A Python Package for Agro-Ecological Zoning

User Guide for PyAEZ (v 2.2)

Published by
Food and Agriculture Organization of the United Nations
and
Asian Institute of Technology

Required citation:

FAO and AIT. 2023. *A Python Package for Agro-Ecological Zoning. User Guide for PyAEZ (v 2.2).* Rome.
Full DOI link.

Contents

Figures

Tables

ACKNOWLEDGEMENTS

Please add this section

The Python Agro-Ecological Zoning (PyAEZ) was born out of the necessity to address country-specific AEZ modeling, as initiated by the FAO-RAP under the activity on 'Capacity building for Agro-ecological Zone mapping and modelling to project climate suitability of crops and land uses' and the 'Strengthening Agro-climatic Monitoring and Information System (SAMIS)' project in Lao PDR.

The authors thank all those from the AIT-GIC and FAO who contributed towards the initial iteration of the code development, Lakmal Nawarathnage, Thaileng Thol, Gianluca Franceschini, Shraddha Sharma, and Kavinda Gunasekara. Thanks to Kittiphon Boonma and Swun Wunna Htet of AIT-GIC for the second iteration of PyAEZ. Special thanks to the theoretical and technical advice and support from Günther Fischer and Freddy Nachtergaele. We are grateful for the funding from FAO and technical support from both FAO and IIASA.

ABBREVIATIONS AND ACRONYMS

AEZ	Agro-Ecological Zones
AIT-GIC	Asian Institute of Technology – Geoinformatics Center
AWC	Available soil Water Capacity
BADC	British Atmospheric Data Centre
CMIP5	Coupled Model Inter-comparison Project Phase 5
CRU	Climate Research Unit
DEM	Digital Elevation Model
ESM	Earth System Model
ETa	Actual evapotranspiration
ETm	Maximum evapotranspiration
ETo	Reference evapotranspiration
FAO	Food and Agriculture Organization of the United Nations
FAO-RAP	FAO – Regional Office for Asia and the Pacific
fc1	Thermal reduction factor
fc2	Moisture reduction factor
fc3	Agro-climatic constraint factor
fc4	Soil reduction factor
fc5	Terrain reduction factor
FI	Fournier Index
Fm	Modified Fournier Index
GAEZ	Global Agro-ecological Zones
HI	Harvest index
HWSD	Harmonized World Soil Database
IIASA	International Institute for Applied System Analysis
LAI	Leaf Area Index
LGP	Length of Growing Period
LGPagc	Wetness days

LGPt10	Frost Free Period
LGPt5	Temperature Growing Period
LUTs	Land Utilization Types
P/ETo	Moisture Availability Index
SMU	Soil Mapping Unit
SQ	Soil Quality

INTRODUCTION

The world population is expected to reach 8.6 billion in 2030 and 9.8 billion in 2050 (UNDESA, 2017). With this eruptive growth in population, an unprecedented increase in demand for food, feed, and fuel is expected, while the agricultural land needed for production continues to shrink in many parts of the world.

The accelerating pace of climate change, combined with global population growth, threatens food security globally. Higher temperatures eventually reduce yields of desirable crops while encouraging weed and pest proliferation and changes in precipitation patterns increase the likelihood of short-run crop failures and long-run production declines (Nelson *et al*., 2009).

Yield increases on existing croplands will, therefore, be an essential component to increase food production (Ray *et al*., 2013). To this end, Agro-Ecological Zoning (AEZ) framework was developed as a tool to analyse the effect of climate on land use and agriculture, as well as helping to optimise the crop cycle to produce the best yield possible. PyAEZ is an open-source Python package which offers AEZ calculations for user to implement for their regional AEZ analyses. This technical document contains detailed descriptions of all the AEZ modules and functions in PyAEZ.

Background

Over the last thirty years, FAO and the International Institute for Applied Systems Analysis (IIASA) have been developing Agro-Ecological Zoning (AEZ). AEZ is a modelling system for land evaluation to support sustainable land use planning, stimulate agricultural investments, monitor the status of agricultural resources, and assess the impacts of climate change on agriculture.

The Agro-Ecological Zoning (AEZ) approach, developed by FAO jointly with IIASA, is based on the principles of land evaluation and defines matching procedures to identify crop-specific limitations of prevailing climate, soil, and terrain resources with simple and robust crop models, under assumed levels of inputs and management conditions. Main outputs are maximum potential and agronomically attainable crop yields and suitability levels for basic land resources units under different agricultural production systems defined by water supply systems and levels of inputs and management circumstances.

While most countries have adopted land evaluation, land suitability assessment, agro-ecological zoning in the past to prepare agricultural investment plans, most of those are outdated. In parallel, various generations of "AEZ Projects" have served as vehicles to consolidate efforts, structure project goals, and promote funding and resources for the further development of these concepts. And while new datasets and technologies are increasingly becoming available, national capacities to develop, update and use agro-ecological zoning remain limited.

The GAEZ assessment, currently at its fourth update (GAEZ v4), uses seven different modules that are run sequentially to generate agro-climatic and crop-specific information. Each module is made up of a series of FORTRAN routines, documented in GAEZ version 4 model documentation (Fischer *et al*., 2021), that are run through custom batch scripts. Additional scripts in Delphi are used for specific modelling (to be clearly specified by IIASA). Data preparation, although fully documented on the theoretical concepts, is mostly undocumented when addressing the required process to input new data in the modelling system. Limited knowledge on data preparation and lack of a systematic system to run FORTRAN routines make the capacity to generate outputs, limited to a restricted number of experts at IIASA.

The main strategic shift of GAEZ v5 is to focus on how the strong scientific basis for GAEZ can be made available to national entities to support decision-making. GAEZ v5 will focus on taking the last three decades of knowledge and building a standardized, repeatable, accessible yet extensible approach for countries to implement their own, fit for purpose, nationally adjusted Agro-Ecological Zoning project(s). Countries need guidance on how to collect relevant local data, what tools they can use to create data if it does not exist, how to engage with farmers, local representatives, and other stakeholders, and how to process, manage, host, and disseminate results of an analysis.

With a growing need to address country-specific agro-ecological zoning modelling, the "Strengthening Agro-climatic Monitoring and Information System (SAMIS)"[1] project in Lao PDR, in collaboration with the Geomatics Unit of the Asian Institute of Technology (AIT) developed a first prototype in Python language, named PyAEZ, that generate national AEZ information. The code, with supporting documentation, and training material is publicly available in the GitHub repository at the https://github.com/gicait/PyAEZ.

What is PyAEZ?

PyAEZ is the first step of GAEZ expansion that utilizes Python scripts to develop users' AEZ projects. The PyAEZ package utilizes climate, soil, and terrain conditions relevant to agricultural production and suitability using crop-specific land resource inventory parameters.

The package is developed with several Python routines and is operated with Jupyter Notebooks, which means it has the capability to be uploaded onto Google Colab, an online Jupyter Notebook system. This compatibility with an online platform such as Google Colab allowed the development team to host two virtual hands-on trainings where attendants were guided through the scientific concepts of AEZ as well as executing the scripts with country-specific input data, through Google Colab.

PyAEZ has been developed to be used within the tropical region, hence some of the complexity of GAEZ in non-tropical regions (e.g., vernalization requirements, permafrost evaluation) is not accounted for. Moreover, the system has not been tested on larger areas where performances of results may be an issue.

PyAEZ package consists of 6 main AEZ modules and 1 additional utility module (Figure 1):

- **Module 1: Climate Regime** – calculation of agro-climatic indicators for evaluation of climatic suitability of crops;

[1] Further information on the SAMIS project can be found on the FAO page: http://www.fao.org/in-action/samis/en/

- **Module 2: Crop Simulation** – simulate an optimal crop cycle for the highest attainable yield;
- **Module 3: Climate Constraints** – application of agro-climatic constraints to the calculated yield of a particular crop;
- **Module 4: Soil Constraints** – application of edaphic constraints to the calculated yield of a particular crop;
- **Module 5: Terrain Constraints** – application of terrain constraints to the calculated yield of a particular crop;
- **Module 6: Economic Suitability Analysis** – evaluation of economic profitability of a crop based on crop price and the calculated yield, and
- **Utilities Calculations,** miscellaneous calculation routines used throughout the 6 main AEZ modules.

The package is also equipped with additional calculation routines for:

i. Water Balance Calculation and applying of yield reduction factors based on water limitation (FAO CropWat algorithm) (Smith, 1992);

ii. Biomass Calculation produced by Photosynthesis activities of plants under given radiation conditions, and

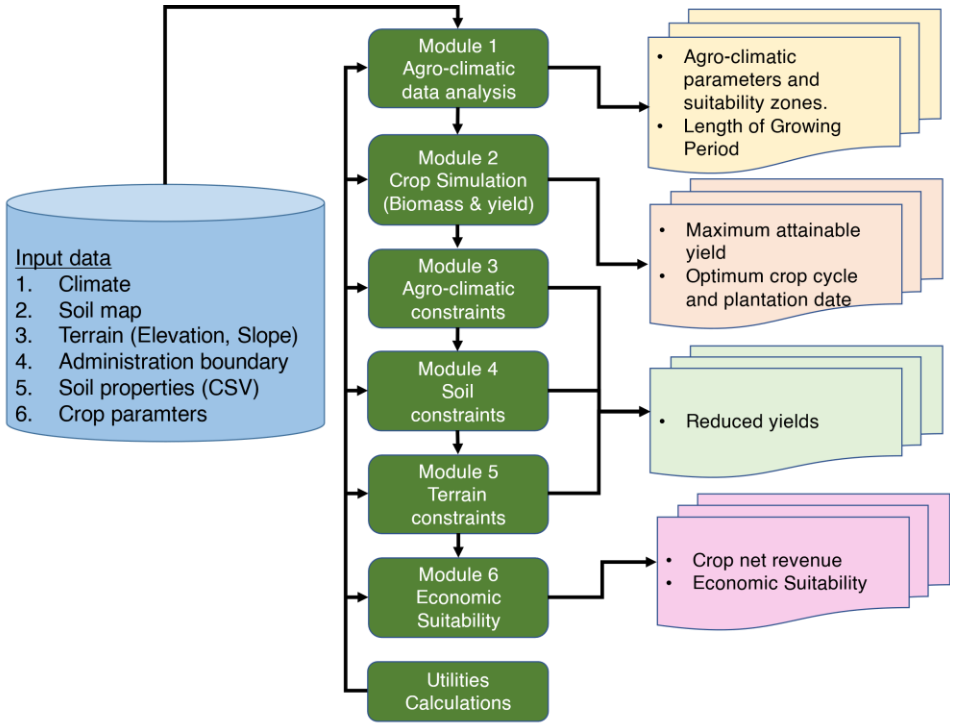
iii. Reference Evapotranspiration Calculation using Penman-Monteith algorithm (Allen *et al*., 1998; Monteith, 1965, 1981).

This documentation provides a step-by-step guideline for anyone looking to develop an AEZ project using PyAEZ package, starting from the installation to the description of the functions in each module, as well as the theoretical concepts behind each function/module.

The code, with supporting documentation, and training material (Jupyter Notebooks and example data) is publicly available and can be downloaded and installed through:

- PyAEZ GitHub repository (https://github.com/gicait/PyAEZ)
- Python package management systems '*pip*' and '*conda*'

Figure 1 Overview of PyAEZ workflow



DATA REQUIREMENT AND PREPARATION

This section will cover all the system and data requirements to run PyAEZ. These subsections also act as an essential checklist for the necessary elements to every PyAEZ project initiation.

Python dependencies

PyAEZ package requires the following additional open-source Python packages to be installed and imported for the AEZ calculations to work:

- NumPy[2]: NumPy array is the format used throughout PyAEZ for pixel-based calculation;
- GDAL[3]: allow the package to utilize and generate geo-referenced output from non-geocoded NumPy arrays;
- SciPy[4]: offers statistical analyses and is interoperable with NumPy array;
- Pandas[5]: allows PyAEZ to read MS Excel sheets with user-defined parameters;
- Numba[6] and NumPy: aware optimizing compiler used to speed up some computationally heavy routines within PyAEZ.

Commented [SWH1]: New library to add

Data preparation

Input-data preparation is essential as the current version of PyAEZ requires users to input data of specific format and shape. Depending on the nature of each aspect, one might prepare to transform into 2D or 3D NumPy arrays, in other cases, preparing additional excel sheet information will be required.

Climate data

PyAEZ requires 6 climatic parameters (Table 1) to be prepared as 3D NumPy data cube for a single year (row, column, day-of-year). We encourage users to use daily climatic data for more accurate results. If the monthly climate data is used, it will need to be interpolated to daily data. The input climate data can be Historical-type data or Future-projected data. Example of the possible climate data sources are Corpernicus' Climate Data Store[7], European Centre for Medium-Range Weather

[2] NumPy: https://numpy.org/install

[3] GDAL: e.g. https://anaconda.org/conda-forge/gdal

[4] SciPy: https://scipy.org/install

[5] Pandas: https://pandas.pydata.org/docs/getting_started/install.html

[6] Numba: https://numba.readthedocs.io/en/stable/user/installing.html

[7] Corpernicus' Climate Data Store: https://cds.climate.copernicus.eu/

Forecasts (ECMWF)[8], Google Earth Engine (GEE)[9], and etc. Users can also utilize own country data from their national sensor network/database.

Table 1 Input climatic parameters

Climatic parameter	Data frequency	Unit	Data format
Minimum air temperature (2m above surface)	Daily or monthly	Degree Celsius	3D NumPy (row, column, time)
Maximum air temperature (2m above surface)	Daily or monthly	Degree Celsius	3D NumPy (row, column, time)
Total precipitation	Daily or monthly	mm/day	3D NumPy (row, column, time)
Solar radiation	Daily or monthly	W/m^2	3D NumPy (row, column, time)
Relative humidity	Daily or monthly	Percentage	3D NumPy (row, column, time)
Windspeed (2m above surface)	Daily or monthly	m/s	3D NumPy (row, column, time)

During the preparation of climatic data, all NaN values (different climate data tend to have some specified no-value values, e.g., -9999) need to be set to zero to prevent any incomputable errors further down the line.

Soil

PyAEZ requires two soil-related data preparations (Table 2).

Table 2 Soil Data Preparation

Data	Data source	Data format
Soil map	• Harmonized World Soil Database (HWSD)[10] • Own local/regional soil map	• 2D NumPy array • Each pixel refers to a unique soil mapping unit

Commented [SWH2]: Hyperlink update for HWSD v2.0.

[8] ECMWF: https://www.ecmwf.int/en/forecasts/datasets

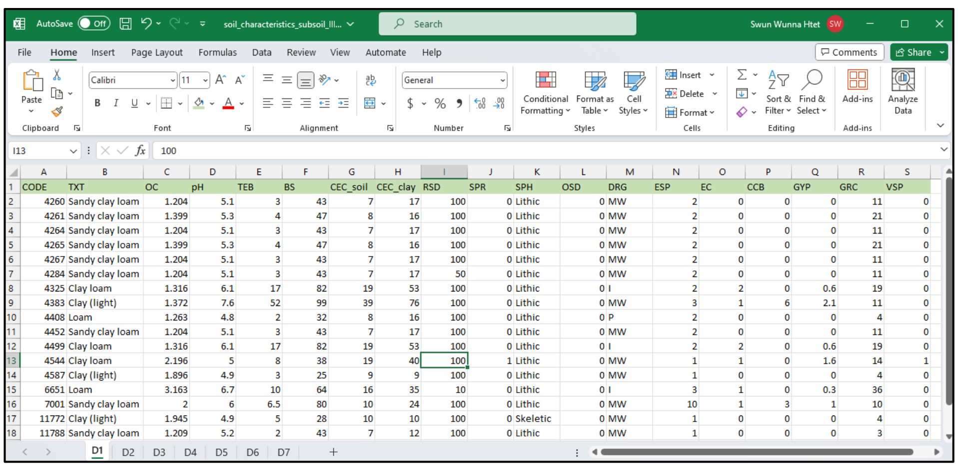
[9] Google Earth Engine: https://developers.google.com/earth-engine/datasets

[10] HWSD https://gaez.fao.org/pages/hwsd

Soil characteristics (Table 3)	• Corresponding to the soil map	• .xlsx file with each soil characteristic parameters as the column headers (Figure 2) • PyAEZ needs 2 .xlsx files, one for topsoil and another for sub-soil*. Each component must have soil properties for seven soil layer classes [D1, D2, D3, D4, D5, D6, D7] in an single excel sheet.
Soil Requirement Reduction Factors	• Corresponding to LUT/crop's edaphic suitability for a single input/management level	• .xlsx files one for rainfed and irrigated conditions. Requires preparing edaphic suitability based on existing

: Pay special attention to the abbreviations when used in the .xlsx file as PyAEZ reads the data using these.

