LÆND

Documentation v.0.3.1

Dorothee Birnkammer, Heidi Hottenroth

May 13, 2022

The development of this version was carried out within the research project InPEQt - Integrated cost- and life-cycle-based planning of decentralized energy systems for energy and resource efficient neighborhood development. The project was funded by the German Federal Environmental Foundation.

Contents

1. About LAEND 3
2. Installing LAEND v0.3.1 4
3. File Structure of LAEND 4
4. Running LAEND 5
5. Configuring the Energy System Model 6
   - 5.1. Scenario.xlsx 6
     - Buses 6
     - Demand 7
     - Commodity_Sources 7
     - Renewables 7
     - Storages 8
     - Transformers_in 9
     - Transformers_out 10
     - Timeseries 11
   - 5.2. Configurations in config.py 11
     - Temporal settings 12
     - Set Optimization Objectives 12
     - Set multiprocessing 12
     - Typical meteorological year (TMY) 12
     - Location settings 13
     - Updating fixed demand load curves 13
     - Input parameters for bdew heat demand 14
     - Updating Fixed Renewable Energy Load Curves 14
     - LCA Update 15
     - Normalization & Weighting 16
     - Emission Constraints 16
     - Defining Climate Neutrality 16
     - Financial Settings 17
     - Technical Settings 17
   - 5.3. Logging 17
6. Results 18

1. About LAEND

LAEND stands for "**L**ife Cycle **A**ssessment based **EN**ergy **D**ecision", which is based on the open source toolbox oemof and the life cycle assessment software openLCA and enables a coupled energy system analysis with environmentally oriented sustainability assessment and optimization. The tool is developed in Python and uses different Python libraries besides the mentioned tools. LAEND is based on oemof 0.4.1. Parts of this version of LAEND were created by Dorothee Birnkammer during her Master Thesis.

LAEND is primarily an extension of oemof that incorporates environmental impacts in the optimization. For the multi-objective optimization, the model minimizes total impacts, which are the sum of weighted and normalized monetary and environmental impacts. Alternatively, LAEND optimizes for a single objective by setting irrelevant values to zero. Environmental impact data in LAEND is imported through a link to openLCA, a program for modeling life cycle systems. Data supplied stems mainly from ecoinvent 3.7. The environmental footprint (EF) method 2.0 is applied as an impact method. This method provides data for 19 indicators and corresponding normalization and weighting factors. LAEND launches the phase of the myopic optimization. The program creates an objective-specific energy system model and then optimizes to minimize impacts. This myopic optimization includes three steps: Create energy system, Optimize for one year and Pass results to next year. Thus the iterative process involves a single-year calculation that serves as the basis for the optimization in the subsequent year. Within an optimization for a single year, perfect foresight is assumed. These steps are repeated until the entire modeling period has been optimized. For the myopic optimization of Year 1, the graph structure of the energy system is created from a set of investment options. All components include information about investment impacts (invcost, invenv) and flows are parameterized with varcost and varenv, summarized as impact data. Next, the energy system model is optimized with perfect foresight for the first year of the modeling period. The solution is a set of selected technologies with corresponding installed capacities. This set is saved along with hourly dispatch results. The set of established installations is passed to the following year in the optimization, indicated by the green arrows. For the second optimization year, the program creates a graph based on previous investments, provided that the technical lifetime of the installation has not been exceeded. The resulting components contain no information about investment impacts but detail the existing capacity capmax. Flow parameters include varcost and varenv. Additionally, the graph is supplemented with new investment options. These are added with complete impact data. In the second year, the optimization problem can decide to employ existing installations or invest in new installations, if required. Again, the results present as a set of technologies with associated capacities. The iterative process stops after the last year in the modeling period has been solved for and results have been saved. If representative years are enabled, the calculation only optimizes the representative years and assumes that the previous year is equal to the preceding representative year.

Optimization outcomes are multiplied with all impact factors to compile results. The final result for one objective encompasses investment and use phase information. Investment data includes the implemented installation, its capacity, and the impact incurred over the modeling period. Use phase data contains the sum of energy required or supplied in each year and the corresponding variable impact. If representative years are used, each value is multiplied by the number of years in the period that one year represents. In sum, this provides the total impacts incurred by optimizing myopically for one objective over the entire modeling period.

As an option political emission reduction targets are handled with emission constraints, an adjustable decarbonization foresight horizon, and model-specific climate neutrality.

Helpful resources

- Book *Optimization of Energy Supply Systems* by Janet Nagel https://doi.org/10.1007/978-3-319-96355-6
- Oemof documentation at https://oemof-solph.readthedocs.io/en/v0.4.1/index.html
- Oemof github at https://github.com/oemof

