`ETLook.radiation.net_radiation_soil` (rn_24, …)	Computes the net radiation for the soil
`ETLook.resistance.soil_resistance` (se_top[, …])	Computes soil resistance
`ETLook.neutral.initial_soil_aerodynamic_resistance` (u_24)	Computes the aerodynamic resistance for soil without stability corrections $r^0_{a,soil}$.
`ETLook.neutral.initial_daily_evaporation` (…)	Computes the soil evaporation based on the Penman Monteith equation adapted for soil.
`ETLook.unstable.evaporation` (rn_24_soil, …)	Computes the evaporation using an iterative approach.
`ETLook.unstable.evaporation_mm` (e_24, lh_24)	Computes the soil evaporation based on the Penman Monteith equation adapted for soils.
`ETLook.meteo.wind_speed_blending_height_daily` (u_24)	Like `wind_speed_blending_height()` but as a daily average
`ETLook.unstable.initial_friction_velocity_soil_daily` (…)	Computes the initial firction velocity without using stability corrections.
`ETLook.unstable.initial_sensible_heat_flux_soil_daily` (…)	Computes the initial sensible heat flux before the iteration which solves the stability corrections.
`ETLook.solar_radiation.daily_solar_radiation_toa` (sc, …)	Computes the daily solar radiation at the top of the atmosphere.

ETLook.resistance.soil_resistance

ETLook.resistance.**soil_resistance**(*se_top*, *land_mask=1*, *r_soil_pow=-2.1*, *r_soil_min=800*)

Computes soil resistance

$$r_{soil} = r_{soil,min}\left(S_{e,top}\right)^a$$

Parameters

r_soil_min [float] Minimum soil resistance $r_{soil,min}$ [sm-1]

se_top [float] Top soil effective saturation $S_{e,top}$ [-]

r_soil_pow [float] Power soil resistance function a [-]

land_mask [int] land use classification l [-]

Returns

r_soil [float] soil resistance r_{soil} [sm-1]

Examples

```
>>> import ETLook.resistance as res >>> res.soil_resistance(se_top=0.9)
998.1153098304111
```

ETLook.neutral.initial_soil_aerodynamic_resistance

ETLook.neutral.**initial_soil_aerodynamic_resistance**(u_24, z_obs=2)

Computes the aerodynamic resistance for soil without stability corrections $r_{a,soil}{}^0$.

$$r^0_{a,soil} = \frac{\ln\left(\frac{z_{obs}}{z_{0,soil}}\right)\ln\left(\frac{z_{obs}}{0.1z_{0,soil}}\right)}{k^2 u_{obs}}$$

where the following constants are used

- $z_{0,soil}$ soil roughness = 0.001 [m]
- k = karman constant = 0.41 [-]

The factor 0.1 is the ratio between the surface roughness for momentum and heat.

Parameters

u_24 [float] daily wind speed at observation height u_obs [m/s]

z_obs [float] observation height z_{obs} [m]

Returns

ra_soil_init [float] aerodynamic resistance without stability corrections $r_{a,soil}{}^0$ [s/m]

ETLook.neutral.initial_daily_evaporation

ETLook.neutral.**initial_daily_evaporation**(*rn_24_soil*, *g0_24*, *ssvp_24*, *ad_24*, *vpd_24*, *psy_24*, *r_soil*, *ra_soil_init*) Computes the soil evaporation based on the Penman Monteith equation adapted for soil.

$$E^0 = \frac{\Delta\left(Q^*_{soil} - G\right) + \rho c_p \frac{\Delta_e}{r_{a,soil}}}{\Delta + \gamma\left(1 + \frac{r_{soil}}{r_{a,soil}}\right)}$$

where the following constants are used

- c_p specific heat for dry air = 1004 [J kg-1 K-1]

- k = karman constant = 0.41 [-]

  Parameters

  rn_24_soil [float] daily net radiation for soilQ^*_{soil}[W m-2]

  g0_24 [float] daily soil heat flux G [W m-2]

  ssvp_24 [float] daily slope of saturated vapour pressure curve Δ [mbar K-1]

  ad_24 [float] daily air density ρ [kg m-3]

  vpd_24 [float] daily vapour pressure deficit Δ_e [mbar]

  psy_24 [float] daily psychrometric constant γ [mbar K-1]

  r_soil [float] soil resistance r_{soil} [m s-1]

  ra_soil_init [float] initial soil aerodynamic resistance $r_{a,soil}$ [m s-1]

  Returns

  e_24_init [float] initial estimate radiation equivalent daily evaporation E^0 [W m-2]

ETLook.unstable.evaporation

ETLook.unstable.**evaporation**(*rn_24_soil*, *g0_24*, *ssvp_24*, *ad_24*, *vpd_24*, *psy_24*, *r_soil*,

h_soil_24_init, *t_air_k_24*, *u_star_24_soil_init*, *disp*, *u_b_24*,
z_b=100, *z_obs=2*, *iter_h=3*)

Computes the evaporation using an iterative approach. The iteration is needed to compute the aerodynamic resistance.Iteration stops either after five iterations or if the difference between two subsequent estimations is less than 0.01. The iteration is started with an estimate on H using the initial guess without stability corrections. Subsequent iterations use the guess with stability corrections.

$$E = \frac{\Delta\left(Q^*_{soil} - G\right) + \rho c_p \frac{\Delta_e}{r_{a,soil}}}{\Delta + \gamma\left(1 + \frac{r_{soil}}{r_{a,soil}}\right)}$$

Parameters rn_24_soil : float

net radiation for the soil Q^*_{canopy} [Wm-2]

g0_24 : float

daily soil heat flux G [Wm-2]

ssvp_24 : float

daily slope of saturated vapour pressure curve Δ_{24}

[mbar K-1]

ad_24 : float

daily air density ρ_{24} [kg m-3]

vpd_24 : float

daily vapour pressure deficit $\Delta_{e,24}$ [mbar]

psy_24 : float

daily psychrometric constant γ_{24} [mbar K-1]

r_soil : float

soil resistance r_{soil} [sm-1]

h_soil_24_init : float

initial estimate of the sensible heat flux for soil H^{soil} [W

m-2]

t_air_k_24 : float

daily air temperature in kelvin T_a [K]

u_star_24_soil_init : float

initial estimate of the daily friction velocity for soil u_*

[m s-1] disp : float displacement height d [m]

u_b_24 : float daily wind speed at blending

height u_b [m] z_b : float blending height z_b [m]

z_obs : float

observation height z_{obs} [m]

iter_h : integer

number of iterations for sensible heat flux n_h [-]

Returns e_24 : float

daily evaporation energy equivalent E_{24} [W m-2]

ETLook.unstable.evaporation_mm

ETLook.unstable.**evaporation_mm**(e_24, lh_24)

Computes the soil evaporation based on the Penman Monteith equation adapted for soils.

$$E = Ed_{sec}\lambda_{24}$$

where the following constants are used

- d_{sec} seconds in the day = 86400 [s]

Parameters e_24 : float

daily evaporation energy equivalent E^0 [W m-2] lh_24 : float

daily latent heat of evaporation λ_{24} [J/kg]

Returns e_24_mm : float

daily evaporation in mm E [mm d-1]

ETLook.unstable.initial_friction_velocity_soil_daily

ETLook.unstable.**initial_friction_velocity_soil_daily**(u_b_24, disp, z_b=100)

Computes the initial firction velocity without using stability corrections.

$$u_* = \frac{ku_b}{ln\left(\frac{z_b-d}{z_{0,m}}\right)}$$

Parameters u_b_24 : float

daily wind speed at blending heigt u_b [m s-1]

disp : float

displacement height d [m]

z_b : float

blending height z_b [m]

Returns u_star_24_soil_init : float

initial estimate of the daily friction velocity for soil u_* [m s-1]

ETLook.unstable.initial_sensible_heat_flux_soil_daily

ETLook.unstable.**initial_sensible_heat_flux_soil_daily**(rn_24_soil, e_24_init, g0_24)

Computes the initial sensible heat flux before the iteration which solves the stability corrections. The first estimation of transpiration is used to estimate the initial sensible heat flux.

$$H_{soil} = Q^*_{soil} - G_0 - E$$

Parameters rn_24_soil : float

daily net radiation for the soil Q^*_{canopy}[W m-2]

g0_24 : float daily soil heat flux G_0 [W m-2]

e_24_init : float

initial estimate of daily evaporation E [W m-2]

Returns h_soil_24_init : float

initial estimate of the sensible heat flux H_{canopy} [W m-2]

ETLook.solar_radiation.daily_solar_radiation_toa

ETLook.solar_radiation.**daily_solar_radiation_toa**(*sc*, *decl*, *iesd*, *lat*, *slope=0*, *aspect=0*) Computes the daily solar radiation at the top of the atmosphere.

$$S_{toa} = S_{sun} d_r \int_{i=-\pi}^{i=\pi} S_{toa}^i$$

Parameters iesd : float

inverse earth sun distance d_r [AU]

decl : float

solar declination δ [rad]

sc : float

seasonal correction s_c [hours]

lat : float

latitude λ [rad]

slope : float

slope Δ [rad]

aspect : float

aspect (0 is north; pi is south) α [rad]

Returns ra_24_toa : float

daily solar radiation at the top of atmosphere S_{toa} [Wm-2]

Examples

```
>>> import ETLook.solar_radiation as solrad
>>> from math import pi
>>> doy = 1
>>> sc = solrad.seasonal_correction(doy)
>>> decl = solrad.declination(doy)
>>> iesd = solrad.inverse_earth_sun_distance(doy) >>> solrad.daily_solar_radiation_toa(sc, decl, iesd, lat=25*pi/180.0)
265.74072308978026
```

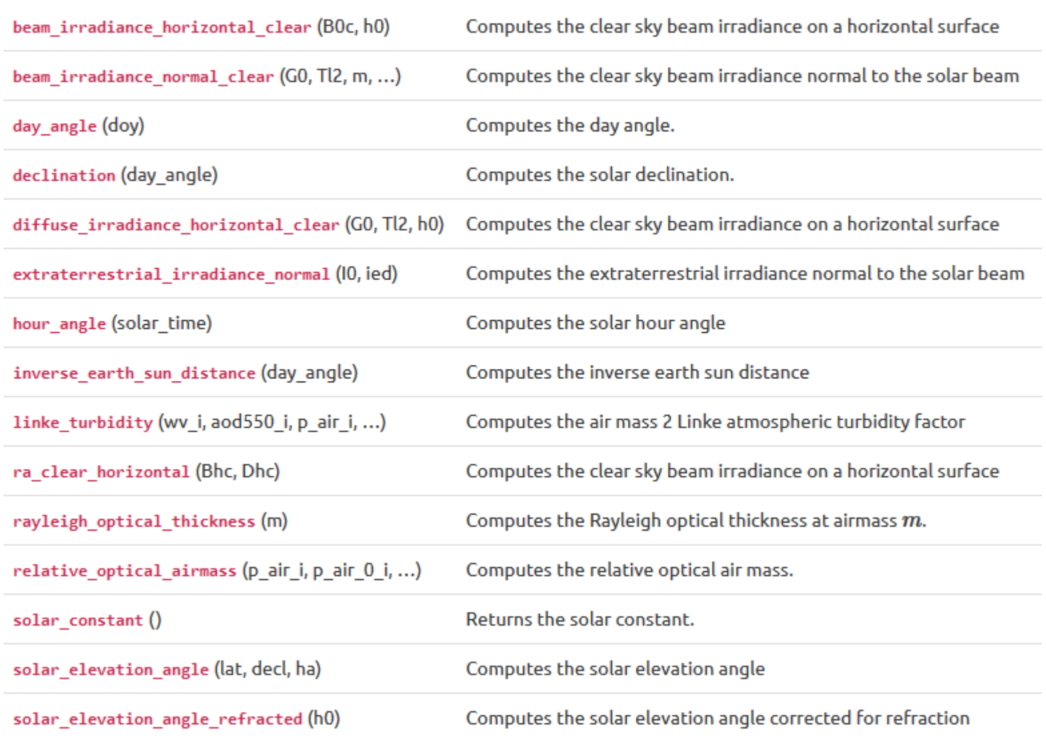
5.7 ETLook functions (ETLook API REFERENCE)

Within the *ETLook* module all physical and empirical functions related to the calculation of the soil moisture, interceptin, evaporation and transpiration are provided. These functions listed here can be used to build function chains.

5.7.1 Instantaneous Radiation (ETLook.clear_sky_radiation)

The clear_sky_radiation module contains all functions related to the calculation of (instantaneous) clear sky radation. Most of these functions are based upon Šúri et Hofierka (2004)[19].

beam_irradiance_horizontal_clear(B0c, Computes the clear sky beam irradiance on a horizontal h0)
surface



`beam_irradiance_horizontal_clear` (B0c, h0)	Computes the clear sky beam irradiance on a horizontal surface
`beam_irradiance_normal_clear` (G0, Tl2, m, ...)	Computes the clear sky beam irradiance normal to the solar beam
`day_angle` (doy)	Computes the day angle.
`declination` (day_angle)	Computes the solar declination.
`diffuse_irradiance_horizontal_clear` (G0, Tl2, h0)	Computes the clear sky beam irradiance on a horizontal surface
`extraterrestrial_irradiance_normal` (I0, ied)	Computes the extraterrestrial irradiance normal to the solar beam
`hour_angle` (solar_time)	Computes the solar hour angle
`inverse_earth_sun_distance` (day_angle)	Computes the inverse earth sun distance
`linke_turbidity` (wv_i, aod550_i, p_air_i, ...)	Computes the air mass 2 Linke atmospheric turbidity factor
`ra_clear_horizontal` (Bhc, Dhc)	Computes the clear sky beam irradiance on a horizontal surface
`rayleigh_optical_thickness` (m)	Computes the Rayleigh optical thickness at airmass m.
`relative_optical_airmass` (p_air_i, p_air_0_i, ...)	Computes the relative optical air mass.
`solar_constant` ()	Returns the solar constant.
`solar_elevation_angle` (lat, decl, ha)	Computes the solar elevation angle
`solar_elevation_angle_refracted` (h0)	Computes the solar elevation angle corrected for refraction

ETLook.clear_sky_radiation.beam_irradiance_horizontal_clear

ETLook.clear_sky_radiation.**beam_irradiance_horizontal_clear**(B0c, h0)

Computes the clear sky beam irradiance on a horizontal surface

$$B_{hc} = B_{0c} \sin h_0$$

Parameters

B0c [float] beam irradiance normal to the solar beam B_{0c} [W/m2]

h0 [float] solar elevation angle h_0 [degrees]

Returns

Bhc [float] beam irradiance at a horizontal surface B_{hc} [W/m2]

ETLook.clear_sky_radiation.beam_irradiance_normal_clear

ETLook.clear_sky_radiation.**beam_irradiance_normal_clear**(*G0*, *Tl2*, *m*, *rotm*, *h0*) Computes the clear sky beam irradiance normal to the solar beam

$$B_{0c} = G_0 \exp(-0.8662 T_{LK} m \delta_R(m))$$

Parameters

G0 [float] ext rad normal to solar beam G_0 [W/m2]

[19] Šúri, M., Hofierka, J., A new GIS-based Solar Radiation Model and Its Application to Photovoltaic Assessments, Transactions in GIS, 2004, 8(2): 175-190.