Figure 2 Example of the Soil Characteristics input for existing Soil Mapping Units with format (.xlsx) (for seven soil depth layers)



CODE	TXT	OC	pH	TEB	BS	CEC_soil	CEC_clay	RSD	SPR	SPH	OSD	DRG	ESP	EC	CCB	GYP	GRC	VSP
4260	Sandy clay loam	1.204	5.1	3	43	7	17	100	0	Lithic	0	MW	2	0	0	0	11	0
4261	Sandy clay loam	1.399	5.3	4	47	8	16	100	0	Lithic	0	MW	2	0	0	0	21	0
4264	Sandy clay loam	1.204	5.1	3	43	7	17	100	0	Lithic	0	MW	2	0	0	0	11	0
4265	Sandy clay loam	1.399	5.3	4	47	8	16	100	0	Lithic	0	MW	2	0	0	0	21	0
4267	Sandy clay loam	1.204	5.1	3	43	7	17	100	0	Lithic	0	MW	2	0	0	0	11	0
4284	Sandy clay loam	1.204	5.1	3	43	7	17	50	0	Lithic	0	MW	2	0	0	0	11	0
4325	Clay loam	1.316	6.1	17	82	19	53	100	0	Lithic	0	I	2	2	0	0.6	19	0
4383	Clay (light)	1.372	7.6	52	99	39	76	100	0	Lithic	0	MW	3	1	6	2.1	11	0
4408	Loam	1.263	4.8	2	32	8	16	100	0	Lithic	0	P	2	0	0	0	4	0
4452	Sandy clay loam	1.204	5.1	3	43	7	17	100	0	Lithic	0	MW	2	0	0	0	11	0
4499	Clay loam	1.316	6.1	17	82	19	53	100	0	Lithic	0	I	2	2	0	0.6	19	0
4544	Clay loam	2.196	5	8	38	19	40	100	1	Lithic	0	MW	1	1	0	1.6	14	1
4587	Clay (light)	1.896	4.9	3	25	9	9	100	0	Lithic	0	MW	1	0	0	0	4	0
6651	Loam	3.163	6.7	10	64	16	35	10	0	Lithic	0	I	3	1	0	0.3	36	0
7001	Sandy clay loam	2	6	6.5	80	10	24	100	0	Lithic	0	MW	10	1	3	1	10	0
11772	Clay (light)	1.945	4.9	5	28	10	10	100	0	Skeletic	0	MW	1	0	0	0	4	0
11788	Sandy clay loam	1.209	5.2	2	43	7	12	100	0	Lithic	0	MW	1	0	0	0	3	0

D1 D2 D3 D4 D5 D6 D7

Table 3 Input soil parameters of topsoil and sub-soil properties

Abbreviation *	Parameter name	Data type
CODE	Soil Mapping Unit ID[11]	Numerical

[11] Soil Mapping Unit ID as obtained from the soil map.

TXT	Soil texture	String
OC	Soil organic carbon	Numerical
pH	Soil pH (0–14)	Numerical
TEB	Total exchangeable bases	Numerical
BS	Base saturation	Numerical
CEC_soil	Cation exchange capacity of soil	Numerical
CEC_clay	Cation exchange capacity of clay	Numerical
RSD	Effective soil depth	Numerical
GRC	Soil coarse material (Gravel) percentage	Numerical
DRG	Drainage classes (VP: very poor, P: poor, I: imperfectly, MW: moderately well, W: well, SE: somewhat excessive, E: excessive	String
ESP	Exchangeable sodium percentage	Numerical
EC	Electricity conductivity [dS/m]	Numerical
SPH	Soil phase rating (0 or 1)	Numerical
SPR	Soil property rating (0 or 1)	Numerical
OSD	Obstacles to Roots and Impermeable Layers	Numerical
CCB	Calcium carbonate content percentage (0-100)	Numerical
GYP	Gypsum content percentage (0-100)	Numerical
VSP	Vertic properties (0 = No or 1 = Yes)	Numerical

Table 4 Description and Setting of LUT/Input Management Specific Soil Requirement Parameters as Excel Sheets (for rainfed and Irrigated Conditions)

For Soil Quality 1 (Nutrient Availability)		Data Type
TXT_val	Refer to Soil Texture. Users must prepare existing unique soil textures specific soil reduction factors. The number of elements in **TXT_val** and **TXT_fct** must be the same.	String
TXT_fct		Numerical (0-100)
OC_val	Refers to soil organic carbon content. Users must prepare the soil reduction factors based on each unique OC value. The number of elements in **OC_val** and **OC_fct** must be the same.	Numerical
OC_fct		Numerical (0-100)
pH_val		Numerical

pH_fct	Refers to soil pH. Users must prepare the soil reduction factors based on each unique pH value. The number of elements in **pH_val** and **pH_fct** must be the same.	Numerical (0-100)
TEB_val	Refers to total exchangeable bases. Users must prepare the soil reduction factors based on each unique TEB value. The number of elements in **TEB_val** and **TEB_fct** must be the same.	Numerical
TEB_fct		Numerical (0-100)
For Soil Quality 2 (Nutrient Retention Capacity)		
TXT_val	Refer to Soil Texture. Users must prepare existing unique soil textures specific soil reduction factors. The number of elements in **TXT_val** and **TXT_fct** must be the same.	String
TXT_fct		Numerical (0-100)
BS_val	Refers to base saturation. Users must prepare the soil reduction factors based on each unique BS value. The number of elements in **BS_val** and **BS_fct** must be the same.	Numerical
BS_fct		Numerical (0-100)
CECsoil_val	Refers to cation exchange capacity of soil. Users must prepare the soil reduction factors based on each unique CEC soil value. The number of elements in **CECsoil_val** and **CECsoil_fct** must be the same.	Numerical
CECsoil_fct		Numerical (0-100)
CECclay_val	Refers to cation exchange capacity of clay. Users must prepare the soil reduction factors based on each unique CEC clay value. The number of elements in **CECclay_val** and **CECclay_fct** must be the same.	Numerical
CECclay_fct		Numerical (0-100)
pH_val	Refers to soil pH. Users must prepare the soil reduction factors based on each unique pH value. The number of elements in **pH_val** and **pH_fct** must be the same.	Numerical
pH_fct		Numerical (0-100)
For Soil Quality 3 (Rooting Conditions)		
RSD_val	Refers to rootable soil depth. Users must prepare the soil reduction factors based on each unique RSD value. The number of elements in **RSD_val** and **RSD_fct** must be the same.	Numerical
RSD_fct		Numerical (0-100)
SPH_val	Refers to soil phase. Users must prepare the soil reduction factors based on each unique SPH class. The number of elements in **pH_val** and **pH_fct** must be the same.	String
SPH_fct		Numerical (0-100)
OSD_val	Refers to obstacles to roots and impermeable layers. Users must prepare the soil reduction factors based on unique OSD classes. The number of elements in **OSD_val** and **OSD_fct** must be the same. If there are not obstacles to roots and impermeables, set 0 for **OSD_val** and 100 for **OSD_fct**.	Numerical
OSD_fct		Numerical (0-100)

SPR_val	Refers to soil property rating. For any unique soil properties other than vertic can be provided as numerical representations, and its corresponding soil reduction factor.	Numerical
SPR_fct		Numerical (0-100)
For Soil Quality 4 (Oxygen Availability)		
DRG_val	Refers to soil drainage classes. Users must provide soil drainage classes AEZ methodology uses such as: • VP (Very poorly drained) • P (Poorly drained) • I (Imperfectly drained) • MW (Moderately well drained) • W (Well drained) • SE (Somewhat excessively drained) • E (Excessively drained) For each soil drainage class, users also provide the corresponding soil reduction factors.	Categorical
DRG_fct		Numerical (0-100)
SPH_val	Refers to soil phase. Users must prepare the soil reduction factors based on each unique SPH class. The number of elements in **pH_val** and **pH_fct** must be the same.	String
SPH_fct		Numerical (0-100)
For Soil Quality 5 (Presence of Salinity and Sodicity)		
ESP_val	Refers to exchangeable sodium percentage. Users must provide the soil reduction factors based on each unique ESP value. The number of elements in **ESP_val** and **ESP_fct** must be equal.	Numerical
ESP_fct		Numerical (0-100)
EC_val	Refers to electricity conductivity. Users must provide the soil reduction factors based on each unique EC value. The number of elements in **EC_val** and **EC_fct** must be the same.	
EC_fct		Numerical (0-100)
SPH_val	Refers to soil phase. Users must prepare the soil reduction factors based on each unique SPH class. The number of elements in **SPH_val** and **SPH_fct** must be the same.	String
SPH_fct		Numerical (0-100)
For Soil Quality 6 (Presence of Lime and Gypsum)		
CCB_val	Refers to calcium carbonate content percentage. Users must provide the soil reduction factors based on each unique CCB value. The number of elements in **CCB_val** and **CCB_fct** must be the same.	Numerical
CCB_fct		Numerical (0-100)
GYP_val	Refers to gypsum content percentage. Users must provide the soil reduction factors based on each unique GYP value. The numbers of **GYP_val** and **GYP_fct** must be the same.	
GYP_fct		Numerical (0-100)

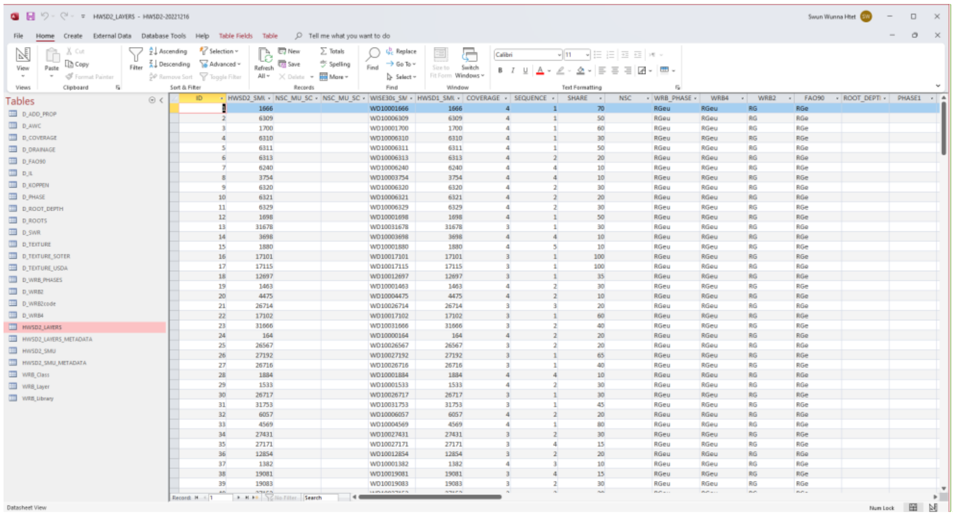
SPH_val	Refers to soil phase. Users must prepare the soil reduction factors based on each unique SPH class. The number of elements in **SPH_val** and **SPH_fct** must be the same.	String
SPH_fct		Numerical (0-100)
For Soil Quality 7 (Workability)		
RSD_val	Refers to rootable soil depth. Users must prepare the soil reduction factors based on each unique RSD value. The number of elements in **RSD_val** and **RSD_fct** must be the same.	
RSD_fct		Numerical (0-100)
GRC_val	Refers to gravel content percentage. Users must provide the soil reduction factors based on each unique GRC value. The number of elements in **GRC_val** and **GRC_fct** must be the same.	Numerical
GRC_fct		Numerical (0-100)
SPH_val	Refers to soil phase. Users must prepare the soil reduction factors based on each unique SPH class. The number of elements in **SPH_val** and **SPH_fct** must be the same.	String
SPH_fct		Numerical (0-100)
TXT_val	Refer to Soil Texture. Users must prepare existing unique soil textures specific soil reduction factors. The number of elements in **TXT_val** and **TXT_fct** must be the same.	String
TXT_fct		Numerical (0-100)
VSP_val	Refers to vertic soil properties. If the SMU has vertic properties, VSP_val must set up 1 and VSP_fct must set up its corresponding soil reduction factor. If there is no vertic properties, VSP_val is 0 and VSP_fct is 0.	Binary (0,1)
VSP_fct		Numerical (0-100)

Note: In each unique row ID in the excel sheet, ***val*** *refers to soil property value, and fct is the LUT/input management soil reduction factors.*

For the users using Harmonized World Soil Database, the newly published database and soil map (v2.0) are available from the mentioned link[10]. The soil parameters are available from Microsoft Database Access, with metadata of each column's definition, respective physical quantity units, together with the detailed description in the HWSD documentation. During the soil parameter preparation for the look-up table python scripts and external excel sheet, it is suggested for the users to explore the documentation, the available database metadata and arrangements and the metadata of Soil Mapping Units existing within the area of interest.

For the reduction factor setting, GAEZ has provided the database open for all individual users to apply for individual crop and specialized level of management settings to refer to in GAEZ Data Portal. (Source: https://gaez.fao.org/pages/glossary). The GAEZ Model Appendices provides the external tables for the crop settings used in AEZ methodology for users to apply.

Figure 3 HWSD v2.0 Microsoft access database



Geographical location and terrain data

PyAEZ requires the elevation and slope maps to be prepared (Table 5) Administrative boundary mask is optional, however, is highly encouraged because it can help minimizing the computational time by considering only area/region of interest.

Table 5 Geographical and terrain data preparation

Data	Data source	Data format
Elevation map	• Global elevation map, or • Own national/regional data	• 2D NumPy array • Unit: metre
Terrain slope map	• Global slope map, or • Own national/regional data	• 2D NumPy array • Unit: percentage
Administrative boundary mask (optional)	• Global mask, or • Own national/regional mask	• 2D NumPy array • Binary: 1 for wanted area, and 0 for unwanted area

Commented [SWH3]: Additional information of HWSD Data preparation.

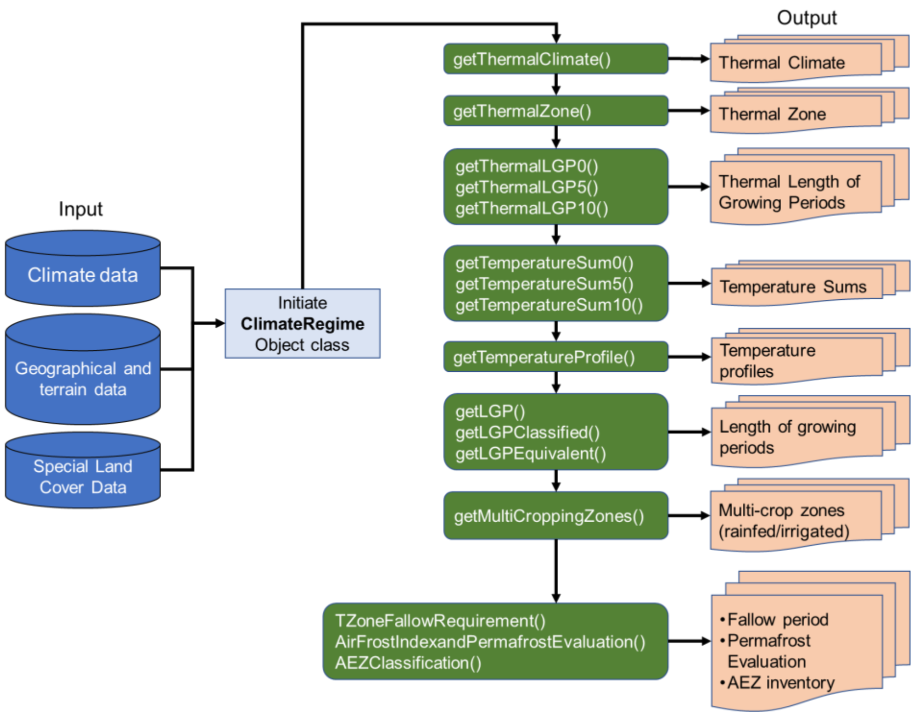
MODULE 1: CLIMATE REGIME

Introduction

This module performs climate data analysis and compiling general agro-climatic indicators. These general agro-climatic indicators summarize climatic profiles in the study area for each grid. Figure 4 shows the overall workflow of Module 1. The key input data for this module is the climatic data, and the geographical and terrain data. This section offers descriptions of the all the functions within Module 1, with example code snippets.