2. Installing LAEND v0.3.1

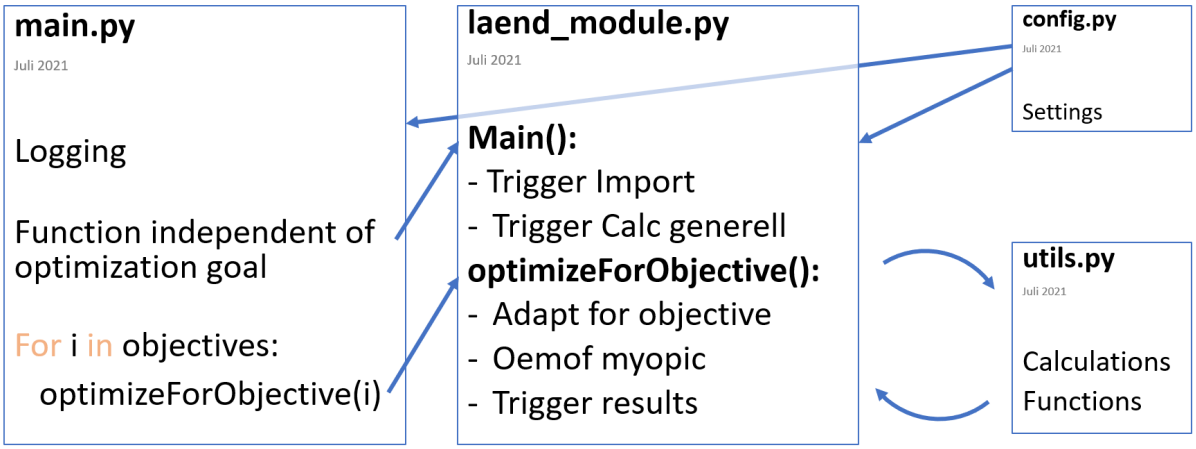
1. Install Python 3.7 or 3.8 (this might be done by installing Anaconda, see https://en.wikipedia.org/wiki/Anaconda_(Python_distribution))
2. Create your virtual environment (e.g. https://docs.anaconda.com/anaconda/navigator/tutorials/manage-environments/)
3. Install oemof 0.4.1 (https://oemof-solph.readthedocs.io/en/v0.4.1/index.html; https://www.youtube.com/watch?v=eFvoM36_szM)
4. Install a solver (LAEND assumes you use the CBC solver, though this can be changed in the config file)
5. Fetch LAEND from GitHub (https://github.com/inecmod/laendv031)
6. Install the python packages as listed in requirements.txt

This version comes with a reduced number of usable energy technologies in order to be used without an ecoinvent licence for LCA data. Therefore, all required LCA data is available in the folder ~\LCA.

You should now be ready to run LAEND.

3. File Structure of LAEND

LAEND is separated into four main files. They are connected as follows:



- Main.py
  Main file to run if you want to run LAEND
- Laend_module.py
  Two most important functions get triggered here:
  - Main(): This function runs all preparations that are independent of the optimization objective

    - optimizeForObjective(): This function gets an optimization objective and then runs the myopic optimization
- utils.py: Document that has all calculations/functions
- Config.py: This file contains all settings.

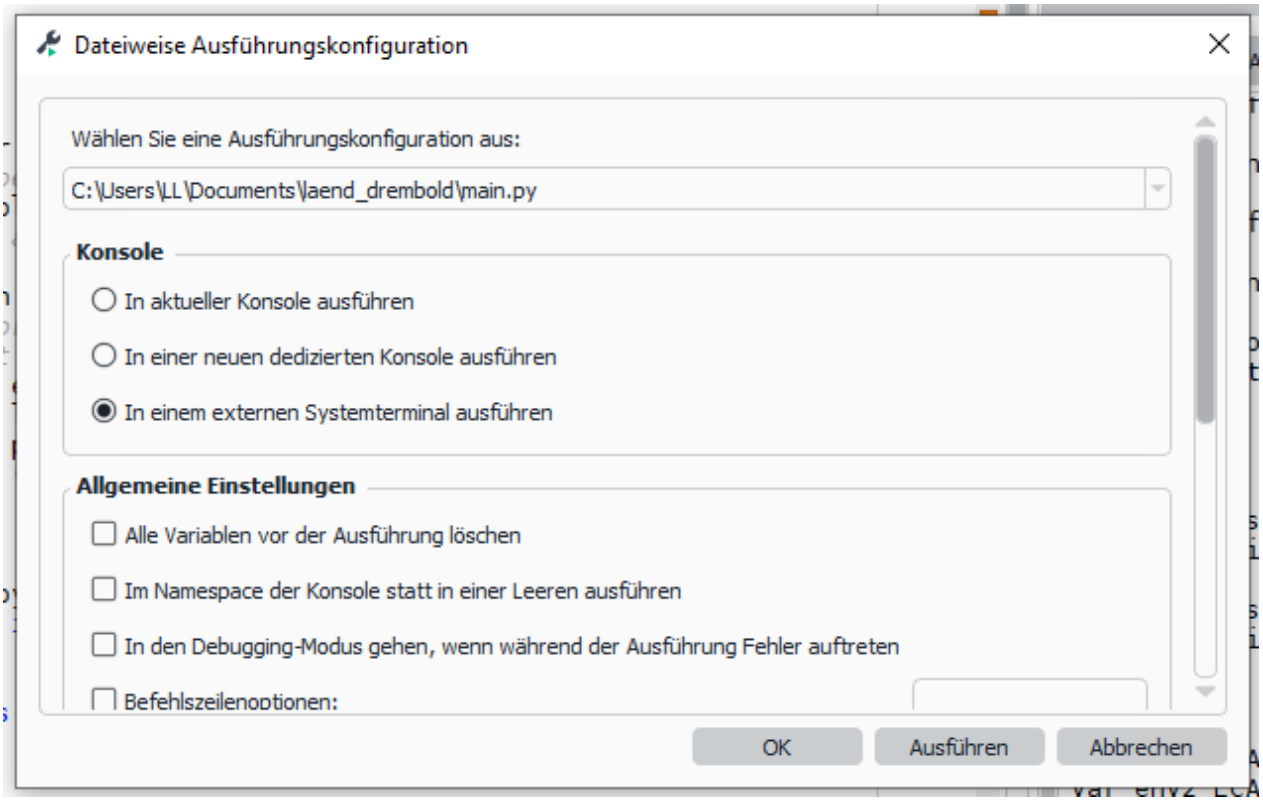
4. Running LAEND

The program can be run in two different ways:

- External System Terminal: Running it in an external system terminal does NOT allow a user to stop the program while it is running, but log files are visible while the program runs.
- In Spyder Control Panel: Running it in the Spyder Kernel using multiprocessing may result in logs not being written to the console, which means that the user cannot read the progress as the program calculates results. The program will still run and will write the logs to log files, but the progress is not immediately visible.