Tl2 [float] airmass 2 Linke atmospheric turbidity factor T_{LK} [-]

m [float] relative optical airmass m [-] rotm [float] Rayleigh

optical thickness at airmass m δ_R [-] h0 [float] solar elevation

angle h_0 [degrees]

Returns

B0c [float] beam irradiance normal to the solar beam B_{0c} [W/m2]

ETLook.clear_sky_radiation.day_angle

ETLook.clear_sky_radiation.**day_angle**(*doy*) Computes the day angle. 0 is january 1st, 2π is december 31st.

$$j' = \frac{2\pi j}{365.25}$$

Parameters

doy [float] day of year *j* [-]

Returns

day_angle [float] day angle j' [rad]

ETLook.clear_sky_radiation.declination

ETLook.clear_sky_radiation.**declination**(day_angle)

Computes the solar declination. The solar declination is computed according to Gruter (1984)[20]

$$\delta = \arcsin(0.3978\sin(j' - 1.4 + 0.0355\sin(j' - 0.0489)))$$

Parameters

day_angle [float] day angle j' [rad]

Returns

decl [float] declination δ [rad]

ETLook.clear_sky_radiation.diffuse_irradiance_horizontal_clear

ETLook.clear_sky_radiation.**diffuse_irradiance_horizontal_clear**(*G0*, *Tl2*, *h0*) Computes the clear sky beam irradiance on a horizontal surface

$$D_{hc} = G_0 Tn(T_{LK}) F_d(h_0)$$

For the estimation of the transmission function $Tn(T_{LK})$ the following function is used:

$$Tn(T_{LK}) = -0.015843 + 0.030543 T_{LK} + 0.0003797 T_{LK}^2$$

The solar altitude function $F_d(h_0)$ is evaluated using the expression:

$$F_d(h_0) = A_1 + A_2 \sin h_0 + A_3 sin^2 h_0$$

[20] Gruter, J. W. (ed), 1984, Radiation Nomenclature, Brussels, CEC, Second Solar Energy Programme, Project F, Solar Radiation Data.

with:

$$A'_1 = 0.26463 - 0.061581 T_{LK} + 0.0031408 T_{LK}^2$$

$$A_1 = 0.0022/Tn(T_{LK}) \text{ if } A'_1 Tn(T_{LK}) < 0.0022$$

$$A_1 = A'_1 \text{ if } A'_1 Tn(T_{LK}) \geq 0.0022$$

$$A_2 = 2.04020 + 0.018945 T_{LK} - 0.011161 T_{LK}^2$$

$$A_3 = -1.3025 + 0.039231 T_{LK} + 0.0085079 T_{LK}^2$$

Parameters

G0 [float] ext rad normal to solar beam G_0 [W/m2]

Tl2 [float] Airmass 2 Linke atmospheric turbidity factor T_{LK} [-]

h0 [float] solar elevation angle h_0 [degrees]

Returns

Dhc [float] Diffuse irradiance at a horizontal surface D_{hc} [W/m2]

ETLook.clear_sky_radiation.extraterrestrial_irradiance_normal

ETLook.clear_sky_radiation.**extraterrestrial_irradiance_normal**(I0, ied)

Computes the extraterrestrial irradiance normal to the solar beam

$$G_0 = I_0 \varepsilon$$

Parameters

I0 [float] solar constant I_0 [W m^{-2}] ied [float]

inverse earth sun distance ε [AU^{-1}]

Returns

G0 [float] ext rad normal to solar bea, G_0 [W m^{-2}]

ETLook.clear_sky_radiation.hour_angle

ETLook.clear_sky_radiation.**hour_angle**(solar_time)

Computes the solar hour angle

$$T = \frac{\pi}{12}(t - 12)$$

Parameters solar_time [float] solar_time t

[hours]

Returns ha [float] solar hour angle T

[rad]

ETLook.clear_sky_radiation.inverse_earth_sun_distance

ETLook.clear_sky_radiation.**inverse_earth_sun_distance**(day_angle)

Computes the inverse earth sun distance

$$\varepsilon = 1 + 0.03344\cos(j' - 0.048869)$$

Parameters

day_angle [float] day angle j' [-]

Returns

ied [float] inverse earth sun distance ε [AU]

ETLook.clear_sky_radiation.linke_turbidity

ETLook.clear_sky_radiation.**linke_turbidity**(wv_i, aod550_i, p_air_i, p_air_0_i)

Computes the air mass 2 Linke atmospheric turbidity factor

$$p_{rel} = \frac{p}{p_0}$$

$$T_{LK} = 3.91\tau_{550} \exp\left(0.689 p_{rel}\right) + 0.376 \ln\left(TCWV\right) + \left(2 + 0.54 p_{rel} - 0.34 p_{rel}^2\right)$$

Parameters

wv_i [float] total column atmospheric water vapor $TCWV$ [kg m-2]

aod550_i [float] Aerosol optical depth at 550nm $aod550$ [-]

p_air_i [float] actual instantaneous air pressure p [hPa]

p_air_0_i [float] air pressure at sea level p_0 [-]

Returns

Tl2 [float] Airmass 2 Linke atmospheric turbidity factor T_{LK} [-]

ETLook.clear_sky_radiation.ra_clear_horizontal

ETLook.clear_sky_radiation.**ra_clear_horizontal**(Bhc, Dhc)

Computes the clear sky beam irradiance on a horizontal surface

$$G_{hc} = B_{hc} + D_{hc}$$

Parameters

Bhc [float] beam irradiance at a horizontal surface B_{hc} [W/m2]

Dhc [float] Diffuse irradiance at a horizontal surface D_{hc} [W/m2]

Returns

ra_hor_clear_i [float] Total clear-sky irradiance on a horizontal surface G_{hc} [W/m2]

ETLook.clear_sky_radiation.rayleigh_optical_thickness

ETLook.clear_sky_radiation.**rayleigh_optical_thickness**(m)

Computes the Rayleigh optical thickness at airmass m. It is calculated according to the improved formula by Kasten (1996)[21] if $m > 20$:

$$\delta_R(m) = 1/\left(6.6296 + 1.7513m - 0.1202m^2 + 0.0065m^3 - 0.00013m^4\right)$$

if $m < 20$:

[21] Kasten F. 1996, The Linke turbidity factor based on improved values of the integral Rayleigh optical thickness. Solar Energy 56: 239-44

$$\delta_R(m) = 1/(10.4 + 0.718m)$$

Parameters

m [float] relative optical airmass m [-]

Returns

rotm [float] Rayleigh optical thickness at airmass m δ_R [-]

ETLook.clear_sky_radiation.relative_optical_airmass

ETLook.clear_sky_radiation.**relative_optical_airmass**(p_air_i, p_air_0_i, h0ref)

Computes the relative optical air mass. It is calculated according to Kasten and Young (1989)[22]

$$m = \left(\frac{p}{p_0}\right) / \left(\sin h_0^{ref} + 0.50572\left(h_0^{ref} + 6.07995\right)^{-1.6364}\right)$$

Parameters

p_air_i [float] actual instantaneous air pressure p [hPa]

p_air_0_i [float] air pressure at sea level p_0 [-]

h0ref [float] solar elevation angle corrected for refraction h_0^{ref} [degrees]

Returns

m [float] relative optical airmass m [-]

ETLook.clear_sky_radiation.solar_constant

ETLook.clear_sky_radiation.**solar_constant**()

Returns the solar constant. The solar constant is defined as the flux density of solar radiation at the mean distance from Sun to Earth. The solar constant is estimated to be 1367 W m^{-2}

Returns

I0 [float] solar constant I_0 [W m^{-2}]

ETLook.clear_sky_radiation.solar_elevation_angle

ETLook.clear_sky_radiation.**solar_elevation_angle**(*lat*, *decl*, *ha*) Computes the solar elevation angle

$$h_0 = \arcsin(C_{31}\cos T + C_{33})$$

where

$$C_{31} = \cos\phi\cos\delta\ ;\ C_{33} = \sin\phi\sin\delta$$

[22] Kasten F., and T.A. Young. 1989, Revised optical air mass tables and approximation formula, Applied Optics 28:4735-8.

Parameters

lat [float] latitude ϕ [rad]

decl [float] declination δ [rad]

ha [float] solar hour angle T [rad]

Returns

h0 [float] solar elevation angle h_0 [degrees]

ETLook.clear_sky_radiation.solar_elevation_angle_refracted

ETLook.clear_sky_radiation.**solar_elevation_angle_refracted**(h0)

Computes the solar elevation angle corrected for refraction

$$h_0^{ref} = h_0 + \Delta h_0^{ref}$$

where

$$\Delta h_0^{ref} = 0.61359 \left(0.1594 + 1.123 h_0 + 0.065656 h_0^2\right) / \left(1 + 28.9344 h_0 + 277.3971 h_0^2\right)$$

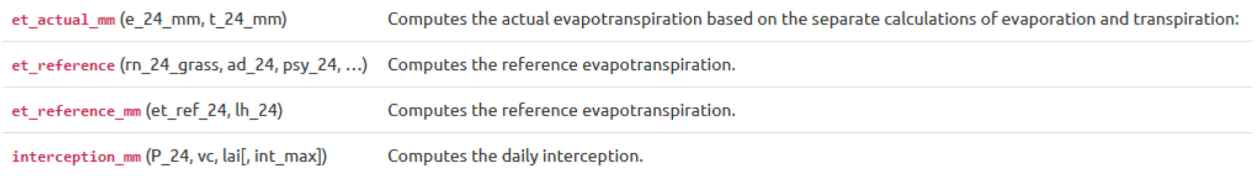
Parameters

h0 [float] solar elevation angle h_0 [degrees]

Returns

h0ref [float] solar elevation angle corrected for refraction h_0^{ref} [degrees]

5.7.2 Evapotranspiration (ETLook.evapotranspiration)



`et_actual_mm` (e_24_mm, t_24_mm)	Computes the actual evapotranspiration based on the separate calculations of evaporation and transpiration:
`et_reference` (rn_24_grass, ad_24, psy_24, ...)	Computes the reference evapotranspiration.
`et_reference_mm` (et_ref_24, lh_24)	Computes the reference evapotranspiration.
`interception_mm` (P_24, vc, lai[, int_max])	Computes the daily interception.

ETLook.evapotranspiration.et_actual_mm

ETLook.evapotranspiration.**et_actual_mm**(*e_24_mm*, *t_24_mm*)

Computes the actual evapotranspiration based on the separate calculations of evaporation and transpiration:

$$ET = E + T$$

Parameters

e_24_mm [float] daily evaporation in mm E [mm d-1]

t_24_mm [float] daily transpiration in mm T [mm d-1]

Returns

et_24_mm [float] daily evapotranspiration in mm ET [mm d-1]

5.7.3 Vegetation Cover (ETLook.leaf)

The leaf module contains all functions related to estimating vegetation cover. These functions only work on an instantaneous basis.

`vegetation_cover` (ndvi[, nd_min, nd_max, vc_pow])	Computes vegetation cover based on NDVI
`leaf_area_index` (vc[, vc_min, vc_max, lai_pow])	Computes leaf area index based on vegetation cover.
`effective_leaf_area_index` (lai)	Computes effective leaf area index, this describes the leaf area which actively participates in transpiration.

5.7.4 Meteorology (ETLook.meteo)

The meteo module contains all functions related to meteorological variables. All meteorological functions can be calculated on a daily or instantaneous basis. Base functions are available also.
The daily functions have a 'daily' extension, instantaneous functions have a 'inst' extension

`air_density` (ad_dry, ad_moist)	Computes air density ρ in [kg m-3]
`air_density_daily` (ad_dry_24, ad_moist_24)	Like `air_density()` but as a daily average
`air_density_inst` (ad_dry_i, ad_moist_i)	Like `air_density()` but as a instantaneous value
`air_pressure` (z[, p_air_0])	Computes air pressure P at a certain elevation derived from the air pressure at sea level P_0.
`air_pressure_daily` (z[, p_air_0_24])	Like `air_pressure()` but as a daily average
`air_temperature_kelvin` (t_air)	Converts air temperature from Celcius to Kelvin, where 0 degrees Celcius is equal to 273.
`air_temperature_kelvin_daily` (t_air_24)	Like `air_temperature_kelvin()` but as a daily average
`air_temperature_kelvin_inst` (t_air_i)	Like `air_temperature_kelvin()` but as an instantaneous value
`disaggregate_air_temperature` (t_air_coarse, …)	Disaggregates GEOS or MERRA or another coarse scale air temperature using two digital elevation models.
`disaggregate_air_temperature_daily` (…[, lapse])	Like `disaggregate_air_temperature()` but as a daily average
`disaggregate_air_temperature_inst` (…[, lapse])	Like `disaggregate_air_temperature()` but as a instantaneous value
`disaggregate_dew_point_temperature_inst` (…)	Disaggregates geos dew point temperature using lapse rate and difference between smoothed coarse scale DEM and fine scale DEM