It is advisable to always run this module first, as several agro-climatic indicators output from Module 1 will get feed into Module 2 (Crop Simulation).

Figure 4 Overview of Module 1 (Climate Regime) workflow



Object Class Creation (mandatory)

PyAEZ codes utilizes '*Object-Oriented Programming*' style, meaning that each module has its own *Classes* containing separate attributes and functions. Therefore, it is essential that the necessary object-classes are initiated at the beginning of each module.

For Module 1, the Class that we need is called '`ClimateRegime`', and is imported and initiated as:

```
# Import PyAEZ Module1:ClimateRegime object class
from pyaez import ClimateRegime
clim_reg = ClimateRegime.ClimateRegime()
```

Setting up Geographical and Terrain Data (mandatory)

The next mandatory step after object class creation is to input user's elevation and geographic latitude information into the object class by using this function.

```
# Load geographical location and elevation data into the object class
clim_reg.setLocationTerrainData(lat_min, lat_max, elevation)
```

Function Arguments	
lat_min	a single value corresponding to the minimum latitude (decimal degrees) of the study area
lat_max	a single value corresponding to the maximum latitude (decimal degrees) of the study area
elevation	2D NumPy array, elevation of the study area in metres
Function Returns	
None	

Read the climate data and load into the Class (mandatory)

The third and final mandatory step of preparation is to incorporate all the required climatic datasets into the object class. Depending on the temporal dimension of climatic datasets, user can use either one of the following functions: one for daily datasets and the other for monthly.

```
# Load climate data from NPY files
min_temp = np.load('PATH_TO_FILE') # Continue loading the rest of climate data
# Deal with NaN or inappropriate negative values, for example:
rel_humidity[rel_humidity<0] = 0
short_rad[short_rad<0]=0
wind_speed[wind_speed<0]=0
```

```
# Use the line below if MONTHLY data are used
clim_reg.setMonthlyClimateData(min_temp, max_temp, precipitation, short_rad,
wind_speed, rel_humidity)
# Use the line below if DAILY climate data are used
clim_reg.setDailyClimateData(min_temp, max_temp, precipitation, short_rad,
wind_speed, rel_humidity)
```

Function Arguments	
min_temp	3D NumPy array, daily or monthly minimum temperature (°C)
max_temp	3D NumPy array, daily or monthly maximum temperature (°C)
precipitation	3D NumPy array, daily or monthly total precipitation (mm/day)
short_rad	3D NumPy array, daily or monthly solar radiation (W/m^2)
wind_speed	3D NumPy array, daily or monthly windspeed at 2m elevation (m/s)
rel_humidity	3D NumPy array, daily or monthly relative humidity (percentage decimal, 0–1)

Function Returns
None

Setting Study Area Inputs (optional)

This function is set up as an optional step which set up the mask layer as input which reduces the computation time outside the pixels of considerations.

```
# Set up mask for the study area (country, regional, or local)
clim_reg.setStudyAreaMask(admin_mask, no_data_value=0)
```

Function Arguments	
admin_mask	2D NumPy array, extracted only region of interest (Binary 0/1)
no_data_value	A single value, pixels equal to this value will be omitted during calculation.

Function Returns
None

Thermal Climate

The Thermal Climate function calculates and classifies latitudinal thermal climate, which will be used later in Module 2 for the assessment of potential crops and Land Utilization Types (LUT) presence in each grid cell. It is advisable to use an average of multiple years of temperature data (e.g. 30 years) rather than a single-year data, to obtain better representation of the climate for the study region.

Table 5 describes the classification of thermal climates based on (i) the monthly mean temperature (sea-level adjusted[12]), (ii) the ratios between summer/winter rainfall and the reference evapotranspiration (P/ETo), and (iii) the temperature amplitude as a measure of continentality (i.e. the difference between temperatures of warmest and coldest month) (Fischer *et al.*, 2021).

```
# Classification of rainfall and temperature seasonality into thermal climate
classes
tclimate =clim_reg.getThermalClimate ()
```

Function Arguments	
None	

Function Returns	
tclimate	2D NumPy array (map) of Thermal Climate classification

[12] Sea-level adjusted monthly mean temperature with a fixed lapse rate of 0.55°C/100 metres of elevation

Table 6 Classification of Thermal Climate classes according to rainfall and temperature seasonality

<table>
<tr><th>Climate</th><th>Pixel value</th><th colspan="2">Rainfall and Temperature Seasonality</th></tr>
<tr><td rowspan="2">Tropics
All months with monthly mean sea-level adjusted temperatures > 18°C, and monthly temperature amplitude* < 15^{0}C</td><td>1</td><td>Tropical lowland</td><td>Tropics with actual mean temperatures (Ta) above 20^{0}C</td></tr>
<tr><td>2</td><td>Tropical highland</td><td>Tropics with actual mean temperatures below 20^{0}C</td></tr>
<tr><td rowspan="3">Subtropics
One or more months with monthly mean temperatures, corrected to sea level, below 18^{0}C, but all above 5^{0}C, and 8–12 months above 10^{0}C</td><td>3</td><td>Low rainfall</td><td>Annual rainfall less than 250 mm</td></tr>
<tr><td>4</td><td>Summer rainfall</td><td><u>Northern hemisphere</u>: P/ETo in April-September ≥ P/ETo in October-March.
<u>Southern hemisphere</u>: P/ETo in October-March ≥ P/ETo in April-September</td></tr>
<tr><td>5</td><td>Winter rainfall</td><td><u>Northern hemisphere</u>: P/ETo in April-September ≤ P/ETo in October-March.
<u>Southern hemisphere</u>: P/ETo in October-March ≤ P/ETo in April-September</td></tr>
<tr><td rowspan="3">Temperate
At least one month with monthly mean temperatures, corrected to sea level, below 5^{0}C and four or more months above 10^{0}C</td><td>6</td><td>Oceanic</td><td>Seasonality less than 20^{0}C**</td></tr>
<tr><td>7</td><td>Sub-continental</td><td>Seasonality 20–35^{0}C **</td></tr>
<tr><td>8</td><td>Continental</td><td>Seasonality more than 35^{0}C**</td></tr>
<tr><td rowspan="3">Boreal
At least one month with monthly mean temperatures, corrected to sea level, below 5^{0}C and 1–3 months above 10^{0}C</td><td>9</td><td>Oceanic</td><td>Seasonality less than 20^{0}C**</td></tr>
<tr><td>10</td><td>Sub-continental</td><td>Seasonality 20–35^{0}C **</td></tr>
<tr><td>11</td><td>Continental</td><td>Seasonality more than 35^{0}C**</td></tr>
<tr><td>Arctic</td><td>12</td><td>Arctic</td><td>All months with monthly mean temperatures,</td></tr>
</table>

			corrected to sea level, below 10°C

** Monthly temperature amplitude = monthly maximum temperature – monthly minimum temperature*
*** Seasonality = the difference in mean temperature of the warmest and coldest month*

Thermal zones

The thermal zone is classified based on actual temperature which reflects on the temperature regimes of major thermal climates (Table 7).

```
# Classification of thermal zone classes
tzone = clim_reg.getThermalZone()
```

Function Arguments	
None	
Function Returns	
tzone	2D NumPy array (map) of Thermal Zones classification

Table 7 Classification of Thermal Zone classes according to rainfall and temperature seasonality

<table>
<tr><th>Climate</th><th>Pixel value</th><th colspan="2">Thermal zones</th></tr>
<tr><td rowspan="2">Tropics
All months with monthly mean sea-level adjusted temperatures > 18°C, and monthly temperature amplitude* < 15°C</td><td>1</td><td>Warm</td><td>Tropics with annual mean temperature above 20^{0}C</td></tr>
<tr><td>2</td><td>Cool/Cold/Very cold</td><td>Tropics with annual mean temperatures below 20^{0}C</td></tr>
<tr><td rowspan="4">Subtropics
One or more months with monthly mean temperatures, corrected to sea level, below 18^{0}C, but all above 5^{0}C, and 8–12 months above 10^{0}C</td><td>3</td><td><i>Warm/Moderately cool</i></td><td>Annual mean temperature above 20^{0}C</td></tr>
<tr><td>4</td><td>Cool</td><td>At least one month with monthly mean temperatures below 5^{0}C and 4 or more months above 10^{0}C</td></tr>
<tr><td>5</td><td>Cold</td><td>At least one month with monthly mean temperatures below 5^{0}C and 1-3 months above 10^{0}C</td></tr>
<tr><td>6</td><td>Very cold</td><td>All months with monthly mean</td></tr>
</table>

			temperatures below 10^0C.
Temperate At least one month with monthly mean temperatures, corrected to sea level, below 5^0C and four or more months above 10^0C	7	*Cool*	At least one month with monthly mean temperatures below 5^0C and 4 or more months above 10^0C
	8	*Cold*	At least one month with monthly mean temperatures below 5^0C and 1-3 months above 10^0C
	9	*Very cold*	All months with monthly mean temperatures below 10^0C
Boreal At least one month with monthly mean temperatures, corrected to sea level, below 5^0C and 1-3 months above 10^0C	10	*Cold*	At least one month with monthly mean temperatures below 5^0C and 1-3 months above 10^0C
	11	*Very cold*	All months with monthly mean temperatures below 10^0C
Arctic	12	*Arctic*	All months with monthly mean temperatures, corrected to sea level, below 10°C

*Monthly temperature amplitude = monthly maximum temperature – monthly minimum temperature

Thermal Length of Growing Period (LGP_t)

The thermal length of growing period (LGP_t) is defined as the number of days in a year during which the daily mean temperature (Ta) is conductive to crop growth and development. PyAEZ utilizes the AEZ three standard temperature thresholds for LGPt:

i. Periods with Ta > 0°C (LGP_{t0}).

ii. Periods with Ta > 5°C (LGP_{t5}) – the period conductive to plant growth and development.

iii. Periods, and Ta > 10°C (LGP_{t10}) – a proxy for the period of low risks for late and early frost occurrences and termed frost-free period.

```
# Calculate Thermal Length of Growing Period (LGPt)
# 3 temperature thresholds
# LGPt>0 degC
lgpt0 = clim_reg.getThermalLGP0()
# LGPt>5 degC
lgpt5 = clim_reg.getThermalLGP5()
# LGPt>10 degC
lgpt10 = clim_reg.getThermalLGP10()
```

Function Arguments	
None	
Function Returns	
lgpt0	2D NumPy arrays [days]
lgpt5	2D NumPy arrays [days]
lgpt10	2D NumPy arrays [days]

Temperature summations (TS)

Temperature summation corresponds to the accumulated temperature which represent the crop-/LUT-specific heat requirements (Fischer *et al.,* 2021).

Reference temperature sums (TS) are calculated for each grid-cell by accumulative daily average temperature (Ta) for days when Ta is above the thresholds as follows: (i) 0°C, (ii) 5°C, and (iii) and 10°C.

```
# Calculate temperature summation at 3 temperature thresholds
# Tsum>0 degC
tsum0 = clim_reg.getTemperatureSum0()
# Tsum>5 degC
tsum5 = clim_reg.getTemperatureSum5()
# Tsum>10 degC
tsum10 = clim_reg.TemperatureSum10()
```

Function Arguments
None

Function Returns	
tsum0	2D NumPy arrays [°C]
tsum5	2D NumPy arrays [°C]
tsum10	2D NumPy arrays [°C]

Temperature profiles

Temperature profiles (Table 8) can be classified into 9 classes of different daily 'temperature ranges' between Ta<-5°C to Ta>30°C. This classification uses 5°C intervals as well as distinguishes the increasing and decreasing temperature trends within a year (Fischer *et al*., 2021). The output from this classification will be used in Module 2 (Crop Simulation), where these profiles are matched with crop-specific temperature profile requirements to assess the crop-growth suitability for any specific locations.

```
# Classification of temperature ranges for temperature profile
tprofile = clim_reg.getTemperatureProfile()
```

Function Arguments	
None	

Function Returns	
tprofile	18 2D NumPy arrays [A1-A9, B1-B9] correspond to each Temperature Profile class [days]

Table 8 Temperature profile classes

Mean daily temperature (°C)	Temperature trend	
	Increasing	**Decreasing**
30	A1	B1
25–30	A2	B2
20–25	A3	B3
15–20	A4	B4
10–15	A5	B5
5–10	A6	B6
0–5	A7	B7
-5–0	A8	B8

< -5	A9	B9

Length of Growing Period (LGP)

The length of growing period (LGP) is defined as the number of days during the year when the temperature regime and moisture supply are conductive to crop growth and development. LGP, therefore, acts as an agro-climatic indicator of the potential productivity of an area of land.

Reference evapotranspiration (ETo)

The reference evapotranspiration (ETo) represents evapotranspiration from a defined reference surface, which closely resembles an extensive surface of green, well-watered grass of uniform height (12 cm), actively growing and completely shading the ground. GAEZ calculates ETo from the attributes in the climate database for each grid-cell according to the Penman-Monteith equation (Allen *et al*., 1998; Monteith, 1965, 1981; Doorenbos & Pruitt, 1977). A description of the implementation of the Penmann-Monteith equations is provided in Appendix 3-1 of Fischer *et al*., (2021).

Maximum evapotranspiration (ETm)

In Module 1, the calculation of maximum evapotranspiration (ETm) for a 'reference crop' assumes that sufficient water is available for uptake in the rooting zone. The value of ETm is related to ETo through applying crop coefficients for water requirement (Kc), reflecting phenological development and leaf area. The Kc values are crop- and climate-specific. They vary generally between 0.3–0.5 at initial crop stages (emergence) to 1.0–1.2 at reproductive stages. PyAEZ utilizes the 'reference crop' whose Kc values depend on the thermal characteristics of a grid cell, as described in Table 8.

$$ETm = Kc \times ETo$$

Actual evapotranspiration (ETa)

The actual uptake of water by the 'reference' crop is characterized by the actual evapotranspiration (ETa, mm/day) resulting in the daily calculations of the reference crop water balance. The calculation of ETa differentiates two possible cases depending on the availability of water for plant extraction:

i. Adequate soil water availability (ETa=ETm), and

ii. Limiting soil water availability (ETa<ETm).

<u>Water balance calculation</u>

The calculation of Eta involves daily soil water balance (Wb), which is defined as the volume of water available for plant uptake. The water balance, Wb, accounts for the accumulation of daily water inflow from precipitation (P), snowmelt (Sm), and outflow from the actual evapotranspiration (Eta), and excess water lost due to runoff and deep percolation (amount of water that exceeds the upper limit of water available to plants, Wx). For the 'j' day of the year, the daily water balance is calculated as:

$$Wb_j = min\ (Wb_{j-1} + Sm_j + P_j - ETa_j, Wx)$$

The upper limit Wx of water available to plants is the product of the available soil water (Sa) and rooting depth (D),

$$Wx = Sa \times D$$

The threshold of readily available soil moisture Wr is, in turn, calculated from Wx and the soil moisture depletion fraction (p),

$$Wr = Wx \times (1 - p)$$

Snow balance calculation

In seasonally cold climates the calculation of a snow balance (Sb, mm) affects the water balance procedure outlined above. The snow balance increases when precipitation falls as snow and decreases with snowmelt and snow sublimation. Precipitation (P) is assumed to fall as snow (P^{snow}) when maximum temperature (Tx) is below a certain temperature threshold (Ts).