External System Terminal

1. Double click on main.py so that it is selected
2. Go to Run > File configuration (Ausführen > Dateiweise Konfiguration)
3. Choose to run the console in an external system terminal. Make sure the file mentioned in the first item is indeed main.py

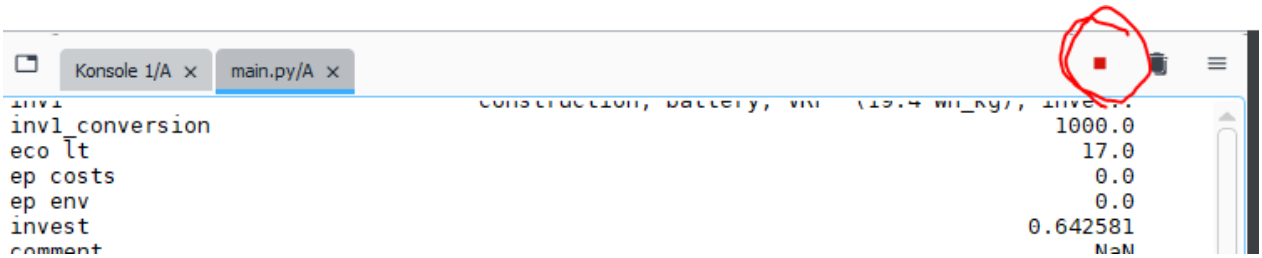


4. Click on run/Ausführen
5. A console window will open to run the program
6. Should the program not show any progress for a few minutes, try hitting enter.

In Spyder Console

1. Open main.py in Spyder

2. Click “run file”
3. If you may want to interrupt the execution click the red square



5. Configuring the Energy System Model

5.1. Scenario.xlsx

Scenario.xlsx is the main input table for LAEND. This chapter explains the input fields in detail.

Basic yes/no inputs:

- 0 = False
- 1 = True

In general, all input data should consistently be put in for the following units:

- Energy: kWh
- Power: kW
- Financial data: €
- Emissions: kg CO_2-Eq

To convert LCA data to these units in some cases conversion factors are necessary. For example, if LCA data for commodity sources like wood chips is given in impact value per kg a conversion factor that is multiplied with var_env1 to convert LCA data to impact value per kWh is applied. E. g. 39 MJ/m3 for natural gas listed in ecoinvent. Conversion factor = 1/39*3.6

Buses

Buses describe the nodes of the energy system. If the bus is not created, the energy system can neither obtain energy from this bus nor feed into it. It’s recommended to keep buses active, even if you don’t use the bus. Too many active buses do not impact the system, but forgetting to activate a bus becomes an issue.

Field	Description	Type of input
label	Name of the bus	string
active	Should the bus be included in the optimization? 0 = False, ignore this row 1 = True, use this bus	Integer (0 or 1)
excess	Should an excess sink be created, set this to active for buses that feed directly into the demand items!	Integer (0 or 1)
shortage	Should a shortage source be created? Creation of a shortage can prevent an unsolvable model.	Integer (0 or 1)

Demand

The items in demand describe energy sinks. The optimization problem seeks to satisfy the requirements of these energy sinks within given constraints.

Field	Description	Type of input	Limitation
label	Name of the sink / demand item	string	Must be unique
active	Should this row be included in the optimization?	Integer	Only 0 or 1
from	Label of the bus that this sink obtains its energy from	string	Must be included in buses/label
nominal value	Multiplication factor The fixed timeseries that is associated with the demand item is multiplied with this factor. If, for example, the fixed timeseries is for a single household, a nominal value of 70 would scale the load curve to 70 identical households	Float	>= 0
fixed	Is the flow to this sink fixed (e.g. load curve)? Currently only fixed flows for demand are integrated! Inputting 0 may result in errors.	Integer	Only 0 or 1
emission_baseline [kgCO2eq/kWh]	Specific emission factor for the energy type from the year 1990 (as current goals are based on emissions from 1990, see thesis) In thesis: $hef_{e,a}$	Float	>= 0
climate neutral by	Year in which climate neutrality should be reached	Integer	> 0
comment	Not used in LAEND, for reference only		

Commodity_Sources

Commodity sources describe energy sources that do not require an investment, but can be purchased based on the unit, e.g. electric energy from the grid. They are set up as sources in oemof/LAEND.

Field	Description	Type of input	Limitation
label	Name of the source / commodity source	string	Must be unique
active	Should this row be included in the optimization?	Integer	Only 0 or 1
to	Label of the bus that this source feeds into	string	Must be included in buses/label
variable costs	Variable cost per sourced unit In thesis: var_{cost}	Float	>= 0
var_env1	Name of LCA dataset for variable environmental impacts	string	
var_env1_conversion	Conversion factor that is multiplied with var_env1 to convert LCA data to impact per kWh.	Float	>= 0
comment	Not used in LAEND, for reference only		
fuel_based	Is this commodity source fuel based?	Integer	Only 0 or 1

Renewables

These items describe renewable energy sources, for example wind, that have a fixed flow based on atmospheric conditions. These renewables are set up as oemof sources with a fixed flow in LAEND.