`dry_air_density` (p_air, vp, t_air_k)	Computes dry air density ρ_d in [kg m-3]
`dry_air_density_daily` (p_air_24, vp_24, …)	Like `dry_air_density()` but as a daily average
`dry_air_density_inst` (p_air_i, vp_i, t_air_k_i)	Like `dry_air_density()` but as an instantaneous value
`latent_heat` (t_air)	Computes latent heat of evaporation λ [J kg-1], describing the amount of energy needed to evaporate one kg of water at constant pressure and temperature.
`latent_heat_daily` (t_air_24)	Like `latent_heat()` but as a daily average
`moist_air_density` (vp, t_air_k)	Computes moist air density ρ_s in [kg m-3]
`moist_air_density_daily` (vp_24, t_air_k_24)	Like `moist_air_density()` but as a daily average
`moist_air_density_inst` (vp_i, t_air_k_i)	Like `moist_air_density()` but as an instantaneous value
`psychrometric_constant` (p_air, lh)	Computes the psychrometric constant γ [mbar K-1] which relates the partial pressure of water in air to the air temperature
`psychrometric_constant_daily` (p_air_24, lh_24)	Like `psychrometric_constant()` but as a daily average
`saturated_vapour_pressure` (t_air)	Computes saturated vapour pressure e_s [mbar], it provides the vapour pressure when the air is fully saturated with water.
`saturated_vapour_pressure_daily` (t_air_24)	Like `saturated_vapour_pressure()` but as a daily average
`slope_saturated_vapour_pressure` (t_air)	Computes the rate of change of vapour pressure Δ in [mbar K-1] for a given air temperature T_a.
`slope_saturated_vapour_pressure_daily` (t_air_24)	Like `slope_saturated_vapour_pressure()` but as a daily average
`vapour_pressure_deficit` (svp, vp)	Computes the vapour pressure deficit Δ_e in [mbar]
`vapour_pressure_deficit_daily` (svp_24, vp_24)	Like `vapour_pressure_deficit()` but as a daily average
`vapour_pressure_from_specific_humidity` (qv, p_air)	Computes the vapour pressure e_a in [mbar] using specific humidity and surface pressure
`vapour_pressure_from_specific_humidity_daily` (…)	Like `vapour_pressure_from_specific_humidity()` but as a daily average
`vapour_pressure_from_specific_humidity_inst` (…)	Like `vapour_pressure_from_specific_humidity()` but as an instantaneous value
`wet_bulb_temperature_kelvin_inst` (t_wet_i)	Converts wet bulb temperature from Celcius to Kelvin, where 0 degrees Celcius is equal to 273.
`wind_speed_blending_height` (u[, z_obs, z_b])	Computes the wind speed at blending height u_b [m/s] using the logarithmic wind profile
`wind_speed_blending_height_daily` (u_24[, …])	Like `wind_speed_blending_height()` but as a daily average

ETLook.meteo.air_density

ETLook.meteo.**air_density**(*ad_dry*, *ad_moist*)

Computes air density ρ in [kg m-3]

$$\rho = \rho_s + \rho_d$$

Parameters

ad_dry [float] dry air density ρ_d [kg m-3]

ad_moist [float] moist air density ρ_s [kg m-3]

Returns

ad [float] air density ρ [kg m-3]

Examples

```
>>> from ETLook import meteo
>>> ad_moist = meteo.moist_air_density(vp=17.5, t_air_k = 293.15) >>> ad_dry = meteo.dry_air_density(p_air=900, vp=17.5,
t_air_k=293.15)
>>> meteo.air_density(ad_dry=ad_dry, ad_moist=ad_moist)
1.0618706622660472
```

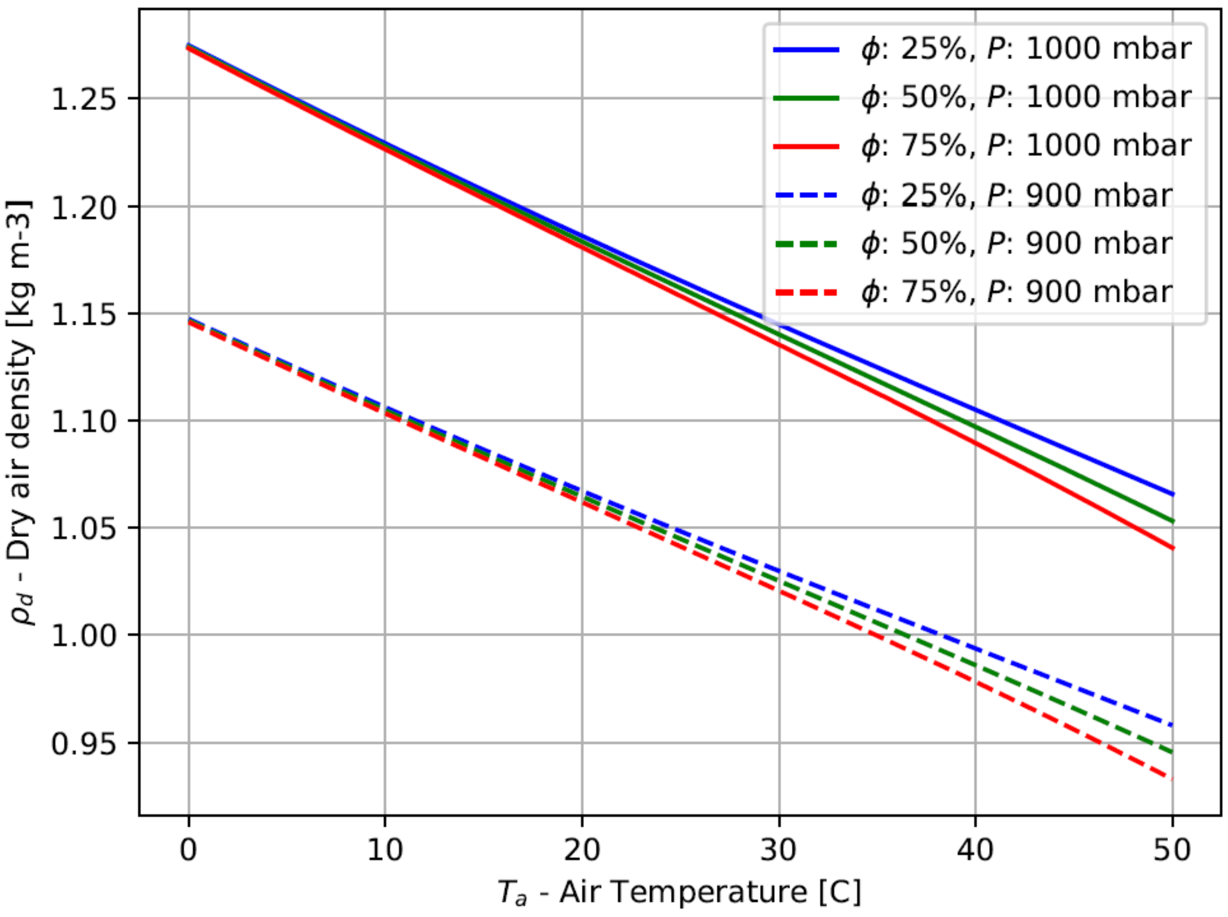
ETLook.meteo.air_pressure

ETLook.meteo.**air_pressure**(*z*, *p_air_0=1013.25*)

Computes air pressure P at a certain elevation derived from the air pressure at sea level P_0. Air pressure decreases with increasing elevation.

$$P = P_0 \left(\frac{T_{ref,0,K} - \alpha_1 (z - z_0)}{T_{ref,0,K}} \right)^{\frac{g}{\alpha_1 R}}$$

where the following constants are used



- P_0 = air pressure [mbar] at sea level z_0 = 1013.25 mbar
- $T_{ref,0,K}$ = reference temperature [K] at sea level z_0 = 293.15 K
- g = gravitational acceleration = 9.807 [m/s2]
- R = specific gas constant = 287.0 [J kg-1 K-1]
- α_1 = constant lapse rate for moist air = 0.0065 [K m-1]

Parameters

z [float] elevation z [m]

p_air_0 [float] air pressure at sea level P_0

[mbar]

Returns

p_air [float] air pressure P [mbar]

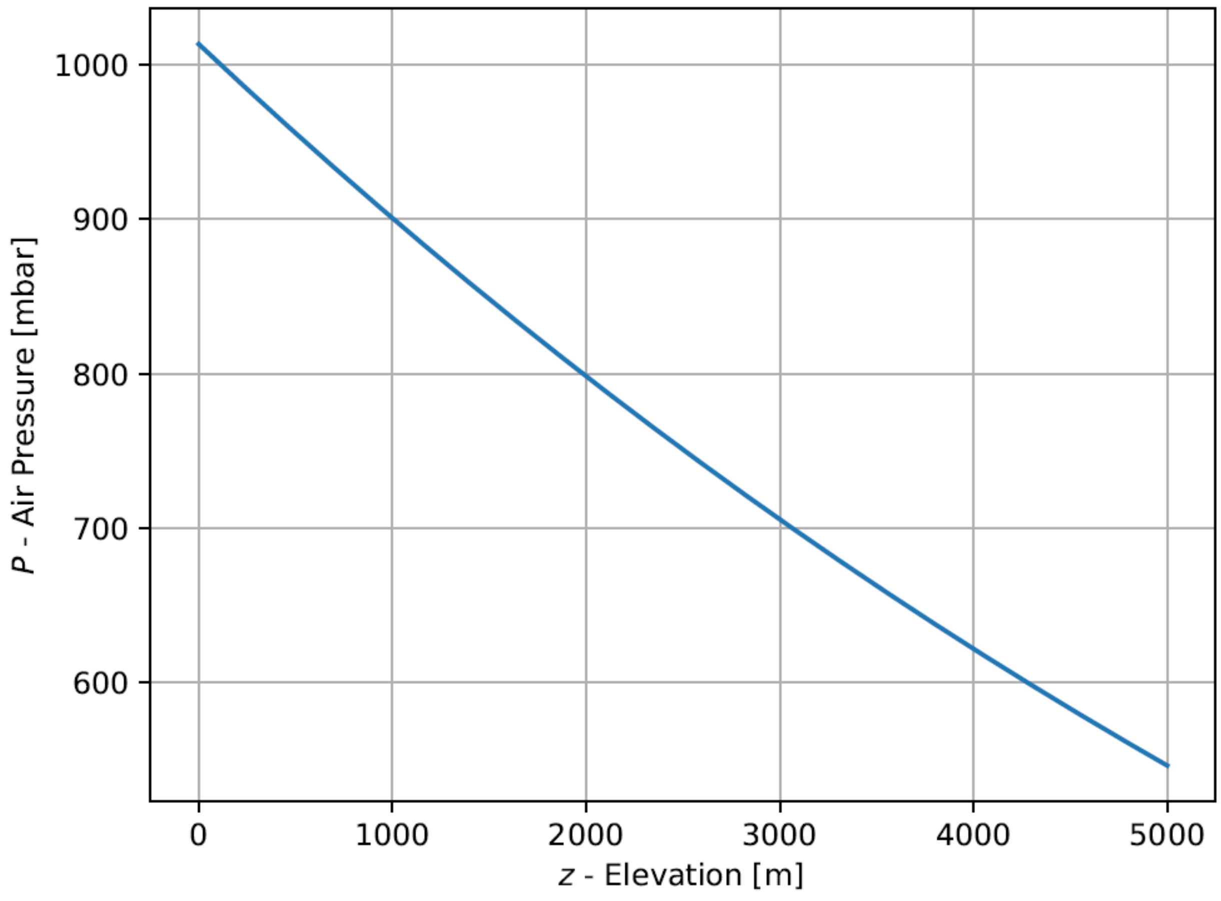
Examples

```
>>> from ETLook import meteo
>>> meteo.air_pressure(z=1000)
900.5832172948869
```

ETLook.meteo.air_temperature_kelvin

ETLook.meteo.**air_temperature_kelvin**(t_air)

Converts air temperature from Celcius to Kelvin, where 0 degrees Celcius is equal to 273.15 degrees Kelvin



Parameters

t_air [float] air temperature T_a [C]

Returns

t_air_k [float] air temperature T_a [K]

Examples

```
>>> from ETLook import meteo
>>> meteo.air_temperature_kelvin(12.5)
285.65
```

ETLook.meteo.disaggregate_air_temperature

ETLook.meteo.**disaggregate_air_temperature**(*t_air_coarse*, *z*, *z_coarse*, *lapse=-0.006*)

Disaggregates GEOS or MERRA or another coarse scale air temperature using two digital elevation models. One DEM for the target resolution, another DEM smoothed from the original air temperature resolution to the target resolution.

$$T_a = T_{a,c} + (z - z_c)L_T - T_{K,0}$$

where the following constant is used

- $T_{K,0}$ = 273.15 K is equal to 0 degrees Celsius

Parameters
t_air_coarse [float] air temperature at coarse resolution Ta,c [K]
z [float] elevation z [m]
z_coarse [float] elevation at coarse resolution zc [m]
lapse [float] lapse rate LT [K m-1]
Returns

t_air [float] air temperature T_a [C]

Notes

The input air temperature is specified in Kelvin. The output air temperature is specified in C.

Examples

```
>>> from ETLook import meteo
>>> meteo.disaggregate_air_temperature(24.5+273.15, 10, 5)
24.47
```

ETLook.meteo.disaggregate_air_temperature_daily

ETLook.meteo.**disaggregate_air_temperature_daily**(*t_air_24_coarse*, *z*, *z_coarse*, *lapse=-0.006*)

Like *disaggregate_air_temperature()* but as a daily average

Parameters

t_air_24_coarse [float] daily air temperature at coarse resolution $T_{a,24,c}$ [K]

z [float] elevation z [m]

z_coarse [float] elevation at coarse resolution z_c [m]

lapse [float] lapse rate L [K m-1]

Returns

t_air_24 [float] daily air temperature $T_{a,24}$ [C]

Notes

The input air temperature is specified in Kelvin. The output air temperature is specified in C.

ETLook.meteo.disaggregate_air_temperature_inst

ETLook.meteo.**disaggregate_air_temperature_inst**(t_air_i_coarse, z, z_coarse, lapse=-

0.006)

Like *disaggregate_air_temperature()* but as a instantaneous value

Parameters

t_air_i_coarse [float] instantaneous air temperature at coarse resolution

$T_{a,i,c}$ [K]

z [float] elevation z [m]

z_coarse [float] elevation at coarse resolution z_c [m]

lapse [float] lapse rate L [K m-1]

Returns

t_air_i [float] instantaneous air temperature $T_{a,i}$ [C]

Notes

The input air temperature is specified in Kelvin. The output air temperature is specified in C.