The snowmelt (Sm) is calculated as a function of daily maximum temperature, the snow melt parameter (δ) and depends on the previously accumulated snow balance. The snow melt factor δ is set to 5.5 mm/°C

$$Sm = min\ (\delta\ \times (Tx - Ts), Sb)$$

Further details of the two possible cases of ETa calculation are as follows:

ETa for adequate soil water availability

A condition of 'adequate soil moisture availability' is defined when:

i. Daily precipitation (P) is greater or equal to ETm, and/or
ii. Combination of P and the difference between Wb and the readily-available-water threshold Wr is greater than Etm

$$ETa = ETm, \qquad for\ \{P \geq ETm \quad P + (Wb - Wr) > Etm$$

ETa for limited water availability

When the soil water is limiting, then ETa falls short of ETm. In this case, ETa is calculated as a fraction ρ of ETm, where,

$$\rho = Wb/Wr$$

The Eta is then calculated as

$$ETa = P + \rho \times ETm$$

This procedure assumes rainfall is immediately available to plants on the day of precipitation, prior to replenishing soil moisture.

LGP calculation

LGP refers to the number of days when average daily temperature is above 5^0C (LGP_{t5}) and ETa of this reference crop exceeds a specified fraction of ETm. In the current GAEZ parameterization, LGP days are considered when ETa≥0.4×ETm, which aims to capture periods when sufficient soil moisture is available that would allow the establishment of the reference crop.

$$LGP = total\ number\ of\ days\ when\ \ ETa/ETm \geq\ 0.4$$

LGP Equivalent

Reference LGPs account for both temperature and soil moisture conditions and do not necessarily account for significant differences in wetness conditions especially within long LGPs (>225 days), for a better reflection of wetness conditions, so-called equivalent LGPs are used. Equivalent LGP is defined based on regression analysis of the reference LGP and the humidity index P/ETo as follows.

A quadratic polynomial is used to express the relationship between the number of growing period days and the annual humidity index. Parameters were estimated using data of all grid cells with essentially year-round temperature growing periods, i.e., with LGP_{t5} = 365.

$$LGP_{eq} = \begin{cases} \{14.0 + 293.66 \times (P/ETo) - 61.25 \times \left(\frac{P}{ETo}\right)^2 & , when \left(\frac{P}{ETo}\right) \leq 2.4 \\ 366, & when \left(\frac{P}{ETo}\right) > 2.4 \end{cases}$$

The equivalent LGP is used in the assessment of agro-climatic constraints, which relate environmental wetness with the occurrences of pest and diseases and workability constraints for harvesting conditions and for high moisture content of crop produce at harvest time.

In PyAEZ, the LGP, LGP classification, and LGP Equivalent are obtained through the following function,

```
# Length of Growing Period (LGP)
lgp = clim_reg.getLGP(Sa=100, D=1)
# Classification of LGP
lgp_class = clim_reg.getLGPClassified(lgp)
# LGP Equivalent
lgp_equv = clim_reg.getLGPEquivalent()
```

Function Arguments	
Sa	A single value or A 2D NumPy array, corresponding to available soil moisture holding capacity (mm/m). Usually, this value varies with soil texture. Hence, Sa can be provided as single value for entire area or 2D NumPy array that represent variation of soil moisture holding capacity depending on soil texture. Default value is 100 mm/m. This is an optional argument
D	A single value, corresponding to corresponding rooting depth in meters. Default value is 1. This is an optional argument.
Function Returns	
lgp	2D NumPy arrays of LGP [days]

Table 9 Kc values used in Module 1 for the calculation of the maximum evapotranspiration (Etm)

Daily temperature condition	Remarks	Kc
Areas with year-round temperature growing period – LGPt5=365 days		
Daily Ta ≥5°C; LGP_{t5}=365 days	In areas with year-round LGP_{t5}, the Kc value stays at 1	1.0
Areas with dormancy period or cold break – LGPt5 less than 365 days		
Daily Ta ≤ 0°C; Tmax < 0°C	Precipitation falls as snow and is added to snow bucket	0.0
Daily Ta ≤ 0°C; Tmax≥0°C	Snowmelt takes place (water balance = precipitation + snow melt); minor evapotranspiration	0.1
0°C < Daily Ta < 5°C; Ta trend upward	Some biological activities before the start of the growing period	0.2
Daily Ta ≥ 5°C; LGPt5<365 days; Case 1	Kc used for the days prior the start of the growing period	0.5
Daily Ta ≥5°C; LGPt5<365 days; Case 2	Kc increases from 0.5 to 1.0 during the first month of LGP	0.5–1.0
Daily Ta ≥5°C; LGPt5<365 days; Case 1	Kc=1 until the daily Ta falls below 5°C	1.0
0°C < Daily Ta < 5°C; Ta trend downward	Reduced biological activities before dormancy	0.2

Multiple cropping zones classification

Multiple cropping zones classification (Table 9) is an additional agro-climatic indicator, which relates to the possibility of cultivating multiple sequential crops under rain-fed and irrigated conditions.

The PyAEZ's core modules perform calculation for single cropping systems. Additionally, several potential multiple cropping zones have been defined through matching the growth cycle with the temperature requirements based on Thermal Climate, Length of Growing period, Thermal Growing Period (LGP_{t0} and LGP_{t10}), and the accumulated temperature summations ($Tsum_{t0}$, $Tsum_{t10}$). For more details on the multiple cropping zones classification please refer to the *GAEZ v4 Model Documentation* (Fischer *et al*., 2021).

```
# Multiple Cropping Zones classification
multi_crop_zone = clim_reg.getMultiCroppingZones(tclimate, lgp, lgpt5, lgpt10,
tsum0,tsum10)
```

Function Arguments	
tclimate	2D NumPy array, Thermal Climate classes
lgp	2D NumPy array, Length of Growing Period
lgpt5	2D NumPy array, Thermal LGP of Ta>5˚C
lgpt10	2D NumPy array, Thermal LGP of Ta>10˚C
tsum0	2D NumPy array, Temperature summation for Ta≥0˚C
tsum10	2D NumPy array, Temperature summation for Ta≥10˚C
Function Returns	
multi_crop_zone	Python List of 2D NumPy arrays, as [multi_crop_rainfed, multi_crop_irrigated].

Table 10 Delineation of multiple cropping zones

Pixel values	Zone	Description
1	A	Zone of no cropping (too cold or too dry for rain-fed crops)
2	B	Zone of single cropping
3	C	Zone of limited double cropping (relay cropping; single wetland rice may be possible)
4	D	Zone of double cropping (note, in Zone D sequential double cropping including wetland rice is not possible)
5	E	Zone of double cropping with rice (sequential double cropping with one wetland rice crop is possible in Zone E)
6	F	Zone of double rice cropping or limited triple cropping (may partly involve relay cropping. A third crop is not possible in case of two wetland rice crops)
7	G	Zone of triple cropping (sequential cropping of three short-cycle crops; two wetland rice crops are possible in Zone G)
8	H	Zone of triple rice cropping (sequential cropping of three wetland rice crops is possible)

Fallow period requirements

Fallow is an agricultural technique that consists of not sowing the arable land during one or more growing seasons. In AEZ framework, the fallow factors have been established by main crop groups and environmental conditions. The crop groups include cereals, legumes, roots and tubers, and a miscellaneous group consisting of long-term annuals/perennials. The fallow factors are expressed as percentage of time during the fallow-cropping cycle the land must be under fallow. PyAEZ determines the fallow requirements using Thermal Zones. For further information on the fallow period requirement, please refer to Appendix 6–10 of the *GAEZ v4 Model Documentation* (Fischer *et al*., 2021).

```
# Fallow period requirements
fallow = clim_reg.TZoneFallowRequirement(tzone)
```

Function Arguments	
tzone	2D NumPy array, corresponding to Thermal Zone
Function Returns	
fallow	2D NumPy array, corresponding to Thermal Zone for Fallow Requirements

Permafrost evaluation

Occurrence of continuous or discontinuous permafrost conditions are used in the suitability assessment. Permafrost areas are characterized by sub-soil at or below the freezing point for two or more years. In this section, PyAEZ utilizes the air frost index (FI) which is used to characterize climate-derived permafrost condition into 4 classes: (i) Continuous permafrost; (ii) Discontinuous permafrost; (iii) Sporadic permafrost; and (iv) No permafrost. For detailed calculations for air frost index please refer to Chapter 3 of the *GAEZ v4 Model Documentation* (Fischer *et al*., 2021).

```
# Permafrost Evaluation
permafrost = clim_reg.AirFrostIndexandPermafrostEvaluation()
```

Function Arguments	
None	
Function Returns	
permafrost	Python List of 2D NumPy arrays, as [frost index, permafrost class]

Agro-ecological Zones classification

The agro-ecological zones (AEZ) methodology provides a framework for establishing a spatial inventory of land resources compiled from global/national environmental data sets and assembled to quantify multiple spatial characteristics required for the assessments of land productivity under location-specific agro-ecological conditions.

The inventory combines spatial layers of thermal and moisture regimes with broad categories of soil/terrain qualities. It also indicates locations of areas with irrigated soils and shows land with severely limiting bio-physical constraints including very cold and very dry (desert) areas as well as areas with very steep terrain or very poor soil/terrain conditions. For further information on the classification criteria, please refer to Chapter 10 of the *GAEZ v4 Model Documentation* (Fischer *et al.*, 2021).

```
# AEZ classification
aez_class = clim_reg.AEZClassification(tclimate, lgp, lgp_equv, lgpt_5,
soil_terrain_lulc, permafrost)
```

Function Arguments	
tclimate	2D NumPy array, Thermal Climate classes
lgp	2D NumPy array, Length of Growing Period
lgp_equv	2D NumPy array, LGP Equivalent
lgpt5	2D NumPy array, Thermal LGP of Ta>5˚C
soil_terrain_lulc	2D NumPy array, soil/terrain/special land cover classes (8 classes) ● 1: Dominantly very steep terrain ● 2: Dominantly hydromorphic soil ● 3: No or few soil/terrain limitations ● 4: Moderate soil/terrain limitations ● 5: Severe soil/terrain limitations ● 6: Irrigated soils ● 7: Water ● 8: Built-up/Artificial
permafrost	2D NumPy array, Permafrost classes
Function Returns	
aez_class	2D NumPy array, 57 classes of AEZ

MODULE 2: CROP SIMULATION

Introduction

This key module simulates all the possible crop cycles to find the best crop cycle that produces maximum yield for a particular grid (Module 2 overview is shown in Figure 5). During the simulation process for each grid, 365 crop cycle simulations are performed. Each simulation corresponds to cycles that start from each day of the year (starting from Julian date 1 to Julian date 365). Similarly, this process is performed by the program for each grid in the study area.

Schematic representation of this process is shown in Figure 6. The attainable yields under irrigated and rain-fed conditions, during each crop cycle, are calculated with the help of several deterministic and empirical models as follows.

- **Total biomass** (de Wit, 1965): This model calculates total biomass produced by photosynthesis activities of plants under radiation condition of each grid. For more detailed calculations, refer to Chapter 4 of the *GAEZ v4 Model Documentation* (Fischer *et al*., 2021).
- **Crop yield from total biomass**: Crop yield is obtained as a portion of useful harvest from the total biomass. This portion is defined by an index call Harvest Index (HI). Harvest index is defined as the amount of useful harvest divided by the total above ground biomass. For more detailed calculations, refer to Chapter 4 of the *GAEZ v4 Model Documentation* (Fischer *et al*., 2021).
- **Effects of water limitation on the crop yield**: This component is carried out for the rain-fed yield calculation only. In the case of irrigated conditions, this component is abandoned, as we are assuming that the water is not a limiting factor for crop growth. To address the water limitation on the crop yield, two major models are considered:
    - Reference Evapotranspiration – the Penman-Monteith equation (Doorenbos & Pruitt, 1977). A description of the implementation of the Penmann-Monteith equations is provided in Appendix 3-1 of Fischer *et al.* (2021)
    - Water balance calculation, together with applying the yield reduction factors based on water limitation (Smith, 1992)
- **Effects of temperature during crop cycle** and screening of crop cycles based on temperature requirements (termed *Thermal Screening*).

Similar to Module 1, we have to import and initiate the Class for the Crop Simulation module,

```
# import Module 2 Class
from pyaez import CropSimulation
# create an instance – initiate the Class
aez = CropSimulation.CropSimulation()
```

Figure 5 Overview of Module 2 (Crop Simulation) workflow

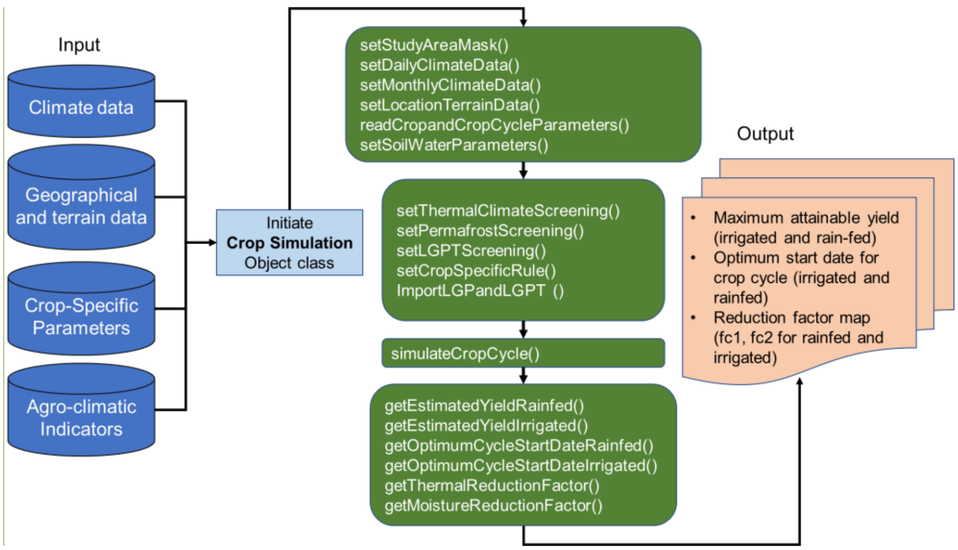
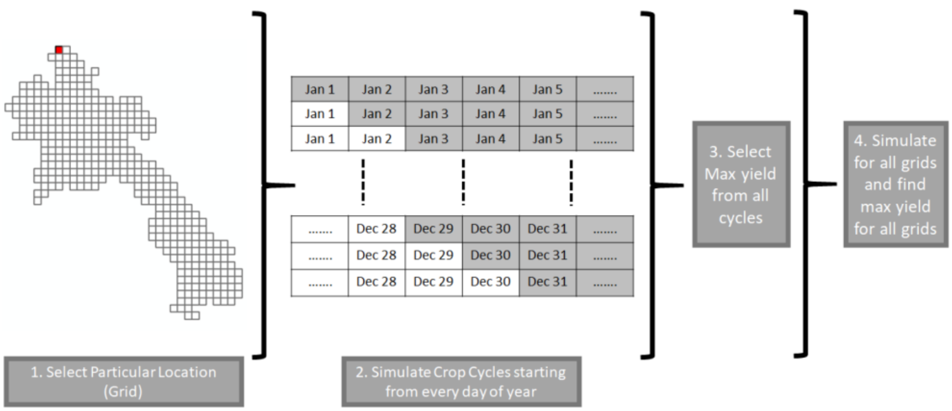


Figure 6 Overview of Crop Simulation Routine



Commented [SWH4]: Updated overall workflow M2.