Field	Description	Type of input	Limitation
label	Name of the renewable source	string	Must be unique
active	Should this row be included in the optimization?	Integer	Only 0 or 1
to	Label of the bus that this source feeds into	string	Must be included in buses/label
variable costs	Variable cost per sourced unit In thesis: var_{cost}	Float	>= 0
var_env1	Name of LCA dataset for variable environmental impacts	string	
var_env1_conversion	Conversion factor to convert to one kWh	Float	>= 0
om	Operation and maintenance cost for one kW and one year In thesis: $inv_{o\&m}$	Float	>= 0
invest	Investment costs for one kW In thesis: inv_{inst}	Float	>= 0
lifetime	Lifetime of technology	Integer	> 0
inv1	Name of LCA dataset for investment	String	
inv1_conversion	Conversion factor to get LCA for 1 kW capacity	Float	> 0
fixed	Is flow from this source fixed? Should always be 1 for renewables	Integer	Only 0 or 1

Storages

Storage items include all technologies that can store energy.

Field	Description	Type of input	Limitation
label	Name of the storage (heat or electricity storages can be modelled)	string	Must be unique
active	Should this row be included in the optimization?	Integer	Only 0 or 1
bus	Label of the bus that this storage takes energy from and feeds back into In the current version, the storage can only take energy and feed it into the same bus. It cannot take energy from one bus and feed it into another bus.	string	Must be included in buses/label
balanced	Should storage level be the same at the end of the optimization period as at the beginning? E.g. if 1 (=True) then battery level at the end of the year must be at level on January 1, when it is first bought	Integer	Only 0 or 1
capacity loss	How much (fraction) of the capacity is lost within one hour?	Float	0 <= x <= 1
efficiency inflow	How much of the energy that the bus feeds into the storage actually ends up in storage?	Float	0 <= x <= 1
efficiency outflow	How much of the energy that leaves the storage actually ends up back at the bus?	Float	0 <= x <= 1

initial capacity	What is the capacity when the storage is purchased?	Float	0 <= x <= 1, must be larger than capacity min
capacity min	Minimum storage capacity, e.g. LFP batteries should not be deeply discharged below 10% of storage capacity	Float	0 <= x <= 1
capacity max	Maximum storage capacity	Float	0 <= x <= 1, must be larger than capacity min
invest_relation_ input_capacity	How quickly can energy be passed to storage in relation to the storage size? E.g. what is the charging speed (in kW) in relation to the storage size (kWh)	Float	0 < x <= 1
invest_relation_outp ut_capacity	How quickly can energy leave storage?	Float	0 < x <= 1
variable input costs	Variable costs associated with moving energy to storage	Float	>= 0
variable output costs	Variable costs associated with moving energy out of storage	Float	>= 0
inv1	Name of LCA dataset for investment	String	
inv1_conversion	Conversion factor to get LCA for 1 kWh capacity	Float	> 0
om	Operation and maintenance cost for one kWh capacity and one year In thesis: $inv_{o\&m}$	Float	>= 0
invest	Investment costs for one kWh In thesis: inv_{inst}	Float	>= 0
lifetime	Lifetime of technology	Integer	> 0
comment	Free field, not used in optimization	String	

Cycle times are not included in LAEND. The system assumes full functionality until the end of the lifetime is reached.

Transformers_in

Transformers_in include all items that transform energy from one bus to another. The investment mode is placed on the input flow. In LAEND, a transformer_in can have one energy input and one or two energy outputs.

Important to note: since all investment data applies to the energy input they should be provided in €/kW_{input}

Field	**Description**	**Type of input**	**Limitation**
label	Name of the transformer	string	Must be unique
active	Should this row be included in the optimization?	Integer	Only 0 or 1
from1	Label of the primary energy input bus	string	Must be included in buses/label
var_from1_costs	Variable costs of from1 → transformer flow In thesis: var_{cost}	Float	>= 0
to1	Label of the primary energy output bus	string	Must be included in buses/label

var_to1_costs	Variable costs of transformer → to1 flow In thesis: var_{cost}	Float	>= 0
conversion_factor1	Conversion factor of energy input to energy that feeds into to1	Float	>= 0
var_env1	Name of LCA product system for variable environmental impacts	string	
var_env1_conversion	Conversion factor to convert to one kWh	Float	>= 0
to2	Label of the primary energy output bus *Can be left blank if not applicable*	string	Must be included in buses/label
var_to2_costs	Variable costs of transformer	Float	>= 0
conversion_factor2	Conversion factor of energy input to energy that feeds into to2	Float	>= 0
var_env2	Name of LCA product system for variable environmental impacts	string	
var_env2_conversion	Conversion factor to convert to one kWh	Float	>= 0
om	Operation and maintenance cost for one kW capacity and one year In thesis: $inv_{o\&m}$	Float	>= 0
invest	Investment costs for one kW (input capacity!) In thesis: inv_{inst}	Float	>= 0
lifetime	Lifetime of technology	Integer	> 0
inv1	Name of LCA product system for investment	String	
inv1_conversion	Conversion factor to get LCA for 1 kW of capacity (input capacity!)	Float	> 0
fixed_cop	Does this technology have a fixed (different for all hours in the year) conversion_factor1?	Integer	Only 0 or 1
fuel_based	Is this transformer fuel based? 0 sets inv_env and var_env to zero for model specific climate neutrality calculation	Integer	Only 0 or 1
comment	Free field, not used in optimization	String	

Transformers_out

Transformers_out include all items that transform energy from one bus to another. The investment mode is placed on the output flow. In LAEND, a transformer_out can have one or two energy inputs and one energy output.