ETLook.meteo.disaggregate_dew_point_temperature_inst

ETLook.meteo.**disaggregate_dew_point_temperature_inst**(t_dew_coarse_i, z,

z_coarse, *lapse_dew=-*
0.002)

Disaggregates geos dew point temperature using lapse rate and difference between smoothed coarse scale DEM and fine scale DEM

Parameters

t_dew_coarse_i [float] coarse instantaneous dew point temperature

$T_{dew,coarse}$ [C]

z [float] elevation z [m]

z_coarse [float] smoothed elevation at coarse resolution z [m]

lapse_dew [float] lapse rate L [K m-1]

Returns

t_dew_i [float] instantaneous dew point temperature $T_{dew,i}$

[C]

ETLook.meteo.dry_air_density

ETLook.meteo.**dry_air_density**(*p_air*, *vp*, *t_air_k*) Computes
dry air density ρ_d in [kg m-3]

$$\rho_d = \frac{P - e_a}{\Re T_{a,K}}$$

where the following constants are used

- $\Re$ = gas constant for dry air = 2.87 mbar K-1 m3 kg-1

Parameters

p_air [float] air pressure P [mbar]

vp [float] vapour pressure e_a [mbar]

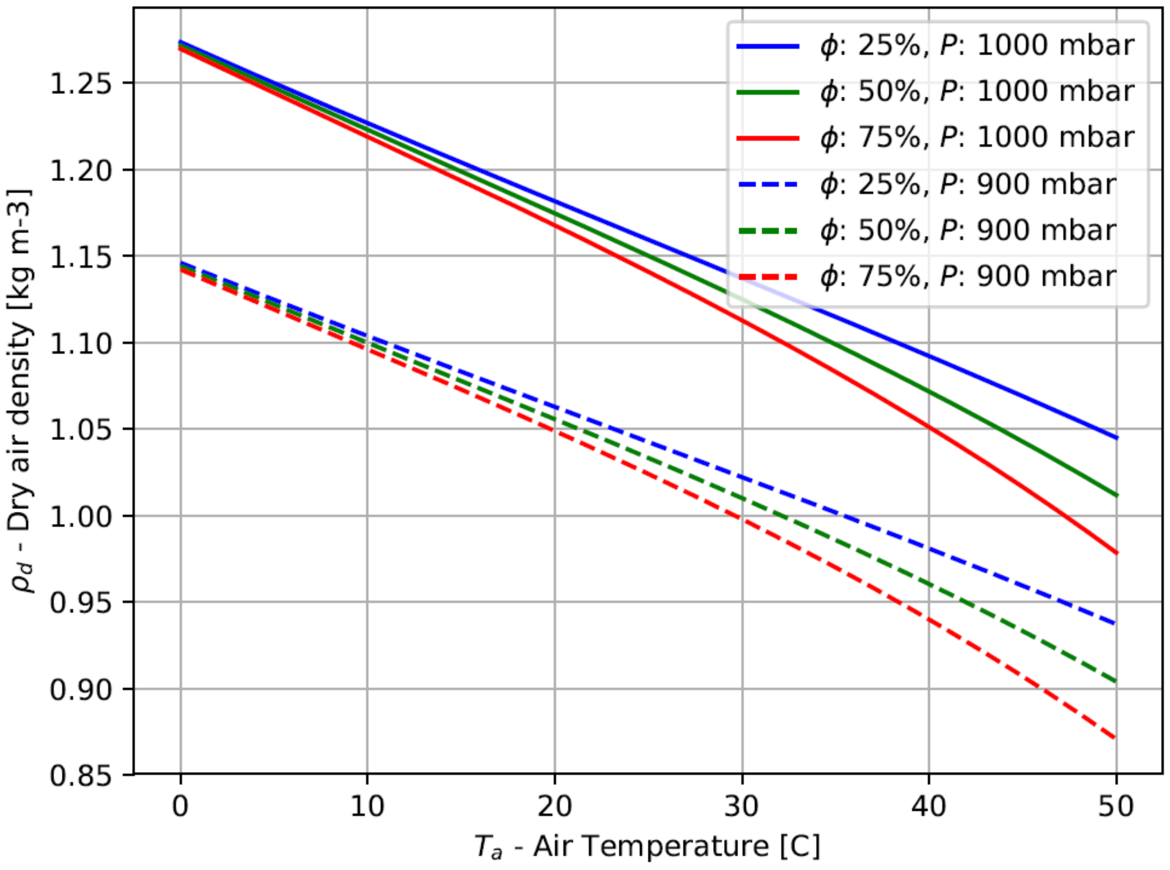
t_air_k [float] daily air temperature T_a [K]

Returns

ad_dry [float] dry air density ρ_d [kg m-3]

Examples

```
>>> from ETLook import meteo
>>> meteo.dry_air_density(p_air=900, vp=17.5, t_air_k=293.15)
1.0489213344656534
```



ETLook.meteo.latent_heat

ETLook.meteo.**latent_heat**(t_air)

Computes latent heat of evaporation λ [J kg-1], describing the amount of energy needed to evaporate one kg of water at constant pressure and temperature. At higher temperatures less energy will be required than at lower temperatures.

$$\lambda = (\lambda_0 + \Delta_\lambda T_a)$$

where the following constants are used

- λ_0 = latent heat of evaporation at 0 C = 2501000 [J kg-1]
- Δ_λ = rate of change of latent heat with respect to temperature = -2361 [J Kg-1 C-1]

Parameters

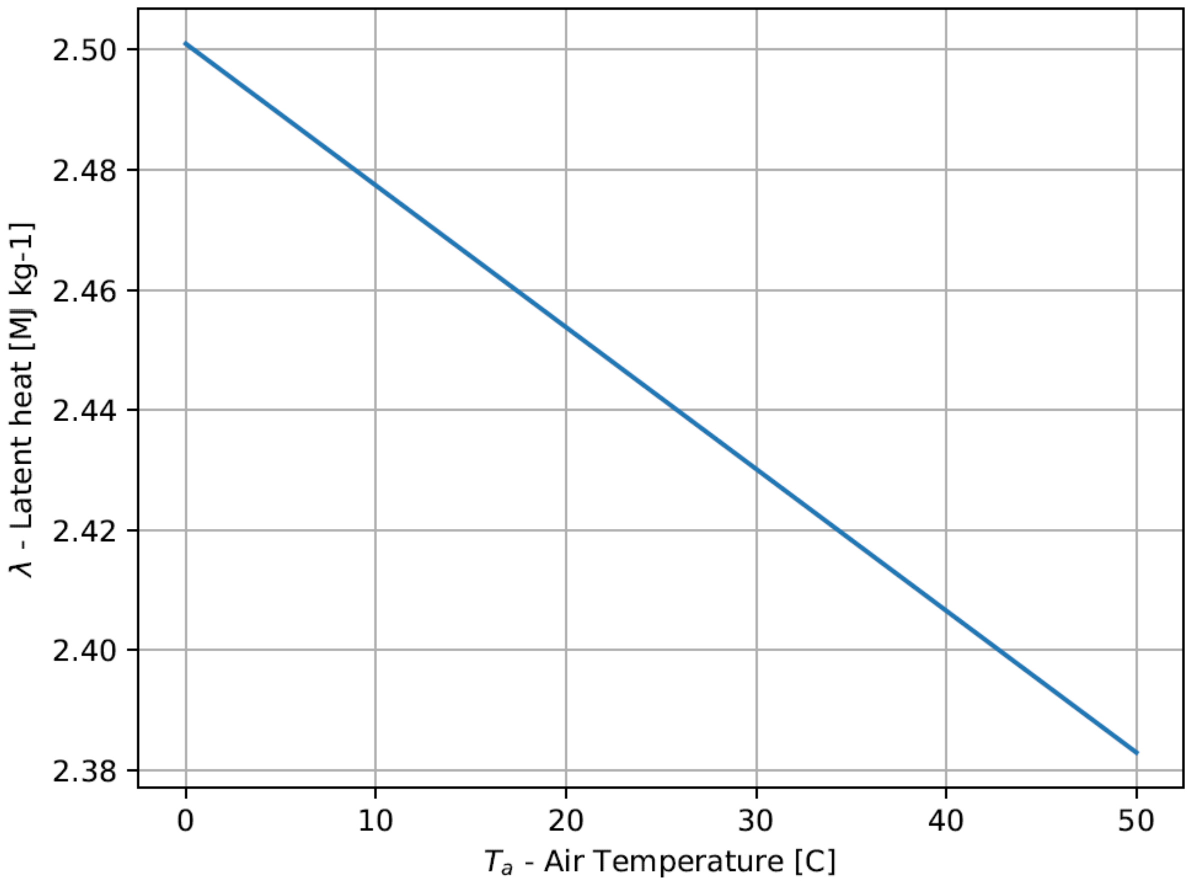
t_air [float] air temperature T_a [C]

Returns

lh [float] latent heat of evaporation λ [J/kg]

Examples

```
>>> from ETLook import meteo
>>> meteo.latent_heat(20)
2453780.0
```



ETLook.meteo.moist_air_density

ETLook.meteo.**moist_air_density**(*vp*, *t_air_k*) Computes moist air density ρ_s in [kg m-3]

$$\rho_s = \frac{e_a}{R_v T_{a,K}}$$

where the following constants are used

- R_v = gas constant for moist air = 4.61 mbar K-1 m3 kg-1

Parameters

vp [float] vapour pressure e_a [mbar] t_air_k

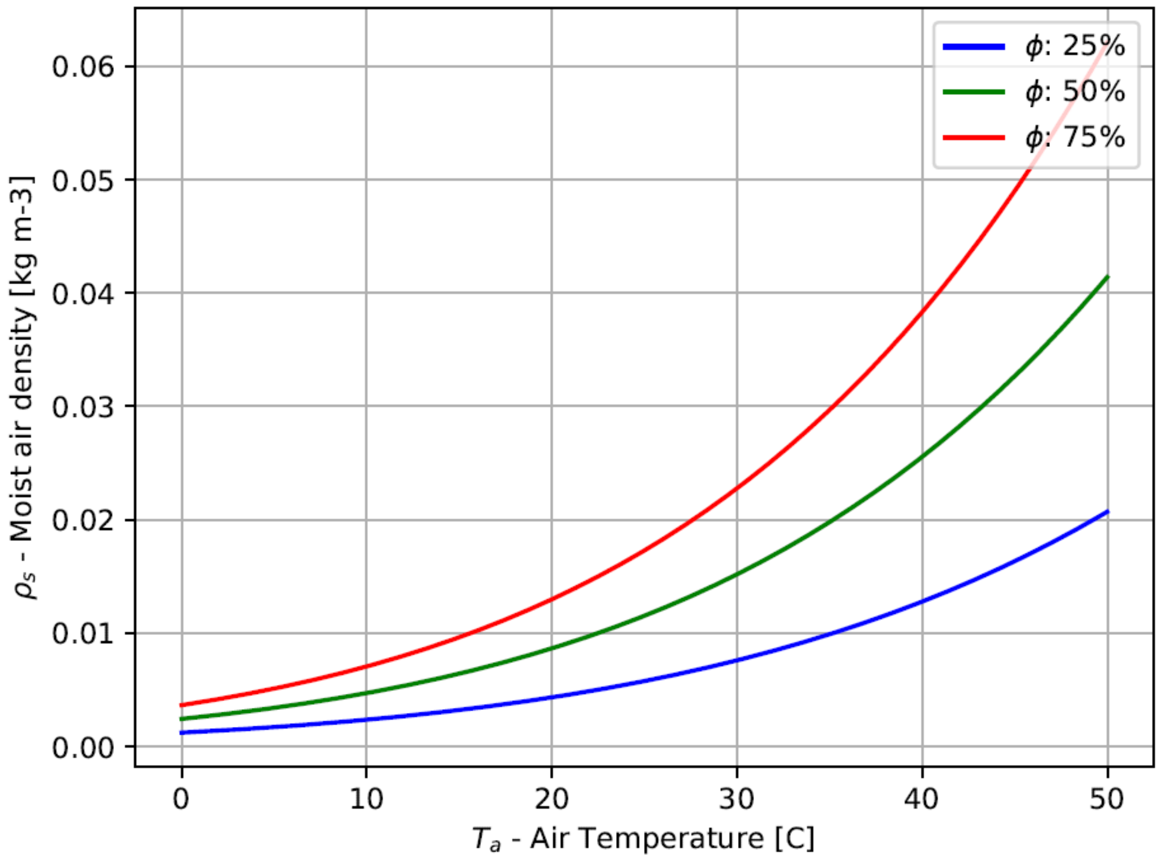
[float] air temperature $T_{a,K}$ [K]

Returns

ad_moist [float] moist air density ρ_s [kg m-3]

Examples

```
>>> from ETLook import meteo
>>> meteo.moist_air_density(vp=17.5, t_air_k = 293.15)
0.012949327800393881
```



ETLook.meteo.psychrometric_constant

ETLook.meteo.**psychrometric_constant**(p_air, lh)

Computes the psychrometric constant γ [mbar K-1] which relates the partial pressure of water in air to the air temperature

$$\gamma = \frac{Pc_p}{\varepsilon\lambda}$$

where the following constants are used

- c_p = specific heat for dry air = 1004 [J Kg-1 K-1]
- ε = ratio of molecular weight of water to dry air

  = 0.622 [-]

Parameters

p_air [float] air pressure P [mbar]

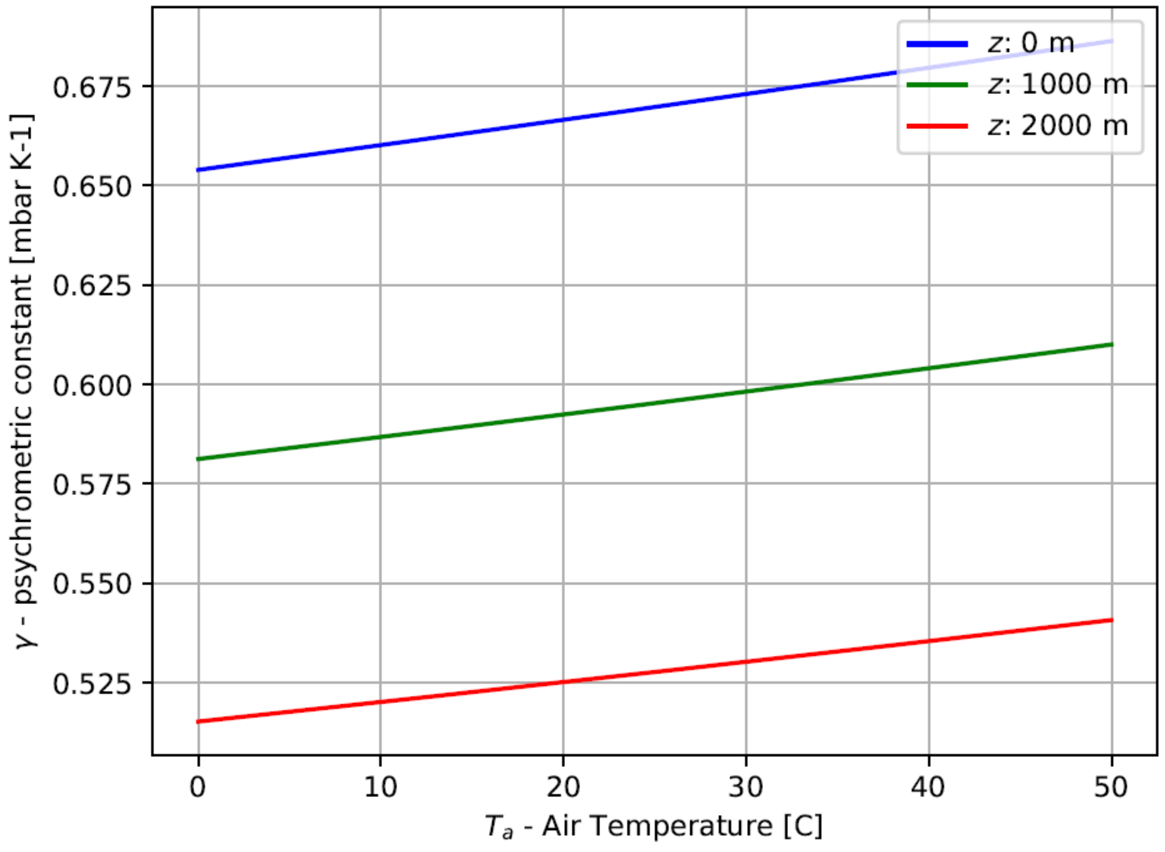
lh [float] latent heat of evaporation λ [J/kg]