Setting up inputs for Module 2

Geographical and terrain input (Mandatory)

```
# Load geographical location and elevation data into the object class
aez.setLocationTerrainData(lat_min, lat_max, elevation)
```

Function Arguments	
`lat_min`	a single value corresponding to the minimum latitude (decimal degrees) of the study area
`lat_max`	a single value corresponding to the maximum latitude (decimal degrees) of the study area
`elevation`	2D NumPy array, elevation of the study area in metres
Function Returns	
None	

Climate data input (Mandatory)

First, we have to read and load the climate data into Module 2 Class before proceeding with any calculations. Climate data can be imported either monthly or daily. If monthly climate data is imported, the monthly values are interpolated into daily.

```
# Use the line below if MONTHLY data are used
aez.setMonthlyClimateData (min_temp, max_temp, precipitation, short_rad,
wind_speed, rel_humidity)
# Use the line below if DAILY data are used
aez.setDailyClimateData (min_temp, max_temp, precipitation, short_rad,
wind_speed, rel_humidity)
```

Function Arguments	
min_temp	3D NumPy array, daily or monthly minimum temperature (°C)
max_temp	3D NumPy array, daily or monthly maximum temperature (°C)
precipitation	3D NumPy array, daily or monthly total precipitation (mm/day)
short_rad	3D NumPy array, daily or monthly solar radiation (W/m2)
wind_speed	3D NumPy array, daily or monthly windspeed at 2m elevation (m/s)
rel_humidity	3D NumPy array, daily or monthly relative humidity (percentage decimal, 0–1)
Function Returns	
None	

Setting Study Area Inputs (optional)

This function is set up as an optional step which set up the mask layer as input which reduces the computation time outside the pixels of considerations.

```
# Set up mask for the study area (country, regional, or local)
clim_reg.setStudyAreaMask(admin_mask, no_data_value=0)
```

Function Arguments	
admin_mask	2D NumPy array, extracted only region of interest (Binary 0/1)
no_data_value	A single value, pixels equal to this value will be omitted during calculation
Function Returns	
None	

Crop/crop cycle and TSUM parameters input (Mandatory)

This function allows to set up crop parameter, crop cycle parameters and TSUM screening parameters from the external excel sheet in xlsx format, essential for PyAEZ crop simulation setting.

```
# Call the function of reading the corresponding crop parameterization
aez.readCropandCropCycleParameters(file_path , crop_name)
```

Function Arguments	
file_path	The file path where all crop/crop cycle parameters and TSUM screening thresholds are provided. (String)
crop_name	The unique ID of crop from excel sheet for crop simulation. (String)
Function Returns	
None	

The excel sheet for this function must set up in such detailed description for each column names as below. If users desire to simulate perennial crops, it is mandatory to set up parameters concerning with perennials in excel sheet. For TSUM screening, for any crop types, it is necessary to provide all six thresholds. If either one of the thresholds is not provided, TSUM screening will NOT be activated.

Table 11 Detailed excel setting for crop/crop cycle and TSUM screening parameters

Abbreviation	Description	Data Type

Commented [SWH7]: Updated function description for crop/crop cycle parameters and TSUM screening setting. Will replace "Crop parameters input, and crop cycle parameters input" functions.

Crop_name	Unique name of the crop/LUT		String
input_level	Input management level defined by AEZ. Must be either 'low', 'intermediate' or 'high' level.		String
HI	Harvest Index		Float
LAI	Leaf Area Index		Float
legume	Is this crop legume? No = 0, Yes = 1		binary
adaptability	Crop adaptability class group of the crop. Value must be either one of 1, 2, 3 or 4, referring to the adaptability class		Integer
cycle_len	Length of the crop cycle (Unit = Days)		Integer
D1	Rooting depth at the beginning of the crop cycle (Unit = meters)	***Note.*** D1 and D2 can be the same value. If affirmative, interpolation will not be applied for each day within the length of crop cycle.	Float/integer
D2	Rooting depth after maturity		
stage_per_1	Percentages of each four stages of a crop cycle. All percentages summation must be 100.	Initial stage (d1)	Float/integer
stage_per_2		Vegetative stage (d2)	
stage_per_3		Reproductive stage (d3)	
stage_per_4		Maturation stage (d4)	
kc_0	Crop water requirements for three stages of a crop cycle	Initial stage (d1)	Float/integer
kc_1		Vegetative stage (d2)	
kc_2		Maturation stage (d4)	
kc_all	A single value of crop water requirement for the entire growth cycle		Float/integer
yloss_f0	Yield loss factors of each three stages of a crop cycle	Initial stage (d1)	Float/integer
yloss_f1		Vegetative stage (d2)	
yloss_f2		Vegetative stage (d3)	
yloss_f3		Maturation stage (d4)	
yloss_f_all	A single value of yield loss factor for the entire growth cycle		Float/integer
min_temp	Minimum temperature requirement for a crop/LUT to grow.		Float/integer
annual/perennial flag	A single value. Annual = 0, Perennial = 1	***Note.*** If users are simulating **perennial** crops, providing the corresponding	Binary (0,1)
aLAI	αLAI		Float/integer

Commented [SW6]: Missed parameter description.

Commented [SW5]: Missed parameter to describe.

bLAI	βLAI	parameterizations is mandatory. These values will be utilized for the newly developed perennial adjustment in crop simulation. For annual crops, setting as annual flag, and the rest of the parameters as “nan” can be done, so adjustment is not required.	
aHI	αHI		
bHI	βHI		
min_cycle_len	Minimum cycle length of the crop (Unit = Days)		Integer
max_cycle_len	Maximum cycle length of the crop (Unit = Days)		
height	Plant height (Unit = meters)		Float/Integer
LnS	TSUM threshold for lower boundary of **Not Suitable**	***Note.*** These threshold points refer to the requirements for Temperature Summation (TSUM) screening progress to the crop simulation. This optional thermal screening function is activated when all six thresholds are provided. If one of the values is missing, the TSUM screening will not be applied. If users don’t want to apply TSUM screening, provide “nan” value to all six variables.	Float/integer
LsO	TSUM threshold for lower boundary of **Sub-Optimal** range		
LO	TSUM threshold for lower boundary of **Optimal** range		
HO	TSUM threshold for upper boundary of **Optimal** range		
HsO	TSUM threshold for upper boundary of **Sub-Optimal** range		
HnS	TSUM threshold for upper boundary of **Not Suitable**		

Soil water parameter input

This function allow user to set up the parameters related to the soil water storage.

```
# Set soil water parameter
aez.setSoilWaterParameters(Sa, pc)
```

Function Arguments	
Sa	A single value or a 2D NumPy array, corresponding to available soil moisture holding capacity [mm/m]. Usually, this value varies with soil texture; thus, Sa can

	be provided as single value for entire area or 2D NumPy array that represents variation of soil moisture holding capacity depending on soil texture.
pc	A single value between 0 and 1, corresponding to soil water depletion fraction below which ETa < ETo.

Function Returns
None

Commented [SWH8]: Missing documentation. Now filled.

Thermal screening input

The functions in this section will screen the suitability of grid-cells for the possible presence of individual LUTs. The crops' temperature requirements will be matched with the prevailing thermal conditions (Thermal Regime characteristics calculated in Module 1).

Thermal climate

PyAEZ 's screening of crop/LUTs about thermal climate results in a 'yes/no' filter for further calculations, with user-defined list of thermal climate classes considered for the specific crop as "not suitable".

Thermal growing period

Growth cycle lengths of crop/LUTs are matched with LGP_{t5}. The result of the matching provides optimum match when the growth cycle can generously be accommodated within LGP_{t5}. Otherwise, the match is considered not suitable.

Permafrost Screening

This new procedure is fully referred to GAEZ v4 documentation stating that areas considered as continuous or discontinuous permafrost classes are considered not suitable (Fischer et al., 2021). In PyAEZ, permafrost screening results in 'yes/no' filter for further calculations using permafrost class output from Module I (See Permafrost evaluation).

Commented [SWH9]: New Thermal Screening: Permafrost screening.

Accumulated temperature sum (TSUM screening)

The matching of the individual crop LUT heat unit requirements with the prevailing temperature sum is the purpose of temperature summation screening. TSUM is evaluated from base temperature of 0 °C for each individual cycle length duration. In version 2.1.0, the new algorithm for TSUM screening is introduced which is implemented with different inputs.

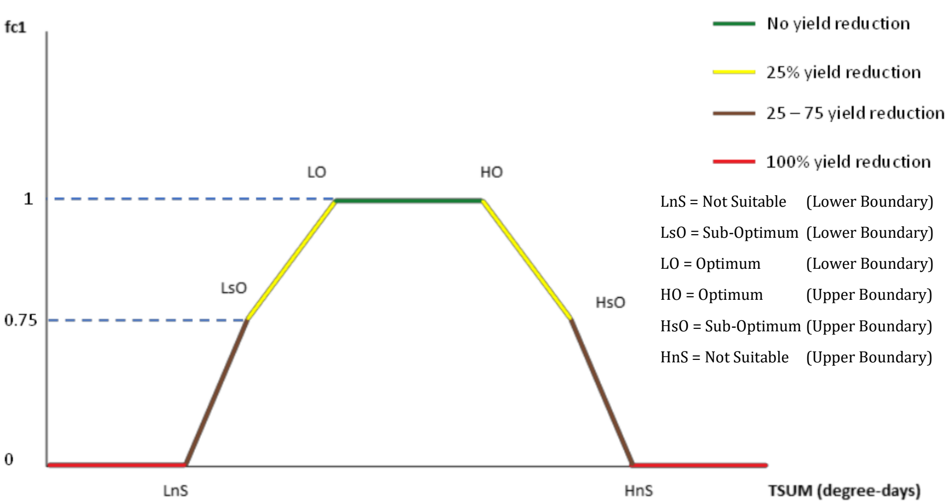
TSUM screening works to evaluate three conditions, each deciding the TSUM suitability termed as: **"Optimum", "Sub-optimum"** and **"Not-suitable"**. **Optimum** condition requires no reduction factor to the calculated yield, while the rest of two conditions calculates the TSUM related reduction factor. Each condition has upper and lower boundaries (See Figure 6); defined as threshold points for the users to provide as below:

- LO : Lower boundary of optimum threshold
- HO : Upper boundary of optimum threshold
- LsO : Lower boundary of sub-optimum threshold

- HsO : Upper boundary of sub-optimum threshold
- LnS : Lower boundary of not-suitable threshold
- HnS : Upper boundary of not-suitable threshold

Any TSUM values within the optimum threshold range (green line) are considered as inducive to plant growth; thus, no reduction factor will be applied. TSUM values between sub-optimum threshold has two components (illustrated as yellow and blue lines). Within yellow range (between LsO – LO or HO – HsO), reduction factor is calculated up to maximum 25%, obtained by linear interpolation between two end points of user-defined thresholds. Within brown range (between LnS – LsO or HsO – HnS), substantial amount of reduction factor up to maximum of 75% are obtained, from linear interpolation. Any TSUM lower than LnS threshold or beyond HnS threshold are considered as temperature too high or too low for plant growth, giving rise to 100% reduction to calculated yield.

Figure 7 New Implementation for TSUM screening



Commented [SWH10]: Updated Thermal Screening: TSUM screening description.

Crop-Specific Rule Screening

Commented [SWH12]: Substituted Thermal Screening: Crop-specific rule screening.

The previous temperature profile screening in version 2.1.0 is substituted with renamed routine called "Crop-Specific Rule". This screening specifies conditions for crop cycle duration in terms of classes of daily mean temperature in 5°C intervals (See Temperature profiles). For activation, users require providing the external excel sheet to provide with corresponding column names. Similar to TSUM screening, users need to provide thresholds for three conditions: **"Optimal", "Sub-Optimal" and "Not-Suitable"**. The unique combination of several temperature profile classes is to be provided as expressions as unique rules for a particular crop type. In PyAEZ, notations for temperature profile classes, For all feasible crop calendars within the LGP (rain-fed) or within the year (irrigated), the temperature profile conditions are tested against optimum and suboptimum crop temperature profile requirements and in each case an "optimum" or "not suitable" match is established. Detailed calculations are fully referred to GAEZ v4 Model Documentation.

```
# Set parameters for Thermal Screening
aez.setThermalClimateScreening(tclimate,no_tclimate)
aez.setLGPTScreening(no_lgpt, optm_lgpt)
aez.setPermafrostScreening(permafrost_class)
aez.setCropSpecificRule(file_path, crop_name)
```

Commented [SWH11]: I might have the change the function name not to be confused with documentation.

Function Arguments	
tclimate	2D NumPy array, corresponding to thermal climate (an output of Module 1)
no_tclimate	A numerical list, corresponding to pixel values of "not suitable" thermal climate zones
no_lgpt	3-elements numerical list, "not suitable" 3 LGPt conditions (as in Module 1)
optm_lgpt	3-elements numerical list, "optimum" 3 LGPt conditions (as in Module 1)
permafrost_class	2D numpy array, corresponding to permafrost classes (as in Module 1)
file_path	The file path where the crop-specific rules are provided (String)
crop_name	The unique ID of crop name to apply crop-specific rule (String)
Function Returns	
None	

The crop specific rule excel-sheet setting is described in Table 12.

Table 12 Detailed excel setting for crop-specific rule setting

Commented [SWH13]: Additional info for crop-specific rule setting for M2.

Abbreviation	Description	Data Type
Crop	Unique crop name	String
Constraint	Expression of different temperature profile classes combination For instance, L61 + L4a. User must use the mathematical symbols used in python (+, -, **, *, /)	String
Type	Constraint type. Must be [≥, ≤, ==].	String
Optimal	The threshold value point for optimal condition.	Numerical

Sub-Optimal	The threshold value point for sub-optimal condition.	Numerical
Not-Suitable	The threshold value point for not-suitable condition.	Numerical

Minimum Cycle Length Checking (New Logic)

For each individual pixel, the algorithm will check whether the location has enough growing days for supporting LUT/crop specific cycle length. This checking logic is done separately for rainfed and irrigated condition, also for perennials and non-perennials as well. If the number of growing days is insufficient (i.e., less than minimum cycle length), the yield simulation will not proceed and iterates to next pixel.

For annuals, LGP_{t5} is used to compare with the minimum cycle length for irrigated condition, while LGP is used to compare.

For perennials, an extensive flow of condition checking is implemented, and cycle will be adjusted based on the concept of effective cycle length (See Perennial crop adjustments). After the adjustment, the effective cycle length is compared with minimum cycle length. If condition is met, the next yield estimation will continue, otherwise, the cycle will be skipped.

Perennial crop adjustments

Commented [SWH14]: Updated documentation for Perennial crop adjustments, and user guides.

If a perennial crop is introduced (based on excel setting for perennial flag as positive), PyAEZ will perform adjustment on the Leaf Area Index (LAI) and the Harvest Index (HI) based on the effective growing period. PyAEZ v2.2 have a routine for the perennial crop simulations; integrated with minimum cycle length checking logic. In additional to Leaf Area Index and Harvest Index adjustment to perennial crops, new conditional logics are added to the simulation as below. The adjusted HI, LAI and cycle length will continue to use for next individual cycle's critical calculation of biomass estimation.

- **Crop cycle length adjustment with temperature growing periods (LGPt) and LGP**
  - **Irrigated Condition**

    Irrigated perennial crops having minimum temperature (from excel sheet) less than or equal to 8℃ will have their crop cycle length checked with LGPt5. If LGPt5 is less than the cycle length, LGPt5 will be chosen as effective cycle length (CYCeff) for the simulation. Otherwise, CYCeff will be the user-defined cycle length.

    For minimum temperature greater than 8℃, the crop cycle length is cross-checked with LGPt10. If LGPt10 is less the cycle length, LGPt10 will be selected as effective cycle length (CYCeff). If that is not the case, Otherwise, CYCeff will be the user-defined cycle length.
  - **Rainfed Condition**

    For rainfed perennial crops, the user-defined cycle length is compared with LGP of a particular location, if LGP is less than cycle length, the LGP will be used as effective cycle length (CYC_{eff}). If not, user-defined cycle length will be applied.

- **Leaf Area Index (LAI) and Harvest Index (HI) adjustment**

  The calculated effective cycle length will be used in perennial adjustment regarding LAI and HI, in addition to user-input adjustment parameterization. The pre-existing function of adjusted LAI and HI are calculated based Chapter 4 of the GAEZ v4 Model Documentation (Fischer et al., 2021).

To implement the mandatory feature of perennial adjustment, new function is introduced for users to initialize before the actual crop simulation.

```
# Set input layers required for minimum temperature requirement
aez.ImportLGPandLGPT(lgp, lgpt5, lgpt10)
```

Function Arguments	
lgp	A 2D NumPy array, corresponding to length of growing periods [days]
lgpt5	A 2D NumPy array, corresponding to temperature growing period at 5°C thresholds [days]
lgpt10	A 2D NumPy array, corresponding to temperature growing period at 10°C thresholds [days]

Function Returns
None

Commented [SWH15]: Removed, and added into new section. Perennial adjustment.