Important to note: since all investment data applies to the energy output they should be provided in €/kW_{output}

Field	**Description**	**Type of input**	**Limitation**
label	Name of the transformer	string	Must be unique
lifetime	Lifetime of technology	Integer	> 0
active	Should this row be included in the optimization?	Integer	Only 0 or 1
from1	Label of the primary energy input bus	string	Must be included in buses/label
var_from1_costs	Variable costs of from1 → transformer flow In thesis: varcost	Float	>= 0

from2	Label of the socondary energy input bus	string	Must be included in buses/label
to1	Label of the primary energy output bus	string	Must be included in buses/label
conversion_factor1	Conversion factor of energy input to energy that feeds into to1	Float	>= 0
fixed_cop	Does this technology have a fixed (different for all hours in the year) conversion_factor1?	Integer	Only 0 or 1
var_to1_costs	Variable costs of transformer → to1 flow In thesis: varcost Variable costs for transformer → to2 flow not yet implemented!	Float	>= 0
var_env1	Name of LCA product system for variable environmental impacts	string	
var_env1_conversion	Conversion factor to convert to one kWh	Float	>= 0
om	Operation and maintenance cost for one kW capacity and one year In thesis: $inv_{o\&m}$	Float	>= 0
invest	Investment costs for one kW (output capacity!) In thesis: inv_{inst}	Float	>= 0
inv1	Name of LCA product system for investment	String	
inv1_conversion	Conversion factor to get LCA dataset for 1 kW of capacity (output capacity!)	Float	> 0
fuel_based	Is this transformer fuel based? 0 sets inv_env and var_env to zero for model specific climate neutrality calculation	Integer	Only 0 or 1
comment	Free field, not used in optimization	String	

Timeseries

Timeseries is the sheet that allows for the import of fixed flows e. g. load curves. Column A lists the hourly timestamps in the following format: yyyy-mm-dd HH:MM:SS e. g. 2016-01-01 03:00:00

All other columns can be filled with fixed timeseries. To create a fixed load curve for a renewable energy source or demand item, create the column title in the format label.fix. To create a fixed cop, e. g. for an air-water heat pump, name the column label.cop. Here is an example:

timestamp	**load_el.fix**	**wind.fix**	**heat_pump_a_w.cop**
2016-01-01 00:00:00	76.7409	0.08515	3.7548
2016-01-01 01:00:00	76.9936	0.08386	3.76269
2016-01-01 02:00:00	69.6969	0.08228	3.77176

For automatically generated timeseries, see section “Updating fixed demand load curves”.

5.2. Configurations in config.py

Config.py is the main configuration file for the optimization. In the first part model specific settings are done, in the second part location specific settings are done.

Temporal settings

- Define the **start year** and the **end year**. Leave the calc_start and calc_end unchanged.
- For testing purposes, the **number of timesteps** can be changed. Chose *None* to calculate the full year. Use a number smaller than 8760 to reduce the number of timesteps. Emission constraints do not scale to this limited view! Some other functionalities may also not work as expected.
- **Aux_years:** Set to true to use representative years and select the steps. Use false to optimize each year individually (see thesis for more info). Choosing aux years with a period of 5 years and starting in the year 2022 will mean, that the first period runs from 2022 to 2026. The year 2022 will serve as the representative year.

Set Optimization Objectives

- **Objective:** Activate an objective by removing the #. Deactivate, by placing a # in the row before the name.
  Possible objectives include the following:
  - System costs.
  - Environmental impact in the form of a single EF 2.0 indicator.
  - Total environmental impacts, named JRCII, which applies the EF 2.0 normalization and weighting factors to arrive at a single factor.
  - Combination of system costs and JRCII. Costs are normalized. Costs and aggregated environmental impacts are each weighted individually (default: 50 %).
  - Combination of normalized system costs and environmental impacts at an equilibrium weighting

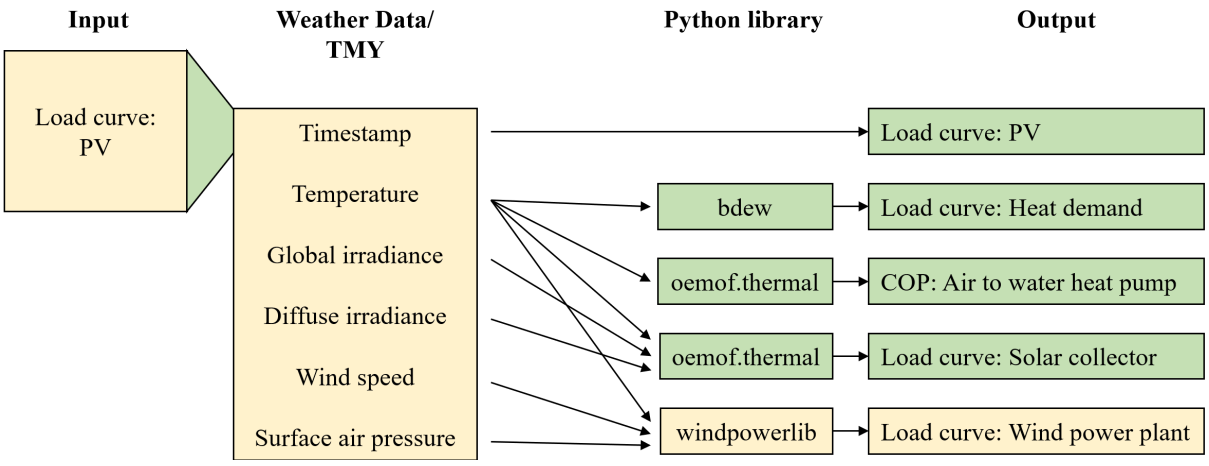
Set multiprocessing

Multiprocessing of several objectives can be enabled or disabled. True if several objectives should run in parallel to speed up the calculation time, False for testing and to use debug mode, since debug messages are not displayed during multiprocessing.