Returns

psy [float] psychrometric constant γ [mbar K-1]

Examples

```
>>> from ETLook import meteo
>>> meteo.psychrometric_constant(p_air = 1003.0, lh = 2500000.0)
0.6475961414790997
>>> meteo.psychrometric_constant(1003.0, 2500000.0)
0.6475961414790997
```



ETLook.meteo.saturated_vapour_pressure

ETLook.meteo.**saturated_vapour_pressure**(t_air)

Computes saturated vapour pressure e_s [mbar], it provides the vapour pressure when the air is fully saturated with water. It is related to air temperature T_a [C]:

$$e_s = 6.108 \exp\left[\frac{17.27T_a}{T_a + 237.3}\right]$$

Parameters

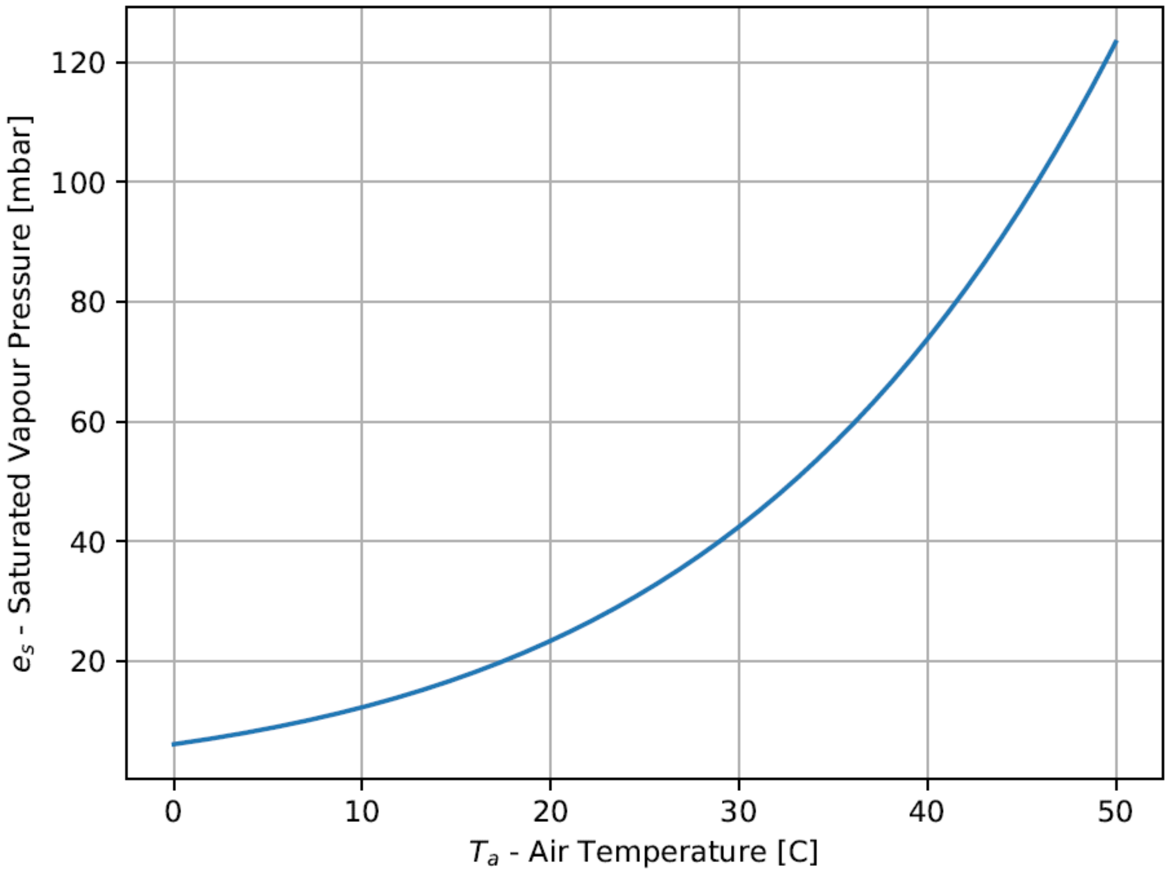
t_air [float] air temperature T_a [C]

Returns

svp [float] saturated vapour pressure e_s [mbar]

Examples

```
>>> from ETLook import meteo
>>> meteo.saturated_vapour_pressure(20)
23.382812709274457
```



ETLook.meteo.slope_saturated_vapour_pressure

ETLook.meteo.**slope_saturated_vapour_pressure**(t_air)

Computes the rate of change of vapour pressure Δ in [mbar K-1] for a given air temperature T_a. It is a function of the air temperature T_a and the saturated vapour pressure e_s [mbar] which in itself is a function of T_a.

$$\Delta = \frac{4098e_s}{(237.3 + T_a)^2}$$

for e_s see *saturated_vapour_pressure()*

Parameters

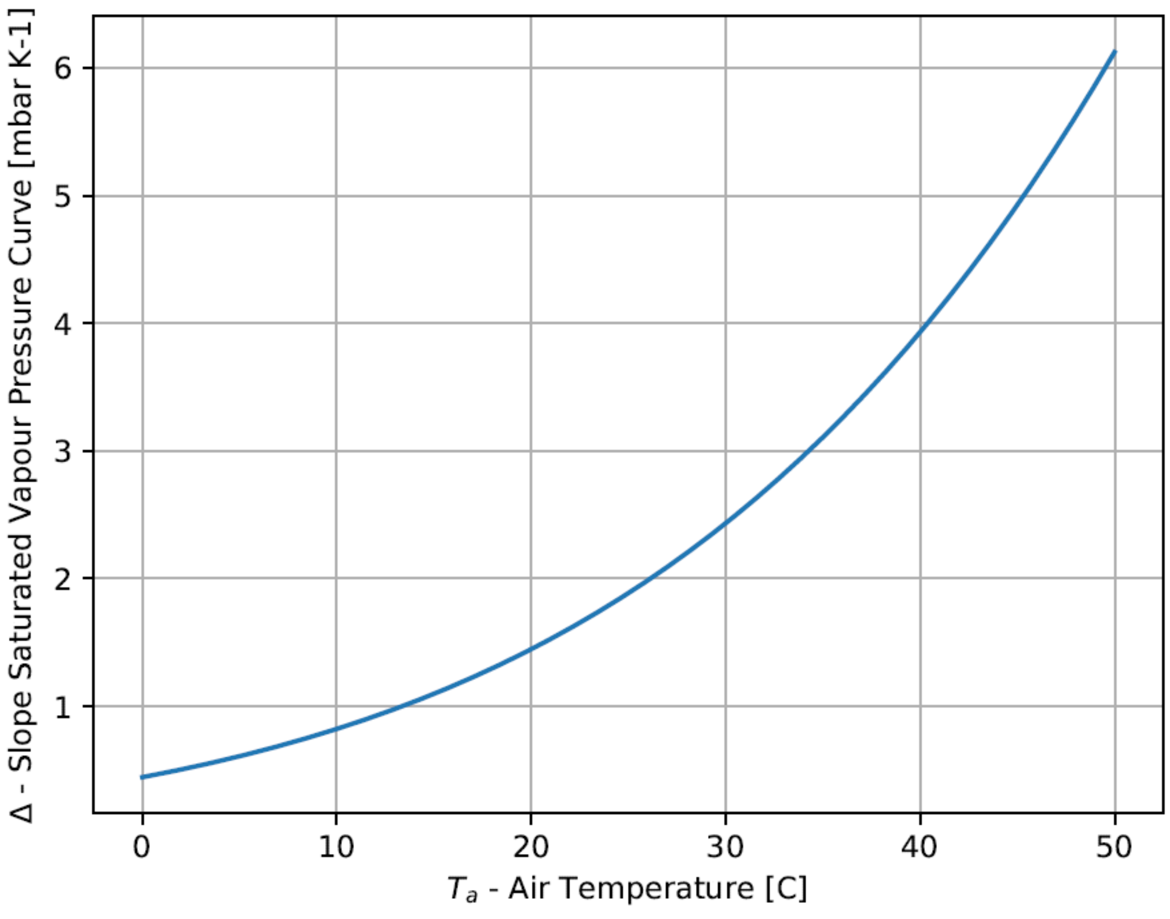
t_air [float] air temperature T_a [C]

Returns

ssvp [float] slope of saturated vapour pressure curve Δ [mbar K-1]

Examples

```
>>> from ETLook import meteo
>>> meteo.slope_saturated_vapour_pressure(20)
1.447401881124136
```



ETLook.meteo.vapour_pressure_deficit

ETLook.meteo.**vapour_pressure_deficit**(*svp*, *vp*) Computes the vapour pressure deficit Δ_e in [mbar]

$$\Delta_e = e_s - e_a$$

Parameters

svp [float] saturated vapour pressure e_s [mbar]

vp [float] actual vapour pressure e_a [mbar]

Returns

vpd [float] vapour pressure deficit Δ_e [mbar]

Examples

```
>>> from ETLook import meteo
>>> meteo.vapour_pressure_deficit(12.5, 5.4)
7.1
>>> meteo.vapour_pressure_deficit(vp=5.4, svp=12.3)
6.9
```

ETLook.meteo.wind_speed_blending_height

ETLook.meteo.**wind_speed_blending_height**(u, z_obs=2, z_b=100)

Computes the wind speed at blending height u_b [m/s] using the logarithmic wind profile

$$u_b = \frac{u_{obs} \ln\left(\frac{z_b}{z_{0,m}}\right)}{\ln\left(\frac{z_{obs}}{z_{0,m}}\right)}$$

Parameters

u [float] wind speed at observation height u_{obs} [m/s]

z_obs [float] observation height of wind speed z_{obs} [m]

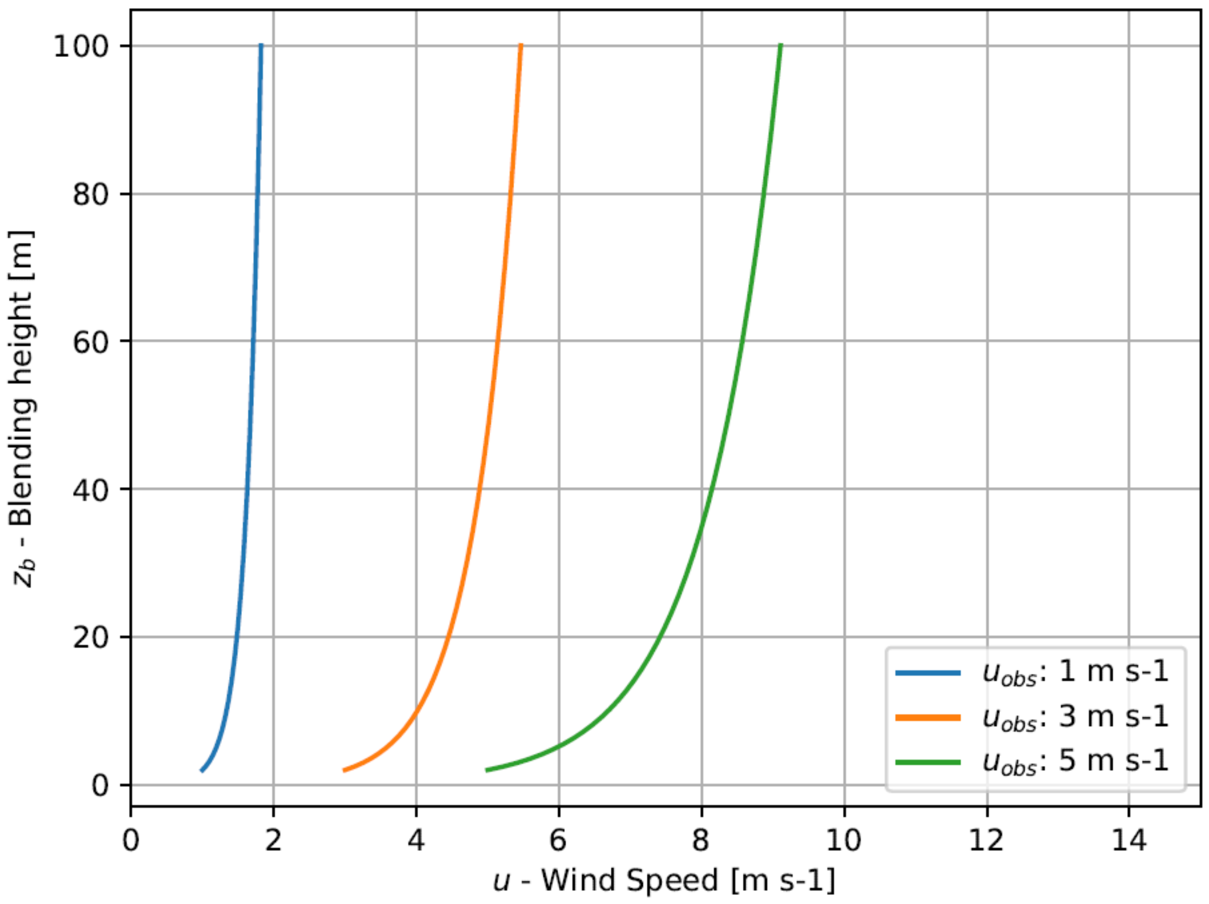
z_b [float] blending height z_b [m]

Returns

u_b [float] wind speed at blending height u_b [m/s]

Examples

```
>>> from ETLook import meteo
>>> meteo.wind_speed_blending_height(u=3.0, z_obs=2, z_b=100)
5.4646162953650572
```



5.7.5 Net Available Energy (ETLook.radiation)

`bare_soil_heat_flux` (doy, dd, stc, …)	Computes the bare soil heat flux
`damping_depth` (stc, vhc)	Computes the damping depth
`interception_wm2` (int_mm, lh_24)	Computes the energy equivalent for the interception in Wm-2 if it is provide in mm/day
`longwave_radiation_fao` (t_air_k_24, vp_24, …)	Computes the net longwave radiation according to the FAO 56 manual.
`longwave_radiation_fao_etref` (t_air_k_24, …)	Computes the net longwave radiation according to the FAO 56 manual.
`net_radiation` (r0, ra_24, l_net, int_wm2)	Computes the net radiation
`net_radiation_canopy` (rn_24, sf_soil)	Computes the net radiation for the canopy
`net_radiation_grass` (ra_24, l_net[, r0_grass])	Computes the net radiation for reference grass
`net_radiation_soil` (rn_24, sf_soil)	Computes the net radiation for the soil
`soil_fraction` (lai)	Computes the effect of the vegetation has in separating the net radiation into a soil and canopy component.
`soil_heat_flux` (g0_bs, sf_soil, land_mask, …)	Computes the soil heat flux
`soil_thermal_conductivity` (se_top)	Computes the soil thermal conductivity
`volumetric_heat_capacity` ([se_top, porosity])	Computes the volumetric heat capacity of the soil

5.7.6 Roughness (ETLook.roughness)

The roughness module contains all functions related to surface roughness

roughness_length (lai, z_oro, z_obst, z_obst_max)	Computes the surface roughness length.
obstacle_height (ndvi, z_obst_max[, …])	Computes the obstacle height.
displacement_height (lai, z_obst[, land_mask, c1])	Computes the displacement height.