Calculations and outputs

Crop cycle simulation

After setting up all the related parameters, we can now run the crop cycle simulations/calculations by executing the function below. Note that depending on the dimension of your area of interest, the calculation duration will be long.

```
# Crop cycle simulation
aez.simulateCropCycle(start_doy=1, end_doy=365, step_doy=1, leap_year=False)
```

Function Arguments	
start_doy	A single value, corresponding to crop simulations starting Julian date. This is an optional argument. Default value is 1
end_doy	A single value, corresponding to crop simulations ending Julian date. This is an optional argument. Default value is 365

step_doy	A single value, corresponding to spacing (in days) between 2 adjacent crop simulations. This is an optional argument. Default value is 1
leap_year	• True or False, depending on whether the simulating year is a leap year or not. This allows handing leap and non-leap year differently. • This is only relevant for monthly climate data because this value will be used in interpolation processes. • In case of daily climate data inputs, length of daily climate data vector will be taken as number of days in a year. This is an optional argument, and the default value is False

Function Returns
None

Estimated Maximum Yield

These functions return the maximum attainable yield under the provided climate conditions in both rain-fed and irrigated conditions. The result's unit is in kilograms per hectare (kg/ha).

```
# Estimation of Maximum Yield for Rainfed scenario
yield_map_rain = aez.getEstimatedYieldRainfed()
# Estimation of Maximum Yield for Irrigated scenario
yield_map_irr = aez.getEstimatedYieldIrrigated()
```

Function Arguments
None

Function Returns	
yield_map_rain	2D NumPy array, the maximum attainable yield under the provided climate conditions, under rain-fed conditions [kg/ha]
yield_map_irr	2D NumPy array, the maximum attainable yield under the provided climate conditions, under irrigated conditions [kg/ha]

Optimum Crop Calendar

Commented [SWH16]: Update for new outputs of optimum starting dates.

```
# Optimum starting date for crop cycle
starting_date_rainfed = aez.getOptimumCycleStartDateRainfed()
starting_date_irrigated = aez.getOptimumCycleStartDateIrrigated()
```

Function Arguments	
None	

Function Returns	
starting_date	2D NumPy array for each function for rainfed or irrigated conditions. Each pixel value corresponds to the Julian day of the optimum crop cycle starting date within user-defined time length for simulation.

Thermal Reduction Factor (fc1)

Commented [SWH17]: New Output: Thermal Reduction Factor.

This new function provides the yield reduction factor due to thermal constraints derived from thermal screening procedures. Value ranges from 0 (most limiting) to 1 (very suitable).

```
# Obtaining fc1 factor map
factor= aez.getThermalReductionFactor()
```

Function Arguments	
None	

Function Returns	
fc1	A python list of two 2-D Numpy arrays, corresponding to fc1 maps such a manner as [fc1_rainfed, fc1_irrigated].

Moisture Reduction Factor (fc2)

Commented [SWH18]: New Output: Moisture Reduction Factor.

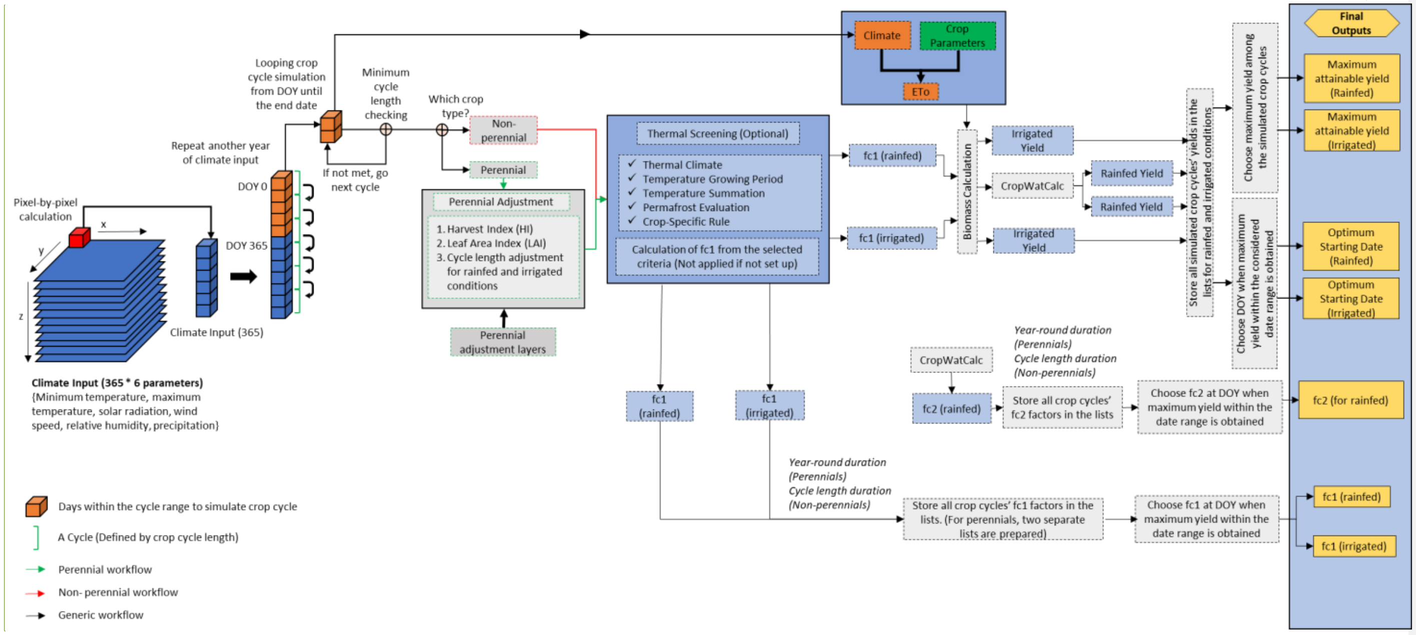
This new function provides the yield reduction factor due to moisture deficit derived from crop water requirement calculation. Value ranges from 0 (most limiting) to 1 (very suitable). Note that Fc2 map is now produced for annuals and perennials crops but is specific only for rainfed condition.

```
# Obtaining fc2 factor map
fc2_map = aez.getMoistureReductionFactor()
```

Function Arguments	
None	

Function Returns	
fc2	2-D Numpy array, corresponding to yield reduction factor due to moisture deficit.

Figure 8 Revised Procedure of Estimation of Maximum Attainable Yield (rain-fed and irrigated) and Optimum Starting Date (Module 2)



Commented [SWH19]: Updated routine of crop simulation.

MODULE 3: CLIMATE CONSTRAINTS

Introduction

In this module, various yield reduction factors will be applied to the maximum attainable yield estimated from Module 2 to consider the constraining effects which are difficult to simulate during the crop cycle simulation (Figure 9). For example, climatic effects can be pests, diseases, and poor workability due to excess soil moisture. These effects, in turn, depend on the different levels of inputs, LGP and LGP Equivalent.

All of the reduction factors used in Modules 3, 4, and 5 are located in 2 parameter files corresponding to irrigated and rain-fed conditions. These files MUST be edited with the reduction factors values corresponding to each crop and input level. Users are strongly encouraged to advise to use specific reduction factors based on national research for national-level analysis.

This module considers four types of agro-climatic constraints:

- Pests, diseases, and weeds damages on plant growth ('b' group).
- Pests, diseases, and weeds damages on produce's quality ('c' group).
- Climatic factors affecting the efficiency of farming operations ('d' group).
- Frost hazards ('e' group).

Commented [SWH20]: Content Update.

Note: The existing 'a' group corresponding with yield reduction due to rainfall variability (by GAEZ v4 documentation) is not considered in PyAEZ. See the *GAEZ v4 Model Documentation* for further details on the climate constraints (Fischer *et al.*, 2021).

In v2.2, Module 3 underwent calculation algorithm validation with respect to GAEZ outputs, and the user-intuitive way of providing agro-climatic constraints by integrating with excel sheets.

Similar to the previous modules, this module starts with importing and initiating the Class. In new version, you need additional information to be provided.

```
# Import and create a Class instgance
import ClimateConstraints
obj_constraints = ClimaticConstraints.ClimaticConstraints(lat_min, lat_max, elevation, mask, no_mask_value)
```

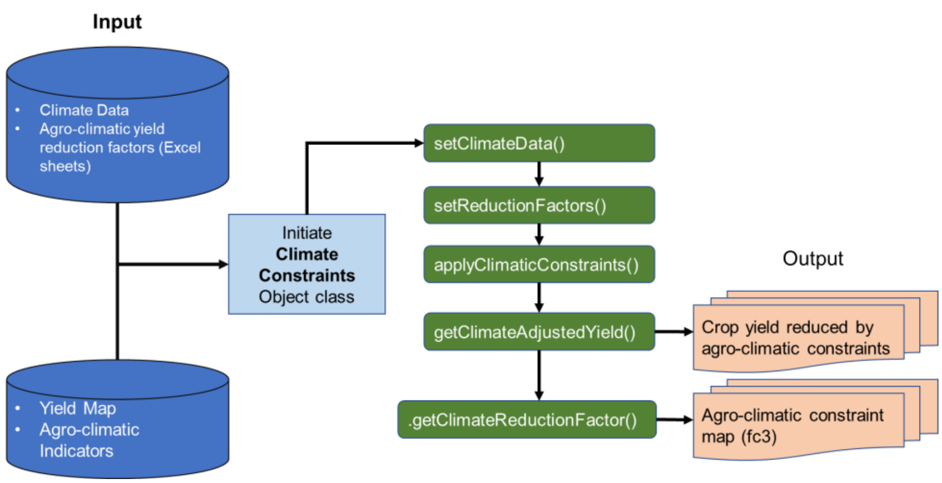
Function Arguments	
lat_min	Float, minimum latitude (Unit = decimal degrees)
lat_max	Float, Maximum latitude (Unit = decimal degrees)
elevation	2-D NumPy array, elevation (Unit = meters)
mask	2-D NumPy array, binary mask layer (0,1)

no_mask_value	Integer (1,0), specify which mask value is considered for calculation

Function Returns
None.

Figure 9 Overview of Module 3 (Climate Constraints) workflow

Commented [SWH21]: New flowchart for Module III.



Setting up parameter files

Starting from v2.2, agro-climatic constraints will be provided using the excel sheet. For Module III, users must prepare two excel sheets of agro-climatic reduction factors; a single excel file must contain three separate sheets for three look-up tables:

1. Mean>20 (for temperature greater than 20°C)
2. Mean<10 (for temperature less than 10°C
3. Lgpt10 (for frost-free periods)

The first and second look-up tables required 'b', 'c' and 'd' constraints factors (represented along row) for 14 wetness days (LGP_{agc}) class intervals. All agro-climatic reduction factors range from 0 (Most Suitable) to 100 (Not Suitable). An example excel sheet setting for mean>20 and mean<10 is provided below in Table 13 and Table 14. Notice that there are two categories for LGP_{agc} of 365 days, namely 365+ and 365-, depending on the number months where monthly precipitation is greater than monthly reference evapotranspiration (ETo). For locations with monthly precipitation greater than monthly ETo for all months, 365+ column will be used, otherwise, 365- column will be used.

For ‘e’ constraints factor settings, users will need to set up in **lgpt10** spreadsheet and set up reduction factors for 13 LGP_{t10} interval classes. An example excel sheet setting for **lgpt10** is provided in Table 15 . If the considered LUT/crops are not **frost-sensitive**, reduction factor for all day intervals should set up zero to cancel out ‘e’ constraint consideration.

During excel sheet preparation, users are advised to set up the column names and row names as strings in excel software before importing into the object class. Note also that the module calculates for a single water supply management. For instance, if users run Module III for rainfed condition, you will need to rerun with different yield and excel sheets for irrigated condition.

The calling of importing the excel sheet of agro-climatic constraints is as follows:

```
# importing the excel sheet of agro-climatic factors
obj_constraints.setReductionFactors(file_path)
```

Function Arguments	
file_path	String, the full file path where the excel sheet for agro-climatic constraints for rainfed/irrigated condition.
Function Returns	
None.	

Table 13 Agro-climatic Constraint Setting for Temperature Greater than 20°C (Maize Irrigated)

type	0,29	30,59	60,89	90,119	120,149	150,179	180,209	210,239	240,269	270,299	300,329	330,364	365-	365+
b	0	0	0	0	0	0	0	0	25	25	25	25	25	50
c	50	50	50	25	0	0	0	0	0	0	0	25	25	50
d	0	0	0	0	0	0	0	10	30	30	30	30	30	30

Table 14 Agro-climatic Constraint Setting for Temperature Less than 10°C (Maize Irrigated)

type	0,29	30,59	60,89	90,119	120,149	150,179	180,209	210,239	240,269	270,299	300,329	330,364	365-	365+
b	0	0	0	0	0	0	0	0	10	10	10	10	10	30
c	30	30	30	10	0	0	0	0	0	0	0	10	10	30
d	0	0	0	0	0	0	0	10	30	30	30	30	30	30

Table 15 Agro-climatic Constraint Setting for Frost-Free Periods (Maize Irrigated)

type	0,29	30,59	60,89	90,119	120,149	150,179	180,209	210,239	240,269	270,299	300,329	330,364	365
lgpt10	0	0	0	0	0	0	0	0	0	0	0	0	0

Importing climate data

We will upload the climate data required for the agro-climatic constraint assessment by using this function after calling the object class.

```
# importing the climate datasets
obj_constraints.setClimateData(min_temp, max_temp, wind_speed, short_rad,
rel_humidity, precipitation)
```

Function Arguments	
min_temp	3-D NumPy array (Daily/Monthly), corresponding to minimum temperature (°C)
max_temp	3-D NumPy array (Daily/Monthly), corresponding to maximum temperature (°C)
wind_speed	3-D NumPy array (Daily/Monthly), corresponding to the wind speed measured at 2 meters above (ms^{-1})
short_rad	3-D NumPy array (Daily/Monthly), corresponding to shortwave radiation (Wm^{-2})
rel_humidity	3-D NumPy array (Daily/Monthly), corresponding to relative humidity (decimal, 0-1)
precipitation	3-D NumPy array, corresponding to the precipitation ($mmday^{-1}$)
Function Returns	
None.	

Applying climate constraints

Commented [SWH22]: New documentation for inputs and outputs.

This function applies the climate-related yield reduction factors to produce the reduced yield:

```
# Apply agro-climatic constraints
yield_out = obj_constraints.applyClimaticConstraints(yield_input, lgp, lgp_equv,
lgpt10, omit_yld0)
```

Function Arguments	
yield_input	2-D NumPy array, corresponding to input yield (kg/ha)
lgp	2-D NumPy array, corresponding to length of growing period (days)

lgp_eq	2-D NumPy array, corresponding to LGP Equivalent (days)
lgpt10	2-D NumPy array, corresponding to frost-free periods (days)
omit_yld_0	Binary (1 = Any pixels with zero yield will not calculate fc3 0 = calculate fc3 regardless of input yield value)
Function Returns	
None.	

Climate Adjusted Yield

This function returns the yield map adjusted with agro-climatic constraints, based on the previous function setting.

```
# Getting climate adjusted yield
clim_adj_yield = obj_constraints.getClimateAdjustedYield()
```

Function Arguments	
None.	
Function Returns	
clim_adj_yield	A 2-D NumPy array, corresponding to the climate-adjusted yield (for rainfed/irrigated condition)

Agro-Climatic Reduction Factor (fc3)

This function returns yield reduction factor due to agro-climatic constraints for a particular management level users defined. The value ranges from 0 to 1 (from the most limiting condition to the most suitable condition).

```
# Getting climate adjusted yield
fc3_map = obj_constraints.getClimateReductionFactor()
```

Function Arguments	
None.	
Function Returns	
fc3	A 2-D NumPy array, corresponding to the yield reduction factor due to agro-climatic constraints.

MODULE 4: SOIL CONSTRAINTS

Introduction

After applying the agro-climatic constraints onto the maximum attainable yield, we will now apply the soil constraints (Figure 10).

The combination of 7 soil qualities (SQ), which are based on the soil characteristics of each soil unit, and the input level gives us a single yield reduction factor – *Soil Rating*, which will be applied to the remaining yield. For more details on this calculation, please refer to the *GAEZ v4 Model Documentation* (Fischer *et al*., 2021).