Typical meteorological year (TMY)

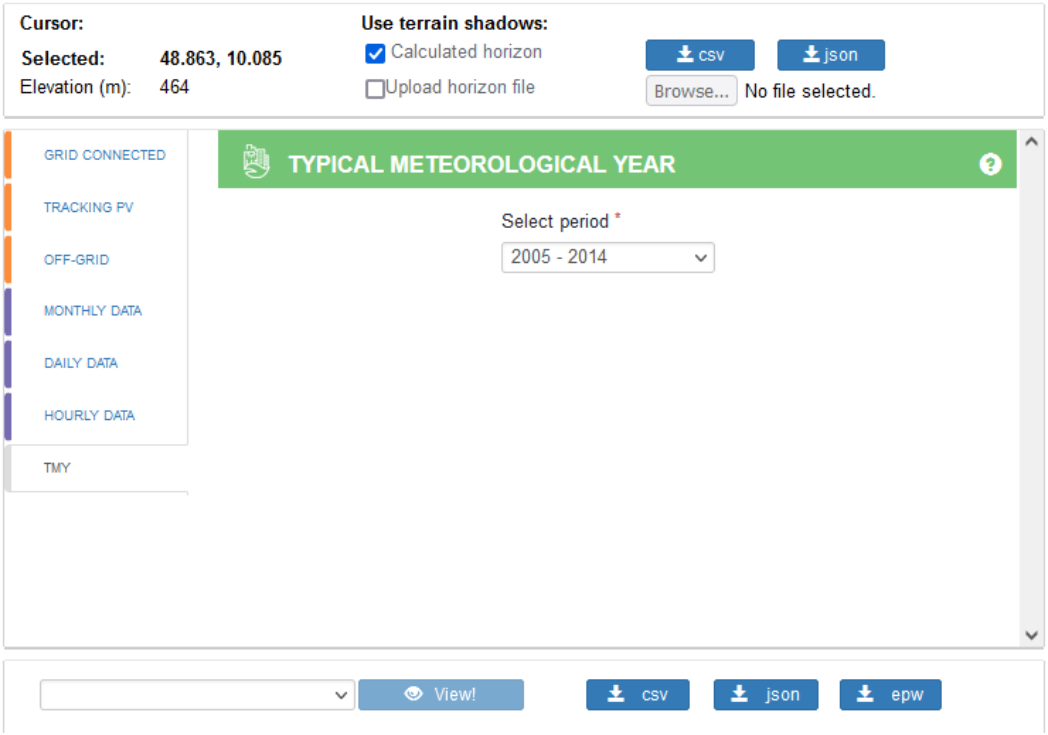
The TMY has to be adapted to the location of the modelled energy system. Change the file location and name of the TMY here.

Several fixed timeseries can be created with a TMY. The TMY acts as a filter for PV data and as the source to generate fixed curves for all other types of timeseries, see image. The colors only indicate what existed in LAEND and what was newly developed for LAEND (see thesis) and are not relevant to how the system works now.



If you want to update the TMY to a new location, go to PVGIS and download the csv. Replace the file tmy_XX.XXX_X.XXX_2007_2016.csv in folder \in\pvgis.

PVGIS: https://re.jrc.ec.europa.eu/pvg_tools/en/#TMY



Select period 2007-2016 for the selected location and download csv.

Location settings

Adapt timezone, longitude and latitude.

Updating fixed demand load curves

In general, the following settings are True/False. If set to true, then the program imports the corresponding csv. This file is then modified and saved as a timeseries in ~\runs\[time]\files\[time]_scenario.xlsx. Please note, that the original scenario.xlsx is never changed by the program. Instead, it changes the copy of the file in the corresponding run folder. The

corresponding *varname* must be exactly the same as the corresponding *label* in the sheet *demand* in scenario.xlsx.

- **Update_electricity_demand:** 'in/el_demand_bdew.csv'
- **Update_heat_demand:** 'in/heat_demand_bdew.csv'

If update_electricity_demand is set to True a corresponding file has to be available in the in folder, which has to be created externally with electricity_demand_example.py. Heat demand profiles will be generated internally if update_heat_demand is set to True.

Input parameters for bdew heat demand

The heat demand is calculated based on the TMY and typical gas demand.

- **Separate_heat_water:** separates space heating and hot water into two load curves. The development included subdividing the total heat load curve into space heating (SH) and hot water (HW) demand to account for different circuit flow temperatures. This becomes relevant when low-temperature heat sources like heat pumps are integrated into the energy system. The average hourly heat demand in July and August is assumed to be equal to the constant HW demand. If this average exceeds the total required thermal energy for one given hour, the HW demand equals the total heat demand during this period. The SH curve is derived by subtracting the HW demand from the total heat load curve. With enabled separation two buses for high and low temperature heat and individual transformers feeding these buses are needed. Setting this to False requires one load curve, one demand profile and only one bus.
- **BDEW input settings:** These settings go into the function that employs the bdew python library to calculate the load curve. Annual demand for private buildings separated in single family houses and multifamily houses and for trade, commerce, service have to be set.
- building_class: 1 – 11; class of building according to bdew classification for share of new and old buildings based on housing unit according to https://www.eko-netz.de/files/eko-netz/download/3.5_standardlastprofile_bgw_information_lastprofile.pdf