5.7.7 Solar radiation (ETLook.solar_radiation)

aspect_rad (aspect_deg)	Converts aspect from degrees to radians.
cosine_solar_zenith_angle (ha, decl, lat[, …])	computes the cosine of the solar zenith angle [-]
daily_solar_radiation_flat (ra_24_toa_flat, …)	Computes the daily solar radiation at the earth's surface
daily_solar_radiation_toa (sc, decl, iesd, lat)	Computes the daily solar radiation at the top of the atmosphere.
daily_solar_radiation_toa_flat (decl, iesd, …)	Computes the daily solar radiation at the top of the atmosphere for a flat surface.
daily_total_solar_radiation (ra_24_toa, …)	Computes the daily solar radiation at the earth's surface taken diffuse and direct solar radiation into account
declination (doy)	Computes the solar declination which is the angular height of the sun above the astronomical equatorial plane in radians
diffusion_index (trans_24[, diffusion_slope, …])	Computes the diffusion index, the ratio between diffuse and direct solar radiation.
hour_angle (sc, dtime[, lon])	Computes the hour angle which is zero at noon and -pi at 0:00 am and pi at 12:00 pm
inst_solar_radiation_toa (csza, iesd)	Computes the instantaneous solar radiation at the top of the atmosphere [Wm-2]
inverse_earth_sun_distance (doy)	Computes the inverse earth sun distance (iesd) in Angstrom Unit where 1 AU is 1.
latitude_rad (lat_deg)	Converts latitude from degrees to radians.
seasonal_correction (doy)	Computes the seasonal correction for solar time in hours
slope_rad (slope_deg)	Converts slope from degrees to radians.
sunset_hour_angle (lat, decl)	Computes the sunset hour angle

ETLook.solar_radiation.aspect_rad

ETLook.solar_radiation.**aspect_rad**(*aspect_deg*) Converts aspect from degrees to radians.

Parameters aspect_deg : float

aspect in degrees s [deg]

Returns aspect : float

aspect (0 is north; pi is south) α [rad]

ETLook.solar_radiation.cosine_solar_zenith_angle

ETLook.solar_radiation.**cosine_solar_zenith_angle**(*ha*, *decl*, *lat*, *slope=0*, *aspect=0*) computes the cosine of the solar zenith angle [-]

$$\varphi = \begin{array}{c} \sin\delta\,\sin\lambda\cos\Delta - \sin\delta \\ \cos\lambda\sin\Delta + \\ \cos\delta\,\cos\lambda\cos\Delta\cos(\omega) + \\ \cos\delta\,\sin\lambda\sin\Delta\sin\alpha\cos(\omega) + \cos\delta \\ \sin\Delta\sin\alpha\sin(\omega) \end{array}$$

Parameters ha : float

hour angle ω [rad]

decl : float

declination δ [rad]

lat : float

latitude λ [rad]

slope : float

slope Δ [rad]

aspect : float

aspect (0 is north; pi is south) α [rad]

Returns csza : float

cosine solar zenith angle φ [-]

Examples

```
>>> import ETLook.solar_radiation as solrad
>>> sc = solrad.seasonal_correction(1)
>>> ha = solrad.hour_angle(sc, dtime=12)
>>> solrad.cosine_solar_zenith_angle(ha, decl=solrad.declination(1), lat=0)
0.92055394167363314
```

ETLook.solar_radiation.daily_solar_radiation_flat

ETLook.solar_radiation.**daily_solar_radiation_flat**(ra_24_toa_flat, trans_24)

Computes the daily solar radiation at the earth's surface

$$S_{\downarrow} = \tau S_{toa}$$

Parameters ra_24_toa_flat : float

daily solar radiation at the top of atmosphere for a flat surface S_{toa} [Wm-2]

trans_24 : float

daily atmospheric transmissivity τ [-]

Returns ra_24 : float

daily solar radiation for a flat surface $S^{\downarrow}$ [Wm-2]

ETLook.solar_radiation.daily_solar_radiation_toa_flat

ETLook.solar_radiation.**daily_solar_radiation_toa_flat**(*decl*, *iesd*, *lat*, *ws*) Computes the daily solar radiation at the top of the atmosphere for a flat surface.

$$S_{toa,f} = \frac{S_{sun}}{\pi} d_{inv,r} * (w_s \sin(\lambda)) \sin(\delta) + \cos(\lambda)) \cos(\delta) \sin(w_s)$$

Parameters decl : float

solar declination δ [rad]

iesd : float

inverse earth sun distance $d_{inv,r}$ [AU]

lat : float

latitude λ [rad]

ws : float

sunset hour angle w_s [rad]

Returns ra_24_toa_flat : float

daily solar radiation at the top of atmosphere for a flat surface $S_{toa,f}$ [Wm-2]

ETLook.solar_radiation.daily_total_solar_radiation

ETLook.solar_radiation.**daily_total_solar_radiation**(*ra_24_toa*, *ra_24_toa_flat*, *diffusion_index*, *trans_24*)

Computes the daily solar radiation at the earth's surface taken diffuse and direct solar radiation into account

$$S\downarrow = I_{diff}\tau S_{toa,f} + (1 - I_{diff})\tau S_{toa}$$

Parameters ra_24_toa : float

daily solar radiation at the top of atmosphere S_{toa} [Wm-2]

ra_24_toa_flat : float daily solar radiation at the top of

atmosphere for a flat surface $S_{toa,f}$ [Wm-2]

diffusion_index : float

diffusion_index I_{diff} [-] trans_24 : float

daily atmospheric transmissivity τ [-]

Returns ra_24 : float

daily solar radiation $S\downarrow$ [Wm-2]

ETLook.solar_radiation.declination

ETLook.solar_radiation.**declination**(doy)

Computes the solar declination which is the angular height of the sun above the astronomical equatorial plane in radians

$$\delta = 0.409 \sin\left(\frac{2\pi J}{365} - 1.39\right)$$

Parameters doy : float julian day of

the year J [-]

Returns decl : float declination δ

[rad]

Examples

```
>>> import ETLook.solar_radiation as solrad
>>> solrad.declination(180)
0.40512512455439242
```

ETLook.solar_radiation.diffusion_index

ETLook.solar_radiation.**diffusion_index**(*trans_24*, *diffusion_slope=-1.33*, *diffusion_intercept=1.15*)

Computes the diffusion index, the ratio between diffuse and direct solar radiation. The results are clipped between 0 and 1.

$$I_{diff} = a_{diff} + b_{diff}\tau$$

Parameters trans_24 : float

daily atmospheric transmissivity τ [-]

diffusion_slope : float

slope of diffusion index vs transmissivity relationship b_{diff} [-]

diffusion_intercept : float

intercept of diffusion index vs transmissivity relationship a_{diff} [-]

Returns diffusion_index : float

diffusion_index I_{diff} [-]

ETLook.solar_radiation.hour_angle

ETLook.solar_radiation.**hour_angle**(sc, dtime, lon=0)

Computes the hour angle which is zero at noon and -pi at 0:00 am and pi at 12:00 pm

$$\omega = \left(\frac{\pi}{12}\right)(t + s_c - 12)$$

Parameters sc : float
seasonal correction sc [hours]
dtime : float
decimal time t [hours]
lon : float longitude ϕ [rad]
Returns ha : float

hour_angle ω [rad]

Examples

```
>>> import ETLook.solar_radiation as solrad >>> solrad.hour_angle(sc=solrad.seasonal_correction(75), dtime=11.4)
-0.19793970172084141
```

ETLook.solar_radiation.inst_solar_radiation_toa

ETLook.solar_radiation.**inst_solar_radiation_toa**(csza, iesd)

Computes the instantaneous solar radiation at the top of the atmosphere [Wm-2]

$$S_{toa}^{i} = S_{sun} d_r \phi$$

Parameters csza : float

cosine solar zenith angle φ [-]

iesd : float

inverse earth sun distance d_r [AU]

Returns ra_i_toa : float

instantaneous solar radiation at top of atmosphere $S_{toa}{}^{i}$ [Wm-2]

Examples

```
>>> import ETLook.solar_radiation as solrad
>>> doy = 1
>>> sc = solrad.seasonal_correction(doy)
>>> ha = solrad.hour_angle(sc, dtime=12)
>>> decl = solrad.declination(doy)
>>> csza = solrad.cosine_solar_zenith_angle(ha, decl, 0)
>>> iesd = solrad.inverse_earth_sun_distance(doy)
>>> solrad.inst_solar_radiation_toa(csza, iesd)
1299.9181944414036
```

ETLook.solar_radiation.inverse_earth_sun_distance

ETLook.solar_radiation.**inverse_earth_sun_distance**(doy)

Computes the inverse earth sun distance (iesd) in Angstrom Unit where 1 AU is 1.496e8 km

$$d_r = 1 + 0.033 \cos\left(\frac{2\pi J}{365}\right)$$

Parameters doy : float

julian day of the year J [-]

Returns iesd : float

inverse earth sun distance dr [AU]

Examples

```
>>> import ETLook.solar_radiation as solrad >>> solrad.inverse_earth_sun_distance(180)
0.96703055420162642
```

ETLook.solar_radiation.seasonal_correction

ETLook.solar_radiation.**seasonal_correction**(doy)

Computes the seasonal correction for solar time in hours

$$b = \frac{2\pi\left(J - 81\right)}{364}$$

$$s_c = 0.1645 sin\left(2b\right) - 0.1255 cos\ left(b - 0.025\left(b\right)$$

Parameters doy : float

julian day of the year J [-]

Returns sc : float

seasonal correction s_c [hours]

Examples

```
>>> import ETLook.solar_radiation as solrad
>>> solrad.seasonal_correction(180)
-0.052343379605521212
```

ETLook.solar_radiation.slope_rad

ETLook.solar_radiation.**slope_rad**(*slope_deg*) Converts slope from degrees to radians.

Parameters slope_deg : float

slope in degrees s [deg]

Returns slope : float

slope Δ [rad]

ETLook.solar_radiation.sunset_hour_angle

ETLook.solar_radiation.**sunset_hour_angle**(lat, decl)

Computes the sunset hour angle

$$w_s = \arccos(-\tan(\lambda)\tan(\delta))$$

Parameters decl : float

solar declination δ [rad]

lat : float

latitude λ [rad]

Returns ws : float

sunset hour angle w_s [rad]

5.7.8 Plant stress (ETLook.stress)

stress_moisture (se_root[, tenacity])	Computes the stress for plants when there is not sufficient soil moisture in the root zone
stress_radiation (ra_24)	Computes the stress for plants when there is not sufficient radiation
stress_temperature (t_air_24[, t_opt, t_min, …])	Computes the stress for plants when it is too cold or hot
stress_vpd (vpd_24[, vpd_slope])	Computes the stress for plants if the vpd increases too much.

5.7.9 Canopy and Soil Resistance (ETLook.resistance)

`atmospheric_canopy_resistance` (lai_eff, …)	Computes canopy resistance excluding soil moisture stress
`canopy_resistance` (r_canopy_0, stress_moist)	Computes canopy resistance
`soil_resistance` (se_top[, land_mask, …])	Computes soil resistance

5.7.10 Neutral Atmosphere (ETLook.neutral)

`initial_canopy_aerodynamic_resistance` (u_24, z0m)	Computes the aerodynamic resistance for a canopy soil without stability corrections $r^0_{a,}$.
`initial_daily_evaporation` (rn_24_soil, g0_24, …)	Computes the soil evaporation based on the Penman Monteith equation adapted for soil.
`initial_daily_evaporation_mm` (e_24_init, lh_24)	Computes the soil evaporation based on the Penman Monteith equation adapted for soil.
`initial_daily_transpiration` (rn_24_canopy, …)	Computes the soil evaporation based on the Penman Monteith equation adapted for soil.
`initial_daily_transpiration_mm` (t_24_init, lh_24)	Computes the canopy transpiration based on the Penman Monteith equation adapted for canopy.
`initial_soil_aerodynamic_resistance` (u_24[, …])	Computes the aerodynamic resistance for soil without stability corrections $r^0_{a,soil}$.