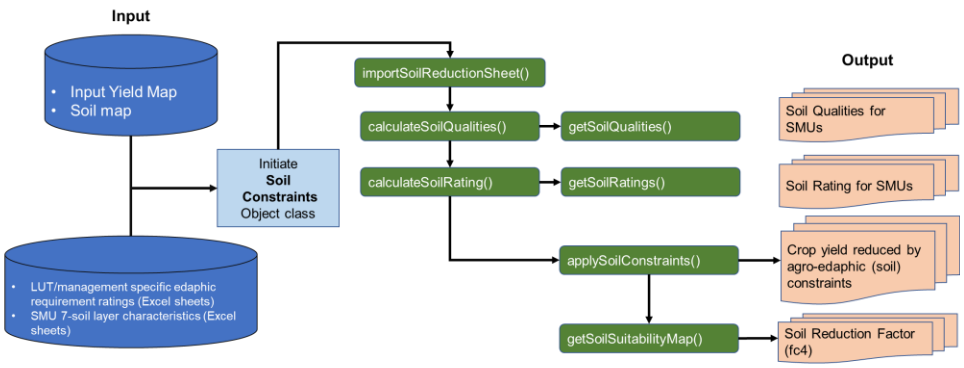
- SQ1: Nutrient availability
- SQ2: Nutrient retention capacity
- SQ3: Rooting conditions
- SQ4: Oxygen availability to roots
- SQ5: Excess salts
- SQ6: Toxicity
- SQ7: Workability (constraining field management)

In v2.2, all soil reduction parameters and SMU's soil characteristics are required to prepare and import them as excel sheets instead of setting the parameters from python sheets. Seven-soil layers evaluation (Nachtergaele et al., 2023) scheme is implemented. The detailed settings of these excel sheets are explained in the next section.

First, we must import Module 4 Class:

```
from pyaez import SoilConstraints
soil_constraints = SoilConstraints.SoilConstraints()
```

Figure 10 Overview of Module 4 (Soil Constraints) workflow



Setting up parameter excel files (Mandatory)

There are two aspects when it comes to soil constraints to prepare from users:

- soil's physical and chemical characteristics of each SMU in an area for top-soil and subsoil conditions,
- input/LUT-specific edaphic requirement and ratings

For soil characteristics preparation for a single water supply condition (rainfed/irrigated), an excel sheet must provide listed soil parameters from Table 3, for seven sheets; each represents a soil depth layer class defined below:

Table 16 Soil Depth Class Definition (Source: HWSD v2.0)

Soil Depth Class	Depth of top layer (cm)	Depth of bottom layer (cm)
D1	0	20
D2	20	40
D3	40	60
D4	60	80
D5	80	100
D6	100	150
D7	150	200

For edaphic requirement excel sheet setting, these settings are LUT/management specific. Thus, users are required to consider whether it is rainfed/irrigated, low/intermediate/high management etc., and specify reduction factors for each soil quality. Notice that in a single soil quality, there are more than two soil properties utilized to calculate soil quality. Each soil property contains the actual value and its corresponding suitability factors which both work together as one. In numerical parameter settings, linear interpolation is applied to SMU-specific soil suitability factors. For categorical, the suitability factors are directly used to determine soil quality.

In Module IV, the first column is regarded as the main headers of the soil parameters in each soil quality. Thus, they must be defined as text format in excel program. When preparing the actual values and their respective soil suitability factors, regardless of how many users provide along row direction, both actual values and their suitability factors must have the same number of elements. As python is case-sensitive during categorical settings, please make sure from users to define string data types of uppercase and lowercase carefully set up for all soil quality sheets.

The example excel sheet for soil characteristics and LUT/management specific edaphic requirement are provided in.

Table 17 Example Excel Sheet of Edaphic Requirement of (Maize, high input)

For Soil Quality 1 (Nutrient Availability)							
TXT_val	Fine	Medium	Coarse				
TXT_fct	90	70	30				
OC_val	0	0.8	1.5	2			
OC_fct	50	70	90	100			
pH_val	3.6	4.1	4.5	5	5.5	6	
pH_fct	10	30	50	70	90	100	
TEB_val	0	1.6	2.8	4	6.5		
TEB_fct	30	50	70	90	100		
For Soil Quality 2 (Nutrient Retention Capacity)							
TXT_val	Fine	Medium	Coarse				
TXT_fct	90	70	30				
BS_val	0	35	50	80			
BS_fct	50	70	90	100			
CECsoil_val	0	2	4	8	10		
CECsoil_fct	30	50	70	90	100		
CECclay_val	0	16	24				
CECclay_fct	70	90	100				

pH_val	3.6	4.1	4.5	5	5.5	6	
pH_fct	10	30	50	70	90	100	
For Soil Quality 3 (Rooting Conditions)							
RSD_val	35	70	85				
RSD_fct	50	90	100				
SPH_val	Lithic	skeletic	hyperskeletic				
SPH_fct	100	50	30				
OSD_val	0						
OSD_fct	100						
SPR_val	0	1					
SPR_fct	100	90					
For Soil Quality 4 (Oxygen Availability)							
DRG_val	VP	P	I	MW	W	SE	E
DRG_fct	50	90	100	100	100	100	100
SPH_val	Lithic	skeletic	hyperskeletic				
SPH_fct	100	50	30				
For Soil Quality 5 (Presence of Salinity and Sodicity)							
ESP_val	10	20	30	40	100		
ESP_fct	100	90	70	50	10		
EC_val	1	2	4	6	12	100	
EC_fct	100	90	70	50	30	10	
SPH_val	Lithic	skeletic	hyperskeletic				
SPH_fct	100	50	30				
For Soil Quality 6 (Presence of Lime and Gypsum)							
CCB_val	3	6	15	25	100		
CCB_fct	100	90	70	50	10		
GYP_val	1	3	10	15	100		
GYP_fct	100	90	70	50	10		
SPH_val	Lithic	skeletic	hyperskeletic				
SPH_fct	100	50	30				

For Soil Quality 7 (Workability)							
RSD_val	35	70	85				
RSD_fct	50	90	100				
GRC_val	10	30	90				
GRC_fct	100	35	10				
SPH_val	Lithic	skeletic	hyperskeletic				
SPH_fct	100	50	30				
TXT_val	Fine	Medium	Coarse				
TXT_fct	90	70	30				
VSP_val	0	1					
VSP_fct	100	90					

Note: Users must set up edaphic suitability factors for each soil characteristic for all soil qualities. All settings must reflect on LUT type, input management level, and water supply system.

Table 18 Example Excel Sheet Setting of SMU-specific Soil Characteristics of a Particular Soil Depth Class from Soil Map and Database (Source: HWSD v2.0)

CODE	TXT	OC	pH	TEB	BS	CEC_soil	CEC_clay	RSD	SPR	SPH	OSD	DRG	ESP	EC	CCB	GYP	GRC	VSP
4260	Sandy clay loam	1.204	5.1	3	43	7	17	100	0	Lithic	0	MW	2	0	0	0	11	0
4261	Sandy clay loam	1.399	5.3	4	47	8	16	100	0	Lithic	0	MW	2	0	0	0	21	0
4264	Sandy clay loam	1.204	5.1	3	43	7	17	100	0	Lithic	0	MW	2	0	0	0	11	0
4265	Sandy clay loam	1.399	5.3	4	47	8	16	100	0	Lithic	0	MW	2	0	0	0	21	0
4267	Sandy clay loam	1.204	5.1	3	43	7	17	100	0	Lithic	0	MW	2	0	0	0	11	0
4284	Sandy clay loam	1.204	5.1	3	43	7	17	50	0	Lithic	0	MW	2	0	0	0	11	0
4325	Clay loam	1.316	6.1	17	82	19	53	100	0	Lithic	0	I	2	2	0	0.6	19	0
4383	Clay (light)	1.372	7.6	52	99	39	76	100	0	Lithic	0	MW	3	1	6	2.1	11	0
4408	Loam	1.263	4.8	2	32	8	16	100	0	Lithic	0	P	2	0	0	0	4	0
4452	Sandy clay loam	1.204	5.1	3	43	7	17	100	0	Lithic	0	MW	2	0	0	0	11	0
4499	Clay loam	1.316	6.1	17	82	19	53	100	0	Lithic	0	I	2	2	0	0.6	19	0
4544	Clay loam	2.196	5	8	38	19	40	100	1	Lithic	0	MW	1	1	0	1.6	14	1
4587	Clay (light)	1.896	4.9	3	25	9	9	100	0	Lithic	0	MW	1	0	0	0	4	0
6651	Loam	3.163	6.7	10	64	16	35	10	0	Lithic	0	I	3	1	0	0.3	36	0
7001	Sandy clay loam	2	6	6.5	80	10	24	100	0	Lithic	0	MW	10	1	3	1	10	0
11772	Clay (light)	1.945	4.9	5	28	10	10	100	0	Skeletic	0	MW	1	0	0	0	4	0
11788	Sandy clay loam	1.209	5.2	2	43	7	12	100	0	Lithic	0	MW	1	0	0	0	3	0

Note: Users must provide such formatted excel sheet for top-soil and sub-soil conditions, each condition must have seven sheets, each representing a soil depth class.

Once all the soil parameter settings are finalized, once the object class is declared, these prepared excel sheets must be imported using the following function.

```
# Importing the excel sheet of soil
soil_constraints.importSoilReductionSheets(rain_sheet_path, irr_sheet_path)
```

Function Arguments	
rain_sheet_path	String, corresponding to the LUT/input management-specific edaphic requirement for rainfed condition
irr_sheet_path	String, corresponding to the LUT/input management-specific edaphic requirement for irrigated condition
Function Returns	
None	

Calculate soil qualities (Mandatory)

This function calculates 7 soil qualities for each SMU based on the input soil characteristics.

```
# Soil Qualities
soil_constraints.calculateSoilQualities(irr_or_rain, topsoil_path, subsoil_path)
```

Function Arguments	
irr_or_rain	String, indicating calculations are considered under either rain-fed condition or irrigated condition. 'R' is for rain-fed condition, and 'I' is for irrigated condition
topsoil_path	String, corresponding to the file path of excel sheet of listed SMU’s top-soil characteristics for all soil depth layers
subsoil_path	String, corresponding to the file path of excel sheet of listed SMU’s sub-soil characteristics for all soil depth layers
Function Returns	
None	

Calculate soil ratings (Mandatory)

This function calculates soil ratings for each soil unit, combining 7 soil qualities based on input level. This must be run after calling soil quality calculation.

```
# Soil rating
soil_constraints.calculateSoilRating(input_level)
```

Function Arguments	
input_level	Single character String, corresponding to input level. 'L' is for Low input level, 'I' is for Intermediate input level, and 'H' is for Hight input level.

Function Returns
None

Extracting soil qualities

This function returns 7 soil qualities calculated for each SMU based on the input soil characteristics.

```
# Extracting soil qualities
soil_qualities = soil_constraints.getSoilQualities()
```

Function Arguments
None

Function Returns	
soil_qualities	A pandas dataframe. Columns are soil qualities from SQ1 to SQ7 provided for each SMU.

Extracting soil ratings

This function returns 7 soil qualities calculated for each SMU based on the input soil characteristics.

```
# Extracting soil qualities
soil_ratings = soil_constraints.getSoilRatings()
```

Function Arguments
None

Function Returns

soil_ratings	A pandas dataframe. Each SMU corresponds to a final soil rating.

Applying soil constraints (Mandatory)

This function applies all soil-related yield reduction factors. The returned yield map is now soil-adjusted yield map.

```
# Soil Constraints
yield_out = soil_constraints.applySoilConstraints(soil_map, yield_in)
```

Function Arguments	
soil_map	2D NumPy array, corresponding to soil unit. Each pixel value must be SMU. This code is used to match the soil rating with the input yield.
yield_in	2D NumPy array, corresponding to the yield before applying the soil reduction factors (either rainfed or irrigated conditions)
Function Returns	
yield_out	2D NumPy array. The yield reduced by soil-related factors [same unit as yield_in]

Getting soil suitability map

This function returns soil suitability map after applying the soil constraints. Soil suitability factor (fc4) ranges from 0 (Not Suitable) to 1 (Very Suitable). Note that this soil suitability reflects the **input/management level** users specified earlier in soil rating calculation.

```
# getting the soil suitability
Fc4_map = soil_constraints.getSoilSuitabilityMap()
```

Function Arguments	
None.	
Function Returns	
Fc4	2D NumPy array. Corresponding to soil constraint factor.

MODULE 5: TERRAIN CONSTRAINTS

Introduction

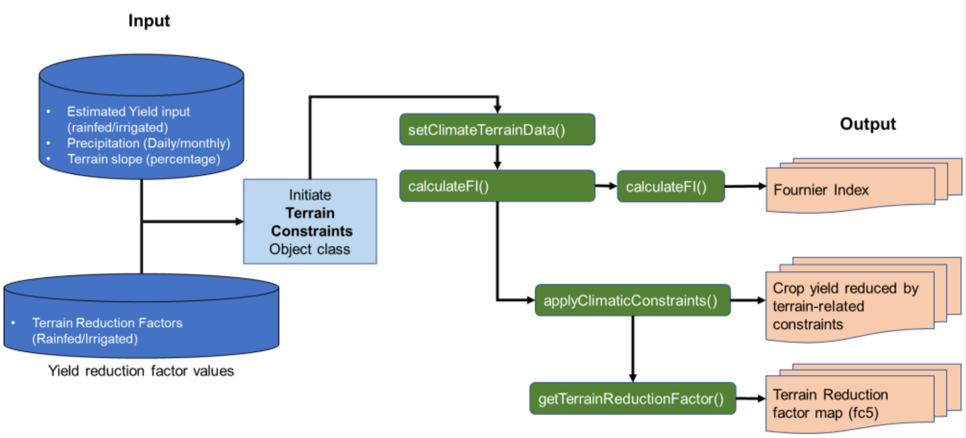
This section introduces the yield reduction due to terrain slope, soil erosion, and Fournier Index (FI) (Figure 11). The FI is based on the monthly precipitation (climate-related). These yield reduction factors will be applied to the maximum attainable yield. For detailed calculations for this section, please refer to the *GAEZ v4 Model Documentation* (Fischer *et al*., 2021).

In v2.2, users require the terrain constraint factors to be provided as excel sheet and python script mode is now cancelled.

First, we must import Module 5 Class:

```
from pyaez import TerrainConstraints
terrain_constraints = TerrainConstraints.TerrainConstraints()
```

Figure 11 Overview of Module 5 (Terrain Constraints) workflow



Setting up parameter excel files

Two excel sheets should be prepared for rainfed and irrigated conditions before importing into the object class. Two class types: eight slope class intervals and 6 Fournier Index (FI classes) must be set up in an excel sheet and their corresponding terrain reduction factors must be set up as shown in Table 19.