	Altbauanteil		mittl. Anteile von	
Klasse	von	bis	Altbau	Neubau
1	85,5%	90,5%	88,0%	12,0%
2	80,5%	85,5%	83,0%	17,0%
3	75,5%	80,5%	78,0%	22,0%
4	70,5%	75,5%	73,0%	27,0%
5	65,5%	70,5%	68,0%	32,0%
6	60,5%	65,5%	63,0%	37,0%
7	55,5%	60,5%	58,0%	42,0%
8	50,5%	55,5%	53,0%	47,0%
9	45,5%	50,5%	48,0%	52,0%
10	40,5%	45,5%	43,0%	57,0%
11			75,0%	25,0%

Altbau bis 1979

building_wind_class: 0=not windy, 1=windy

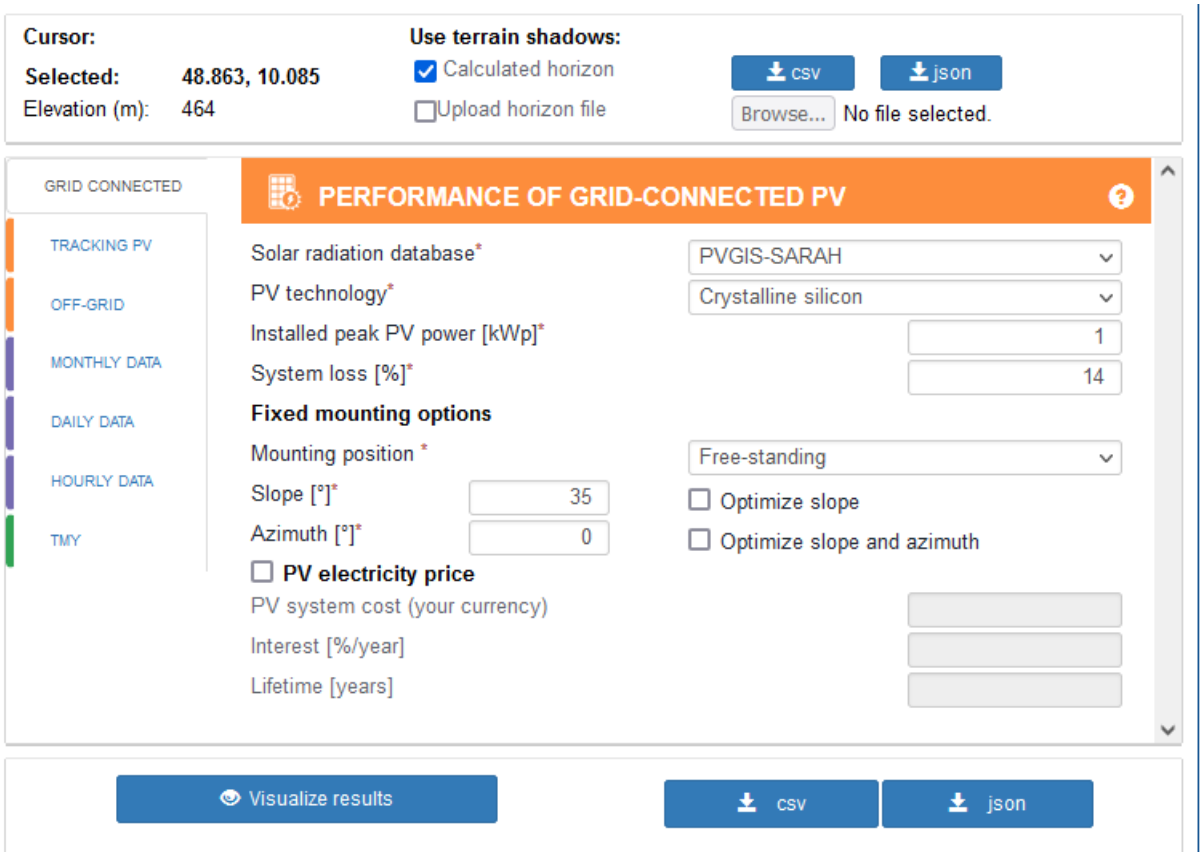
Updating Fixed Renewable Energy Load Curves

In general, the following renewable energy settings always contain the following:

- Update_technology: True/False, determines if the load curves in ~\runs\[time]\files\[time]_scenario.xlsx get updated
- Filename_technology: path to the necessary file, please watch whether it's an csv or xlsx file before changing this!
- Varname_technology: This string must be exactly the same as the label in sheet renewables in scenario.xlsx for the update to work.

- **PV:** Setting this to true will import the new pvgis file and update in/pvgis_tmy.csv. Currently, two variables can be equipped with the same load curve.

  Update PV data
  To update PV data, proceed as with the TMY, but download the grid connected data. Make sure that the data contains all years that the TMY uses. Save the file and make sure that the file in config.py links to it.



LCA Update

By default, LAEND will use the excel files in folder \LCA as input data. If a technology in scenario.xlsx does not have an associated file, the program will automatically try to connect to openLCA to get the correct file. If you want to force the program to update all LCA files from the database, use these settings:

- **Update_LCA_data:** False (default); use True to force an update (only possible with an ecoinvent licence for openLCA; this version is usable without)
- **LCA_impact_method:** This is the LCA impact method that is pulled from openLCA. Changing the impact method is not as easy as changing this setting. All normalisation and weighting files must have identical headers. Also check utils.py function *def determineCweightForSolver(objective):* as this led to bugs before.
- **System_impacts_index:** Do NOT change this list unless you're updating the LCIA method!