ETLook.neutral.initial_daily_evaporation_mm

ETLook.neutral.**initial_daily_evaporation_mm**(*e_24_init*, *lh_24*) Computes the soil evaporation based on the Penman Monteith equation adapted for soil.

$$E_0 = E_0 d_{sec} \lambda_{24}$$

where the following constants are used

- d_{sec} seconds in the day = 86400 [s]

Parameters

e_24_init [float] initial estimate daily evaporation E^0 [W m-2]

lh_24 [float] daily latent heat of evaporation λ_{24} [J/kg]

Returns

e_24_init_mm [float] initial estimate daily evaporation in mm E^0 [mm d-1]

ETLook.neutral.initial_daily_transpiration_mm

ETLook.neutral.**initial_daily_transpiration_mm**(t_24_init, lh_24)

Computes the canopy transpiration based on the Penman Monteith equation adapted for canopy.

$$T_0 = T_0 d_{sec} \lambda_{24}$$

where the following constants are used

- d_{sec} seconds in the day = 86400 [s]

Parameters

t_24_init [float] initial estimate daily transpiration E^0 [W m-2]

lh_24 [float] daily latent heat of evaporation λ_{24} [J/kg]

Returns

t_24_init_mm [float] initial estimate daily transpiration in mm T^0 [mm d-1]

5.7.11 Unstable Atmosphere (ETLook.unstable)

evaporation (rn_24_soil, g0_24, ssvp_24, ...)	Computes the evaporation using an iterative approach.
evaporation_mm (e_24, lh_24)	Computes the soil evaporation based on the Penman Monteith equation adapted for soils.
friction_velocity (u_b, z_b, z0m, disp, sf)	Computes the friction velocity
initial_friction_velocity_daily (u_b_24, z0m, ...)	Computes the initial friction velocity without using stability corrections.
initial_sensible_heat_flux_canopy_daily (...)	Computes the initial sensible heat flux before the iteration which solves the stability corrections.
initial_sensible_heat_flux_soil_daily (...)	Computes the initial sensible heat flux before the iteration which solves the stability corrections.
monin_obukhov_length (h_flux, ad, u_star, t_air_k)	Computes the Monin-Obukhov length.
ra_canopy (h_canopy_init, t_air_k, ...[, ...])	Computes the aerodynamical resistance for canopy using an iterative approach.
ra_soil (h_soil_24_init, t_air_k, ...[, ...])	Computes the aerodynamical resistance for canopy using an iterative approach.
stability_correction_heat_obs (x_b_obs)	Computes the stability correction for heat at observation height.
stability_factor (x_b)	Computes the stability correction for heat at blending height.
stability_parameter (monin, disp[, z_b])	Computes the stability parameter introduced by Monin and Obukhov.
stability_parameter_obs (monin, z_obs)	Computes the stability parameter introduced by Monin and Obukhov.
transpiration (rn_24_canopy, ssvp_24, ad_24, ...)	Computes the transpiration using an iterative approach.
transpiration_mm (t_24, lh_24)	Computes the canopy transpiration based on the Penman Monteith equation adapted for canopy.

ETLook.unstable.friction_velocity

ETLook.unstable.**friction_velocity**(*u_b*, *z_b*, *z0m*, *disp*, *sf*)

Computes the friction velocity

$$u_* = \frac{ku_b}{ln\left(\frac{z_b - d}{z_{0,m}}\right) - \psi_{h,b}}$$

Parameters u_b : float

windspeed at blending height u_b [m]

z_b : float

blending height z_b [m]

z0m : float

roughness length $z_{0,m}$ [m]

disp : float

displacement height d [m]

sf : float stability factor at blending height $\psi_{h,b}$

[m]

Returns u_star : float

friction velocity u_* [m s-1]

ETLook.unstable.monin_obukhov_length

ETLook.unstable.**monin_obukhov_length**(h_flux, ad, u_star, t_air_k)

Computes the Monin-Obukhov length. The Monin-Obukhov length is used to describe the effects of buoyancy on turbulent flows. The Monin-Obukhov length is usually negative during daytime.

$$L = \frac{-\rho c_p u_*^3 T_a}{kgH_{canopy}}$$

Parameters h_flux : float

sensible heat flux H [W m-2]

ad : float

air density ρ [kg m-3]

u_star : float

Monin Obukhov length L [m]

t_air_k : float

air tempererature in kelvin T_a [K]

Returns monin : float

monin obukhov length L [m]

ETLook.unstable.ra_canopy

ETLook.unstable.**ra_canopy**(*h_canopy_init*, *t_air_k*, *u_star_init*, *ad*, *z0m*, *disp*, *u_b*, *z_obs=2*, *z_b=100*, *iter_ra=3*)

Computes the aerodynamical resistance for canopy using an iterative approach. The iteration is needed to compute the frication velocity at blending height Iteration stops either after five iterations or if the difference between two subsequent estimations is less than 0.01.

$$\begin{cases} L = \frac{-\rho c_p u_*^3 T_a}{kgH_{canopy}} \\ x_b = 1 - 16\left(\frac{z_b - d}{L}\right)^{0.25} \\ \psi_{h,b} = 2\ln\left(\frac{1+z_b}{2}\right) + \ln\left(\frac{1+z_b^2}{2}\right) - 2\arctan\left(x_b\right) + 0.5\pi \\ u_* = \frac{ku_b}{ln\left(\frac{z_b-d}{z_{0,m}}\right) - \psi_{h,b}} \end{cases}$$

The friction velocity is independent of height. So this value can be used to calculate together with the stability correction for heat on observation heigth the aerodynamical resistance.

$$x_{obs} = 1 - 16\left(\frac{z_{obs}}{L}\right)^{0.25}$$

$$\psi_{h,obs} = 2\ln\left(\frac{1+x_{obs}^2}{2}\right)$$

$$r_{a,canopy} = \frac{\ln\left(\frac{z_{obs}-d}{0.1z_{0,m}}\right) - \psi_{h,obs}}{ku_*}$$

Parameters h_canopy_init : float

initial estimate of the sensible heat flux H_{canopy} [W m-2]

t_air_k : float

air tempererature in kelvin T_a [K]

u_star_init : float

initial estimate of the daily friction velocity u_* [m s-1]

ad : float

air density ρ [kg m-3]

z_b : float

blending height z_b [m]

z_obs : float

observation height z_{obs} [m]

z0m : float

roughness length $z_{0,m}$ [m]

disp : float

displacement height d [m]

u_b : float

windspeed at blending height u_b [m/s]

iter_ra : integer

number of iterations for aerodynamical resistance n_{ra} [-]

Returns ra_canopy : float

aerodynamical resistance for canopy $r_{a,canopy}$ [s m-1]

ETLook.unstable.ra_soil

ETLook.unstable.**ra_soil**(*h_soil_24_init*, *t_air_k*, *u_star_24_init*, *ad*, *disp*, *u_b*, *z_obs=2*, *z_b=100*, *iter_ra=3*)

Computes the aerodynamical resistance for canopy using an iterative approach. The iteration is needed to compute the friction velocity at blending height Iteration stops either after five iterations or if the difference between two subsequent estimations is less than 0.01.

$$\begin{cases} L = \frac{-\rho c_p u_*^3 T_a}{kgH_{soil}} \\ x_b = 1 - 16\left(\frac{z_b - d}{L}\right)^{0.25} \\ \psi_{h,b} = 2\ln\left(\frac{1+z_b}{2}\right) + \ln\left(\frac{1+z_b^2}{2}\right) - 2\arctan\left(x_b\right) + 0.5\pi \\ u_* = \frac{ku_b}{ln\left(\frac{z_b - d}{z_{0,soil}}\right) - \psi_{h,b}} \end{cases}$$

The friction velocity is independent of height. So this value can be used to calculate together with the stability correction for heat on observation heigth the aerodynamical resistance.

$$x_{obs} = 1 - 16\left(\frac{z_{obs}}{L}\right)^{0.25}$$
$$\psi_{h,obs} = 2\ln\left(\frac{1+x_{obs}^2}{2}\right)$$
$$r_{a,soil} = \frac{\ln\left(\frac{z_{obs}-d}{0.1z_{0,soil}}\right) - \psi_{h,obs}}{ku_*}$$

Parameters h_soil_24_init : float

initial estimate of the sensible heat flux for soil H^{soil} [W m-2]

t_air_k : float air tempererature in kelvin T_a [K]

u_star_24_init : float

initial estimate of the daily friction velocity u_* [m s-1]

ad : float

air density ρ [kg m-3]

z_b : float

blending height z_b [m]

z_obs : float

observation height z_{obs} [m]

disp : float

displacement height d [m]

u_b : float

windspeed at blending height u_b [m]

iter_ra : integer

number of iterations for aerodynamical resistance n_{ra} [-]

Returns ra_soil : float

aerodynamical resistance for soil $r_{a,soil}$ [s m-1]

ETLook.unstable.stability_correction_heat_obs

ETLook.unstable.**stability_correction_heat_obs**(x_b_obs)

Computes the stability correction for heat at observation height.

$$\psi_{h,obs} = 2\ln\left(\frac{1+x_{obs}^2}{2}\right)$$

Parameters x_b_obs : float

stability parameter used in stability correction for observation height x_{obs} [-]

Returns sf_obs : float

stability correction for heat for observation height $\psi_{h,obs}$ [-]

ETLook.unstable.stability_factor

ETLook.unstable.**stability_factor**(x_b)

Computes the stability correction for heat at blending height.

$$\psi_{h,b} = 2\ln\left(\frac{1+x_b}{2}\right) + \ln\left(\frac{1+x_b^2}{2}\right) - 2\arctan\left(x_b\right) + 0.5\pi$$

Parameters x_b : float

stability parameter used in stability correction x_b [-]

Returns sf : float

stability correction for heat $\psi_{h,b}$ [-]

ETLook.unstable.stability_parameter

ETLook.unstable.**stability_parameter**(monin, disp, z_b=100)

Computes the stability parameter introduced by Monin and Obukhov. This parameter includes effects of both shear stress and buoyancy on turbulence. It is applicable to blending height.

$$x_b = 1 - 16\left(\frac{z_b - d}{L}\right)^{0.25}$$

Parameters monin : float

monin obukhov length L [m]

z_b : float

blending height z_b [m]

disp : float

displacement height d [m]

Returns x_b : float

stability parameter used in stability correction x_b [-]

ETLook.unstable.stability_parameter_obs

ETLook.unstable.**stability_parameter_obs**(monin, z_obs)

Computes the stability parameter introduced by Monin and Obukhov. This parameter includes effects of both shear stress and buoyancy on turbulence. It is applicable to observation height.

$$x_{obs} = 1 - 16\left(\frac{z_{obs}}{L}\right)^{0.25}$$

Parameters monin : float

monin obukhov length L [m]

z_obs : float

observation height z_{obs} [m]

Returns x_b_obs : float
stability parameter used in stability correction for observation height x_{obs} [-]

5.7.12 Soil Moisture (ETLook.soil_moisture)

The soil_moisture module contains all functions related to soil moisture data components.

`psi_m` (y)	Computes the stability correction for momentum based on Brutsaert (1999) [R7].
`psi_h` (y)	Computes the stability correction for momentum based on Brutsaert (1999) [R8].
`aerodynamical_resistance_bare` (u_i, L_bare[, ...])	Computes the aerodynamical resistance for a dry bare soil.
`aerodynamical_resistance_full` (u_i, L_full[, ...])	Computes the aerodynamical resistance for a full canopy.
`aerodynamical_resistance_soil` (u_i_soil)	Computes the aerodynamical resistance of the soil
`atmospheric_emissivity_inst` (vp_i, t_air_k_i)	Computes the atmospheric emissivity according to Brutsaert [R8].
`friction_velocity_bare_inst` (u_b_i_bare[, ...])	Like `initial_friction_velocity_inst()` but with bare soil parameters
`friction_velocity_full_inst` (u_b_i_full[, ...])	Like `initial_friction_velocity_inst()` but with full vegetation parameters
`initial_friction_velocity_inst` (u_b_i, z0m, disp)	Computes the initial instantaneous friction velocity without stability corrections.
`maximum_temperature` (t_max_bare, t_max_full, vc)	Computes the maximum temperature at dry conditions
`maximum_temperature_bare` (ra_hor_clear_i, ...)	Computes the maximum temperature under dry bare soil conditions
`maximum_temperature_full` (ra_hor_clear_i, ...)	Computes the maximum temperature under fully vegetated conditions
`monin_obukhov_length_bare` (h_bare, ad_i, ...)	Like `unstable.monin_obukhov_length()` but with bare soil parameters.
`monin_obukhov_length_full` (h_full, ad_i, ...)	Like `unstable.monin_obukhov_length()` but with full canopy parameters.
`net_radiation_bare` (ra_hor_clear_i, ...[, ...])	Computes the net radiation for the bare soil with zero evaporation
`net_radiation_full` (ra_hor_clear_i, ...[, ...])	Computes the net radiation at full canopy with zero evaporation
`sensible_heat_flux_bare` (rn_bare[, ...])	Computes the bare soil sensible heat flux
`sensible_heat_flux_full` (rn_full[, ...])	Computes the full canopy sensible heat flux
`soil_moisture_from_maximum_temperature` (...)	Computes the relative root zone soil moisture based on estimates of maximum temperature and wet bulb temperature and measured land surface temperature
`wind_speed_blending_height_bare` (u_i[, ...])	Computes the wind speed at blending height u_b [m/s] using the logarithmic wind profile
`wind_speed_blending_height_full_inst` (u_i[, ...])	Computes the wind speed at blending height u_b [m/s] using the logarithmic wind profile
`wind_speed_soil_inst` (u_i, L_bare[, z_obs])	Computes the instantaneous wind speed at soil surface

ETLook.soil_moisture.psi_m

ETLook.soil_moisture.**psi_m**(*y*)

Computes the stability correction for momentum based on Brutsaert (1999)[23].

$$\Psi_M(y) = \ln(a+y) - 3by^{\frac{1}{3}} + \\ \frac{ba^{\frac{1}{3}}}{2}\ln[\frac{(1+x)^2}{(1-x+x^2)}] + \\ \sqrt{3}ba^{\frac{1}{3}}\arctan[\frac{(2x-1)}{\sqrt{3}}] + \Psi_0$$

where the following constants are used

[23] Brutsaert, W., Aspect of bulk atmospheric boundary layer similarity under free-convective conditions, Reviews of Geophysics, 1999, 37(4), 439-451.

- *a* = 0.33
- *b* = 0.41 in which

$$x = (\frac{y}{a})^{\frac{1}{3}}$$

and

$$y = \frac{-(z-d)}{L}$$

where L is the monin obukhov length defined by *ETLook.unstable.monin_obukhov_length()*, z and d are the measurement height and displacement height respectively. All aforementioned parameters are different for the bare soil and full canopy solutions.

The symbol Ψ_0 denotes a constant of integration, given by

$$\Psi_0 = -\ln a + \sqrt{3}ba^{\frac{1}{3}}\frac{\pi}{6}$$

Notes

This function should not be used as an input function for a ETLook tool. This function is used internally by *aerodynamical_resistance_bare()* and *aerodynamical_resistance_full()* and wind_speed_soil().