Table 19 Example Excel Sheet Preparation of Terrain Reduction Factors (Maize, High Input)

	Slope Classes								
	Classes	**0,0.5**	**0.5,2**	**2,5**	**5,8**	**8,16**	**16,30**	**30,45**	**45,100**
FI Classes	**0,1300**	100	100	75	50	25	0	0	0
	1300,1800	100	100	100	100	100	75	0	0
	1800,2200	100	100	100	100	75	25	0	0
	2200,2500	100	100	100	100	50	0	0	0
	2500,2700	100	100	100	100	25	0	0	0
	2700,10000	100	100	100	100	25	0	0	0

Once setup is completed, we will use the following function to import into the object class.

```
# Set up terrain reduction factor excel sheets
terrain_constraints.importTerrainReductionSheet(irr_file_path, rain_file_path)
```

Function Arguments	
irr_file_path	String, corresponding to the full file path of terrain reduction factor excel sheet for irrigated condition.
rain_file_path	String, corresponding to the full file path of terrain reduction factor excel sheet for rainfed condition.
Function Returns	
None	

Setting up inputs

Climate and terrain inputs

This function allows users to set up the monthly precipitation and terrain slope data. This is a mandatory step before executing further calculations.

```
# Set up climate and slope data
terrain_constraints.setClimateTerrainData(precipitation, slope)
```

Function Arguments	
precipitation	3D NumPy array corresponding to monthly or daily precipitation. Since Fournier Index (FI) is a ratio, unit conversion factors will be cancelled out
slope	2D NumPy array, corresponding to terrain slope. [Percentage Slope]

Function Returns
None

Calculate Fournier Index

This function calculates Fournier Index (FI) based on the input monthly precipitation. FI is a simple index that indicates the potential of soil erosion based on monthly precipitation.

```
# Calculate Fournier Index
terrain_constraints.calculateFI()
```

Function Arguments
None

Function Returns
None

Extract Fournier Index

This function returns Fournier Index (FI), which is based on the input monthly precipitation. This is an optional function. FI can be extracted with this function if required.

```
# Extract Fournier Index
fi = terrain_constraints.getFI()
```

Function Arguments
None

Function Returns	
fi	2D NumPy array, corresponding to Fournier Index (FI) based on the input monthly precipitation

Applying terrain constraints

This function applies the terrain-related yield reduction factors.

```
# Apply Terrain Constraints
yield_out = terrain_constraints.applyTerrainConstraints(yield_in, irr_or_rain)
```

Function Arguments	
yield_in	2D NumPy array, corresponding to the yield before applying the terrain-related reduction factor. This can be the yield under either irrigated or rain-fed conditions from Module 4
irr_or_rain	Single character String, indicating yield in is in either rain-fed or irrigated condition. 'R' is for rain-fed condition, and 'I' is for irrigated condition
Function Returns	
yield_out	2D NumPy array. The yield reduced by soil-related factors [same unit as yield_in]

Getting terrain suitability map

This function returns the terrain suitability map which can be obtained after applying the terrain constraints for a single water supply condition (rainfed/irrigated). The terrain reduction factor ranges from 0 (Not Suitable) to 1 (Very Suitable).

```
# getting the terrain suitability map
fc5 = terrain_constraints.getTerrainReductionFactor()
```

Function Arguments	
None.	
Function Returns	
fc5	2D NumPy array. Corresponding to terrain reduction factor (0-1)

MODULE 6: ECONOMIC SUITABILITY ANALYSIS

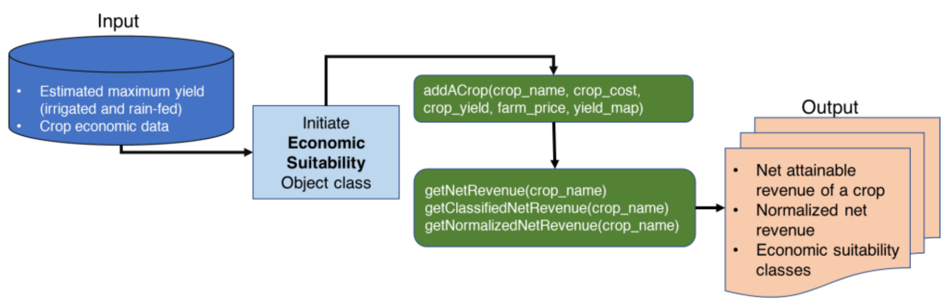
Introduction

Economical Suitability Analysis Module is the most recent addition to AEZ framework (Figure 12). This module converts AEZ's final crop suitability (a result of the previous 5 modules) into an economic suitability. Additionally, all crops of interest are compared to the umbrella crop (crop with the highest economical potential) in order to indicate and map out its comparative advantage in terms of an attainable net revenue relative to the best available option. For more detailed calculations, refer to Module 6 chapter in National Agro-Economic Zoning for Major Crops in Thailand (NAEZ) report (FAO, 2017a).

First, we have to import the Module 6 Class and create an instance of that Class as below.

```
import EconomicSuitability
econ_su = EconomicSuitability.EconomicSuitability()
```

Figure 12 Overview of Module 6 (Economic Suitability) workflow



Crop parameters inputs

This function allows users to set up the crop parameters for an economic analysis. The key inputs for Module 6 are the crop yield information generated from the previous 5 modules in PyAEZ, market prices, and the costs of the crop.

This function will be called multiple times as it performs a comparative economic analysis. This is a mandatory function to run before any further calculations.

```
# Crop parameter input
econ_su.addACrop(crop_name, crop_cost, crop_yield, farm_price, yield_map)
```

Function Arguments	
crop_name	A single string value, corresponding to the crop name that you are adding. This name will be used later to extract output for each crop
crop_cost	• 1D NumPy array, corresponding to the cost of production for each yield values in crop yield variable • Values of crop_cost and crop_yield must be corresponding to each other, and they must be in ascending order • Units of this variable must be in cost per hectare • All the costs and prices in this module must be in same currency
crop_yield	• 1D NumPy array, corresponding to the yield values • Values of crop_cost and crop_yield must be corresponding to each other, and they must be in ascending order • Units of this variable must be in tonnes per hectare
farm_price	• 1D NumPy array, corresponding to the historical crop price that farmers sell • The price array is used to calculate distribution (mean) of prices • Unit: price (same currency throughout unit per tonne)
yield_map	2D NumPy array, corresponding to yield map of the crop. Unit: tonnes per hectare
Function Returns	
None	

Net Revenue

This function returns net revenue from the crop identified with 'crop_name'.

```
# Get the Net Revenue
crop_rev = econ_su.getNetRevenue(crop_name)
```

Function Arguments	
crop_name	A single string value, corresponding to the crop name
Function Returns	
crop_rev	2D NumPy array, net revenue of the input crop_name. Unit: revenue per hectare

Classified Net Revenue

This function returns classified net revenue for the crop ‘crop_name’. The classification scheme for crop net revenue is outlined in Table 20.

```
# Net revenue classification
crop_rev_class = econ_su.getClassifiedNetRevenue(crop_name)
```

Function Arguments	
crop_name	A single string value, corresponding to the crop name

Function Returns	
crop_rev_class	2D NumPy array, classified net revenue of the input crop_name

Table 20 Net Revenue Classification

Pixel value	Net Revenue Class	Description
0	Not suitable	Net revenue less than 0%
1	Very marginal	Net revenue between 0% and 10%
2	Marginal	Net revenue between 10% and 20%
3	Moderate	Net revenue between 20% and 30%
4	Medium	Net revenue between 40% and 50%
5	Good	Net revenue between 50% and 63%
6	High	Net revenue between 63% and 75%
7	Very high	Net revenue is equivalent to 75% or more than the overall maximum

Normalized Net Revenue

This function returns the normalized net revenue for the crop ‘crop_name’. The normalization is done, firstly, by assigning the highest possible net revenue, among crops passed through the module, to 1 (i.e. an umbrella crop). Secondly, the net revenue values of other crops are normalized as a portion of the umbrella crop (0–1 scale). This normalization process is performed separately for each pixel.

```
# Normalized net revenue
crop_rev_norm = econ_su.getNormalizedNetRevenue(crop_name)
```

Function Arguments	
crop_name	A single string value, corresponding to the crop name
Function Returns	
crop_rev_nor m	2D NumPy array, normalized net revenue of the input crop_name. Output values between 0 and 1

UTILITY CALCULATIONS

Introduction

This section will outline the additional calculation routines used throughout the PyAEZ's 6 main modules. These functions are contained within a Class called 'UtilitiesCalc'.

The functions are as follows:

Functions in UtilitiesCalc	Description
interpMonthlyToDaily	Perform monthly-to-daily interpolation for climate data
averageDailyToMonthly	Aggregate daily climate data into monthly data
generateLatitudeMap	Generate latitude map as 2D NumPy array, by linearly interpolating the bottom and top latitudes of the study area
classifyFinalYield	Classify yield estimation and produce suitability map according to AEZ's classification scheme
saveRaster	Saving 2D NumPy arrays as GeoTIFF raster files
averageRasters	Averaging a list of rasters in the time-dimension
windSpeedAt2m	Convert windspeed from a particular altitude to 2m above the surface

To use this UtilitiesCalc Class, we first must import and create a Class instance:

```
import UtilitiesCalc
obj_utilities = UtilitiesCalc.UtilitiesCalc()
```

Monthly-to-daily interpolation

This function performs interpolation of monthly climate data into daily climate data with quadratic spline interpolation as recommended in AEZ framework. The interpolation is performed between `cycle_begin` and `cycle_end` Julian dates.

```
# Monthly-to-daily interpolation
daily_vector = obj_utilities.interpMonthlyToDaily(monthly_vector, cycle_begin,
cycle_end, no_minus_values=False)
```

Function Arguments

monthly_vector	1D NumPy array with 12 elements corresponding to the monthly climate data
cycle_begin	A single value corresponding to the beginning Julian date of the crop cycle
cycle_end	A single value corresponding to the ending Julian date of the crop cycle
no_minus_values	True or False. If this argument is True, negative values will be forced to be zero. This helps getting rid of any unrealistic negative interpolated values in the climate parameters such as precipitation data. If this argument is False, then negative values are allowed. By default, this argument is set as False and it's not a mandatory argument to pass
Function Returns	
daily_vector	1D NumPy array, corresponding to the output daily climate data between cycle_begin and cycle_end Julian dates

Daily-to-monthly aggregation

This function aggregates daily climate data into monthly climate data. The aggregation is done by averaging the data in each month.

```
# Daily-to-monthly aggregation
monthly_vector = obj_utilities.averageDailyToMonthly(daily_vector)
```

Function Arguments	
daily_vector	1D NumPy array with 365 elements corresponding to the daily climate data
Function Returns	
monthly_vector	1D NumPy array with 12 elements corresponding to the aggregated monthly climate data

Create latitude map

The latitude map is created by linearly interpolating the bottom and the top latitude values of the study area, as defined by the user's input.

```
# Generate latitude map
lat_map = obj_utilities.generateLatitudeMap(lat_min, lat_max, im_height, im_width)
```

Function Arguments	
lat_min	A single value corresponding to the minimum latitude as decimal degree

lat_max	A single value corresponding to the maximum latitude as decimal degree
im_height	A single value corresponding to height of resulting latitude map as number of pixels
im_width	A single value corresponding to width of resulting latitude map as number of pixels
Function Returns	
lat_map	2D NumPy array, corresponding to latitude map. The resulting dimension of the latitude map will be im_height and im_width respectively

Classify the final crop yield

This function classifies yield estimations and produces suitability maps according to classification scheme defined in AEZ framework. The classification scheme consists of 5 classes (very suitable, suitable, moderately suitable, marginally suitable, and not suitable) (Table 21).

```
# Classification of yield estimation
est_yield_class = obj_utilities.classifyFinalYield(est_yield)
```

Function Arguments	
est_yield	2D NumPy array corresponding to the estimated yield
Function Returns	
est_yield_class	2D NumPy array, corresponding to the suitability map after yield classification

Table 21 Yield suitability classification

Pixel value	Suitability Class	Description
1	Not suitable	Yields between 0% and 20% of the overall maximum yield
2	Marginally suitable	Yields between 20% and 40% of the overall maximum yield
3	Moderately suitable	Yields between 40% and 60% of the overall maximum yield
4	Suitable	Yields between 60% and 80% of the overall maximum yield
5	Very suitable	Yields are equivalent to 80% or more of the overall maximum yield

Saving GeoTIFF rasters

This function allows saving 2D numpy array as GeoTIFF raster file. This function can be used to save any output of this PyAEZ package as a GeoTIFF raster file.

```
# Save 2D NumPy to GeoTIFF
obj_utilities.saveRaster(ref_raster_path, out_path, numpy_raster)
```

Function Arguments	
ref_raster_path	String, locating reference raster. This must be GeoTIFF raster file. Projection information is copied from this raster to final raster. Any input GeoTIFF raster to PyAEZ package with Projection information can be passed for this argument.
out_path	String, the desired location to save the output GeoTIFF file (with .tif extension).
numpy_raster	2D NumPy array, corresponding to the raster that user wants to save. Please make sure that the array dimensions for the reference GeoTIFF file and 2-D numpy array to insert Projection information the same to migitate error.

Commented [SWH23]: More description.

Function Returns	
None	

Averaging raster files

This function averages list of raster files in time dimension. Some calculations in the AEZ framework are recommended to perform with averaged climate data for 30 years. This function can be used for such calculations.

```
# Averaging raster files
avg_raster = obj_utilities.averageRaster(raster_3d)
```

Function Arguments	
raster_3d	3D NumPy array, corresponding to any climate data. The averaging will be done by the time dimension (across the years)

Function Returns	
avg_raster	2D NumPy array, the averaged climate data – into 'one year' worth of data

Calculate wind speed at 2m altitude

This function converts wind speed from a particular altitude to wind speed at 2m altitude. All of the wind speed values used in PyAEZ calculations are at 2m altitude, however, it is common for climate data services to offer the wind speed at 10m altitude, hence this conversion need to be applied in cause wind speed at 2 meters is not available.

```
# Converting to wind speed at 2m altitude
wind_speed_2m = obj_utilities.windSpeedAt2m(wind_speed, altitude)
```

Function Arguments	
wind_speed	A NumPy array (can be 1D, 2D or 3D), corresponding to wind speed
altitude	A single value corresponding to the altitude (above ground) [m]
Function Returns	
wind_speed_2m	Converted wind speed at 2m altitude as a NumPy array. Units will be same as unit of wind_speed

REFERENCES

Allen, R. G., Pereira, L. S., Raes, D., & Smith, M. (1998). Crop Evapotranspiration (guidelines for computing crop water requirements). *FAO Irrigation and Drainage Paper, 56.* Rome, Italy, Food and Agriculture Organization of the United Nations.

de Wit, C. T. (1965). Photosynthesis of leaf canopies. *Agricultural Research Reports*, *663*.

Doorenbos, J. & Pruitt, W. O. 1977. *Guidelines for predicting crop water requirements, FAO Irrigation and Drainage Paper N 24*. FAO Irriga edition. Rome, Italy, Food and Agriculture Organization of the United Nations. 154 pp.

FAO. (2017a). *National Agro-Economic Zoning for Major Crops in Thailand (NAEZ) (Project TCP/THA/3403): NAEZ model implementation and results : final report*

FAO. (2017b). The future of food and agriculture: trends and challenges. In D. Godoy, J. Dewbre, C. J. Amegnaglo, Y. Y. Soglo, A. F. Akpa, M. Bickel, S. Sanyang, S. Ly, J. Kuiseu, S. Ama, B. P. Gautier, R. Eberlin, E. Oduro-ofori, P. Aboagye Anokye, N. E. A. Acquaye, V. M. Dandelar, & J. Mineo (Eds.), *The future of food and agriculture: trends and challenges* (Vol. 4, Issue 4). Food and Agricultural Organisation of the United Nations.

Fischer, G., Nachtergaele, F., Velthuizen, H. van, Chiozza, F., Franceschini, G., Henry, M., Muchoney, D., & Tramberend, S. (2021). *Global Agro-Ecological Zones v4 – Model Documentation*. Food and Agricultural Organisation of the United Nations. https://doi.org/10.4060/cb4744en

Kim, C. (2010). The Impact of Climate Change on the Agricultural Sector : Implications of the Agro - Industry for Low Carbon , Green Growth Strategy and Roadmap for the East Asian Region Table of Contents. *Low Carbon Green Growth Roadmap for Asia and the Pacific*, 1–51.

Monteith, J. L. 1965. Evapotranspiration and the environment. *The State and Movement of Water in Living Organisms. XIXth Symposium*. Society for Xp. Biology, Swansea. pp. 205–234. Paper presented at, 1965, Cambridge, UK.

Monteith, J. L. 1981. Evapotranspiration and surface temperature. *Quarterly Journal Royal Meteorological Society*, 107: 1–27.

Nachtergaele, F., Velthuizen, H. Van, & Verelst, L. (2023). Harmonized World Soil Database version 2.0. In *Food and Agriculture …*. FAO; International Institute for Applied Systems Analysis (IIASA); https://doi.org/10.4060/cc3823en

Nelson, G. C., Rosegrant, M. W., Koo, J., Robertson, R. D., Sulser, T., Zhu, T., Ringler, C., Msangi, S., Palazzo, A., Batka, M., Magalhaes, M., Valmonte-Santos, R., Ewing, M., & Lee, D. R. (2009). *Climate change: Impact on agriculture and costs of adaptation*. https://doi.org/10.2499/0896295354

Ray, D. K., Mueller, N. D., West, P. C., & Foley, J. A. (2013). Yield Trends Are Insufficient to Double Global Crop Production by 2050. *PLOS ONE*, *8*(6), e66428.

UNDESA. (2017). *World population projected to reach 9.8 billion in 2050, and 11.2 billion in 2100.* United Nations Department of Economic and Social Affairs.