Normalization & Weighting

These settings are necessary for the multi-objective optimization.

- **Filename_weight_and_normalisation:** Excel filename for the LCA normalisation and weighting file. Ensure that headings here match the LCA excel files (see above)
- **Normalization_cost_gdp:** The costs get normalized by gdp, change it here.
- **Normalization_per_person:** Instead of using a global value to normalize, the values can also be normalized per person. If set true, the normalization values are divided by the global population (variable "normalization_person_population").
- **Weight_cost_to_env:** The objective "EnvCosts" allows for an individual weighting of aggregated environmental impacts and costs. Set the ratio of costs to environmental impacts here.
- **Weight_cost_to_env_equilibrium**: 1/17 if equal weighting between every single goal for multi-objective "Equilibrium"

Emission Constraints

Please take a look at the thesis for the functionality of the emission constraint, emission factor and model-specific climate neutrality.

- **Emission_constraint:** True to activate, False to run without a constraint
- **Ec_horizon:** Window of foresight for the emission constraint in years (must be integer>1)
- **Ec_impact_category:** The impact category that the emission factor is pulled from. Ensure this is an exact match with the openLCA export data.
- **Ec_buffer:** If the optimization for one year with an emission constraint failed, then the same year runs again with an emission constraint that is increased by this buffer. After 3 tries the program aborts.
- **Ef_fuel_based_only:** Is the emission factor only applied to fuel based technologies? Whether or not a technology is fuel based can be set in scenario.xlsx

Defining Climate Neutrality

- **Def_cn_calculate_climate_neutrality**: True/False to activate/deactivate calculation for climate neutrality
  If set to false, it will only calculate based on the values given in scenario.xlsx
- **Def_cn_include_investment:** True/False. Should the investment impacts be included in the calculation for climate neutrality? If set to false, then all manufacturing impacts are set to zero for the climate neutrality optimization

- **Def_cn_year_climate_neutrality**: Integer. If scenario.xlsx only lists emission targets until a specific year, for example 2030, then this value determines the end of the linear decrease. However, if the model specific climate neutrality (e.g. 1t CO2/year) > last given emission constraint (0.5kg CO2/kWh for an annual demand of 1000kWh), then it will linearly increase the emission limit to reach the higher climate neutrality. This does not influence the final outcome, as the higher value (of climate neutrality or the political goal) is selected as the emission constraint (see thesis).
- **Def_cn_fuel_based_only:** True/False. Use this setting to employ only fuel-based technology when calculating climate change. If set to true, all technologies set to non-fuel-based in scenario.xlsx will have variable environmental impacts of zero.

Financial Settings

- **InvestWacc:** This weighted average cost of capital (wacc) is used to calculate the annuity of investments and employs the function oemof.economics.annuity
- **Invest_min_threshhold (in kW/kWh):** Currently, the program can use extremely small capacities. This threshold does not limit the systems' ability to employ new small capacities. However, if a capacity below this threshold is selected in year 1, then it does not get passed on to the next calculation year as an employed capacity.
- **InvestTimeSteps:** Do not change!
- **Invest_sell_if_unused:** True/False. If set to false, then the technology keeps getting passed to the next year's optimization until the end of life is reached. Whether this functionality works by setting on True must be tested!

Technical Settings

Logging level

Changing these technical settings should be handled with caution. Most importantly, this section contains information about the employed solver. No testing has been done with activated solver options.

5.3. Logging

Logging runs through the oemof library *logging*. Unfortunately, this library is fairly hardcoded. This means that if you want to change the logging level e. g. only show info tags and remove all debug messages, this has to be adjusted in oemof. Locate the site packages > oemof > tools > logger.py. In line 95, you can set the level. Please note that this applies to the screen level and to the file level.

Set to DEBUG to get all messages, set to INFO to get information messages.

```
if logpath is None:
    logpath = extend_basic_path('log_files')

file = os.path.join(logpath, logfile)

log = getLogger('')

# Remove existing handlers to avoid interference.
log.handlers = []
log.setLevel(DEBUG)
```

```
if file_format is None:
    file_format = (
        "%(asctime)s - %(levelname)s - %(module)s - %(message)s")
file_formatter = Formatter(file_format, file_datefmt)
```

Changing the settings in laend_module.py will not affect the results.

6. Results

The program saves all results in the same folder with all the other program files. In the folder *runs*, a new folder with the date and time of the optimization run is created. Here, the following can be found:

- **Laend_config.py:** copy of the config file for later reference
- **Files:**
    - [time]_scenario.xlsx: copy of file at start of run for later reference
    - Results for *objective_[time]*.xlsx: Result for this single objective
        - Impacts: Table of the environmental and financial impacts. In the case of representative (aux) years, the results from the one year will be multiplied with the number of years until the next optimization year/end of the modeling period. Highlight the entire sheet and create a pivot table for easier analysis.
        - Tech: Shows the capacity and energy flows for each technology.
    - Flows for *objective_[time]*.csv: Dispatch results for each optimized hour.
    - Results for total: If all optimizations were successful, then the files *Results for objective.xlsx* are aggregated into a single excel table for easier analysis.
- **Logs:** These log files help to find errors. While running LAEND in the (Spyder) console they also help to see if the program is still running while multiprocessing.
- **Oemof_dumps:** These are all of the optimization runs that the program optimized and saved. These tend to get quite large, so deleting unnecessary runs every now and then is recommended. These dumps can be restored to see individual oemof results.