ETLook.soil_moisture.psi_h

ETLook.soil_moisture.**psi_h**(*y*)

Computes the stability correction for momentum based on Brutsaert (1999)[14*].

$$\Psi_H(y) = [\frac{(1-d)}{n}]\ln\frac{(c+y^n)}{c}$$

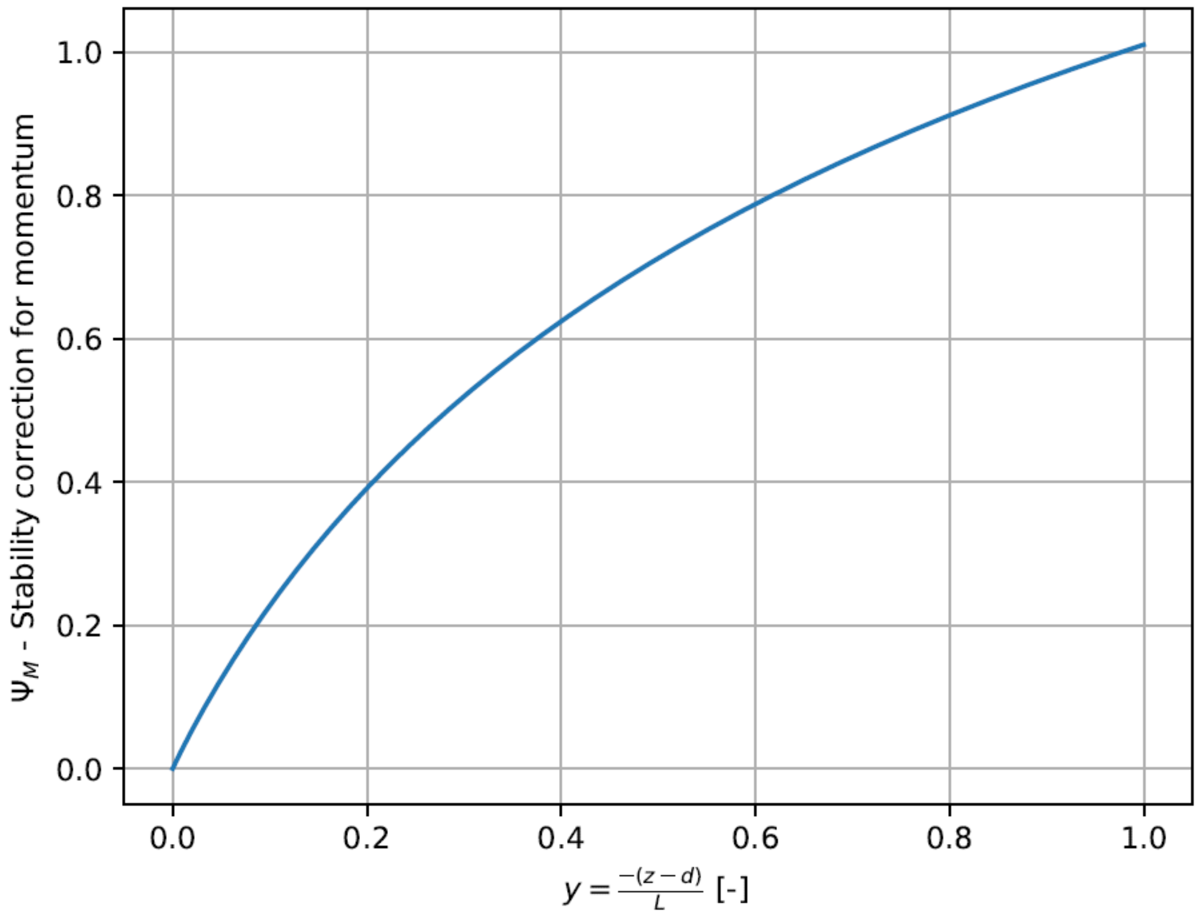
where the following constants are used

- *c* = 1.00
- *d* = 0.057
- *n* = 0.78

in which

$$y = \frac{-(z-d)}{L}$$

where L is the monin obukhov length defined by *ETLook.unstable.monin_obukhov_length()*, z and d are the measurement height and displacement height respectively. All aforementioned parameters are different for the bare soil and full canopy solutions.



Notes

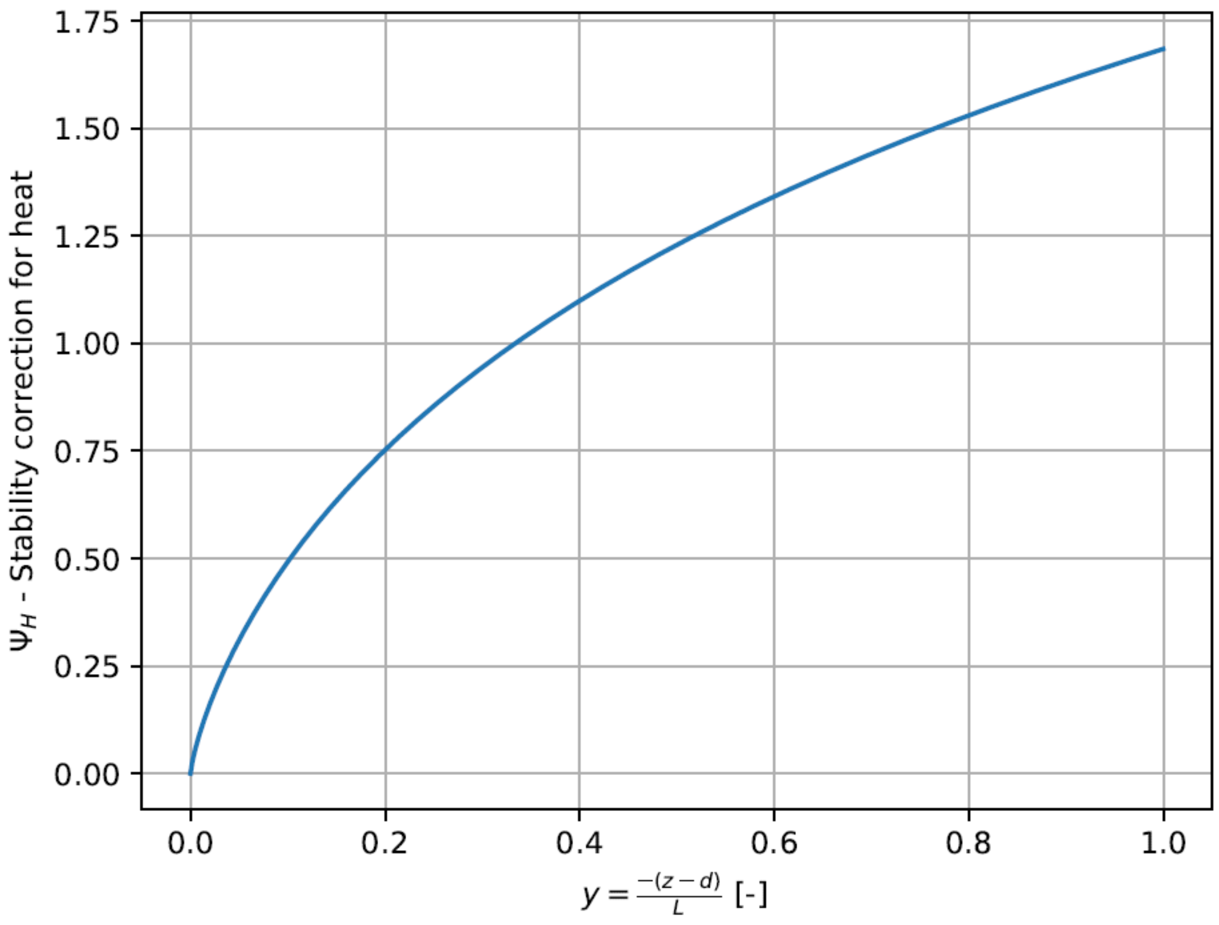
This function should not be used as an input function for a tool. This function is used internally by *aerodynamical_resistance_bare()* and *aerodynamical_resistance_full()* and wind_speed_soil().

ETLook.soil_moisture.initial_friction_velocity_inst

ETLook.soil_moisture.**initial_friction_velocity_inst**(u_b_i, z0m, disp, z_b=100)

Computes the initial instantaneous friction velocity without stability corrections.

$$u_* = \frac{ku_b}{\ln\left(\frac{z_b - d}{z_{0,m}}\right)}$$



Parameters

u_b_i [float]

instantaneous wind speed at blending height u_b [m s-1]

z0m [float] surface roughness $z_{0,m}$ [m]

disp [float] displacement height d [m]

z_b [float] blending height z_b [m]

Returns

u_star_i_init [float] initial estimate of the instantaneous friction velocity $u_{*,i}$ [m s-1]

6 Function parameters

This chapter gives the values used for static parameters in the functions described in the previous chapter, such as stomatal resistance and maximum obstacle height for the different land cover types according to the land cover classes. The GlobCover and WaPOR land cover classes are listed in the following 2 tables.

Table 7: Static parameters for the GlobCover land cover classes

Landcover class	Bulk stomatal resistance	Maximum obstacle height
Tree Cover, broadleaved, evergreen	100	1.0
Tree Cover, broadleaved, deciduous, closed	120	0.6
Tree Cover, broadleaved, deciduous, open	100	1.2
Tree Cover, needle-leaved, evergreen	150	2.0
Tree Cover, needle-leaved, deciduous	180	5.0
Tree Cover, mixed leaf type	175	8.0
Tree Cover, regularly flooded, fresh water	200	6.0
Tree Cover, regularly flooded, saline water	300	10.0
Mosaic: Tree Cover / Other natural vegetation	350	8.0
Tree Cover, burnt	250	7.0
Shrub Cover, closed-open, evergreen	200	5.0
Shrub Cover, closed-open, deciduous	175	4.0
Herbaceous Cover, closed-open	250	2.0
Sparse herbaceous or sparse shrub cover	150	1.0
Regularly flooded shrub and/or herbaceous cover	250	0.3
Cultivated and managed areas	200	4.0
Mosaic: Cropland / Tree Cover / Other natural vegetation	150	3.5
Mosaic: Cropland / Shrub and/or grass cover	150	2.0
Bare Areas	400	10.0
Water Bodies	100	0.1
Snow and Ice	100	0.1
Artificial surfaces and associated areas	100	0.1

Table 8: Static parameters for the WaPOR land cover classes

Landcover class	**Bulk stomatal resistance**	**Maximum obstacle height**
Shrubland	175	1
Grassland	175	2
Cropland: rainfed	125	1.5
Cropland: irrigated or post-flooding	125	1.5
Cropland: fallow	125	1.5
Urban areas	100	0.1
Bare areas	400	10.0
Permanent snow and ice	100	0.1
Water bodies	100	0.1
Temporary water bodies	100	0.1
Shrub or herbaceous cover: flooded fresh/saline/brackish water	150	2
Moss and lichen	180	0.3
Tree cover: needleleaved evergreen closed (>40%)	300	10
Tree cover: broadleaved evergreen closed (>40%)	180	5
Tree cover: needleleaved deciduous closed (>40%)	Not present	Not present
Tree cover: broadleaved deciduous closed (>40%)	190	8
Tree cover: mixed forest type closed (broadleaf & needleleaf)	200	8
Tree cover: Unknown closed forest type	180	5
Tree cover: needleleaved evergreen open (15-40%)	250	5
Tree cover: broadleaved evergreen open (15-40%)	180	4
Tree cover: needleleaved deciduous open (15-40%)	Not present	Not present
Tree cover: broadleaved deciduous open (15-40%)	200	4
Tree cover: mixed forest type open (broadleaf & needleleaf)	180	5
Tree cover: Unknown open forest type	250	3
Coastal water	100	0.1

References

Bastiaanssen, W.G.M., Cheema, M.J.M., Immerzeel, W. W., Miltenburg, I.J. and Pelgrum, H., (2012). Surface energy balance and actual evapotranspiration of the transboundary Indus Basin estimated from satellite measurements and the ETLook model. Water Resources Research, 48(11).

Braden, H., Energiehaushalts- und Verdunstungsmodell für Wasser- und Stoffhaushalts-untersuchungen landwirtschaftlich genutzter Einzugsgebiete. Mitteilungen der Deutschen Bodenkundlichen Gesellschaft, (1985), 42, 254-299

Brutsaert, W., On a derivable formula for long-wave radiation from clear skies, Water Resour. Res, 1975, 11, 742-744.

Brutsaert, W., Aspect of bulk atmospheric boundary layer similarity under free-convective conditions, Reviews of Geophysics, 1999, 37(4), 439-451.

Chen, J., et al. "A simple method for reconstructing a high-quality NDVI time-series data set based on the Savitzky-Golay filter" Remote Sensing of Environment, 91, 332-344 (2004).

Chen, J., et al., "A simple and effective method for filling gaps in Landsat ETM+ SLC-off images" Remote Sensing of Environment 115 1053-1064 (2011)

Gruter, J. W. (ed), 1984, Radiation Nomenclature, Brussels, CEC, Second Solar Energy Programme, Project F, Solar Radiation Data

Hazaymeh, K., & Hassan, Q. K., "Spatiotemporal image-fusion model for enhancing the temporal resolution of Landsat-8 surface reflectance images using MODIS images" Journal of Applied Remote Sensing, 9(1) (2015). Changes applied to methodology.

von Hoyningen-Hüne, J., Die Interception des Niederschlags in landwirtschaftlichen Beständen. Schriftenreihe des DVWK, 1983, 57, 1-53

Jiménez-Muñoz, J.C., Cristóbal, J., Sobrino, J.A., Sòria, G., Ninyerola, M. and Pons X. "Revision of the Single-Channel Algorithm for Land Surface Temperature Retrieval From Landsat Thermal-Infrared Data" IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING 47 (1): 339-349 (2009)

Kasten F. 1996, The Linke turbidity factor based on improved values of the integral Rayleigh optical thickness. Solar Energy 56: 239-44

Kasten F., and T.A. Young. 1989, Revised optical air mass tables and approximation formula, Applied Optics 28:4735-8

Rahman, H. and G. Dedieu, SMAC: a simplified method for the atmospheric correction of satellite measurements in the solar spectrum, International Journal of Remote Sensing 1994, 15(1), 123-143, doi.org/10.1080/01431169408954055

Rozenstein, O., Qin, Z., Derimian, Y. And Karnieli, A. Derivation of Land Surface Temperature for Landsat-8 TIRS Using a Split Window Algorithm, Sensors, 2014, 14(4):5768-5780, https://doi.org/10.3390/s140405768

Šúri, M., Hofierka, J., A new GIS-based Solar Radiation Model and Its Application to Photovoltaic Assessments, Transactions in GIS, 2004, 8(2): 175-190.

Swets, D.L, Reed, B.C., Rowland, J.D., Marko, S.E., 1999. A weighted least-squares approach to temporal NDVI smoothing. In: Proceedings of the 1999 ASPRS Annual Conference, Portland, Oregon, pp. 526-536.

Tasumi, M. Allen, R. G, and R. Trezza. 2006. DEM based solar radiation estimation model for hydrological studies. Hydrological Science and Technology 22:197-208.

Yang, Y., H. Guan, D. Long, B. Liu, G. Qin, J, Qin, and O. Batelaan, Estimation of Surface SoilMoisture from Thermal Infrared Remote Sensing Using an Improved Trapezoid Method, Remote Sens. 2015, 7, 8250-8270; doi:10.3390/rs70708250