Spreadsheet Energy System Model Generator Documentation

Release v0.1.1

Jan 12, 2021

Structure of Energy System

The Spreadsheet Energy System Model Generator allows the modeling and optimization of energy systems without the need for programming skills. The components defined in this spreadsheet are defined with the included Python program and the open source Python library “oemof”, assembled to an energy system and optimized with the open source solver “cbc”. The modeling results can be viewed and analyzed using a browser-based results output.

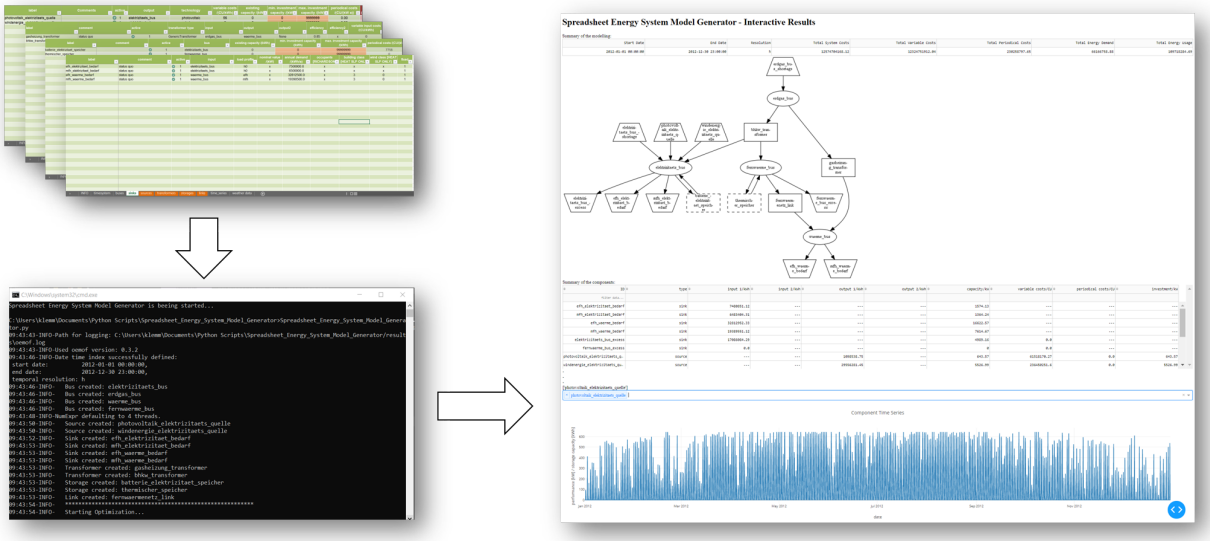

Fig. 1: From an input spreadsheet to interactive modelling results: The Spreadsheet Energy System Model Generator

CHAPTER 1

Structure of Energy System

1.1 Structure of Energy Systems

Energy systems in the sense of the Spreadseet Energy System Model Generator are designed according to the specifications of the oemof library. Accordingly, energy systems can be represented with the help of mathematical graph theory. Thus, energy systems are exemplified as "graphs" consisting of sets of "vertices" and "edges". In more specific terms, vertices stand for components and buses while directed edges connect them. The status variable of the energy flow indicates which amount of energy is transported between the individual nodes at what time. Possible components of an oemof energy system are

- sources,
- sinks,
- transformers, and
- storages.

Buses furthermore form connection points of an energy system. The graph of a simple energy system consisting of each one source, one transformer, one sink, as well as two buses, could look like the example displayed in the following figure.

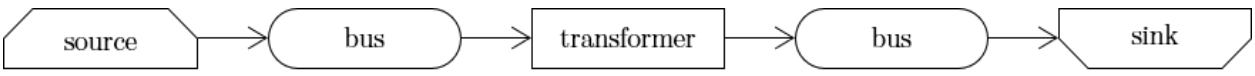

Fig. 1: Graph of a simple energy system, consisting of one source, two buses, one transformer, and one a sink.

An oemof energy system must be in equilibrium at all times. Therefore sources must always provide exactly as much energy as the sinks and transformer losses consume. In turn, the sink must be able to consume the entire amount of energy supplied. If there is no balance, oemof is not able to solve the energy system.

1.1.1 Buses

The modelling framework oemof does not allow direct connections between components. Instead, they must always be connected with a bus. The bus in turn can be connected to other components, so that energy can be transported via the bus. Buses can have any number of incoming and outgoing flows. Buses can not directly be connected with each other. They do not consider any conversion processes or losses.

1.1.2 Sources

Sources represent the provision of energy. This can either be the exploitation of of an energy source (e.g. gas storage reservoir or solar energy, no energy source in physical sense), or the simplified energy import from adjacent energy systems. While some sources may have variable performances, depending on the temporary needs of the energy system, others have fixed performances, which depend on external circumstances. In the latter case, the exact performances must be entered to the model in form of time series. With the help of oemofs "feedinlib" and "windpowerlib", electrical outputs of photovoltaik (pv)-systems and wind power plants can be generated automatically. In order to ensure a balance in the energy system at all times, it may be useful to add a "shortage" source to the energy system, which supplies energy in the event of an energy deficit. In reality, such a source could represent the purchase of energy at a fixed price.

Photovoltaic Systems

The following Figure sketches the fractions of radiation arriving at a PV-module as well as further relevant parameters.

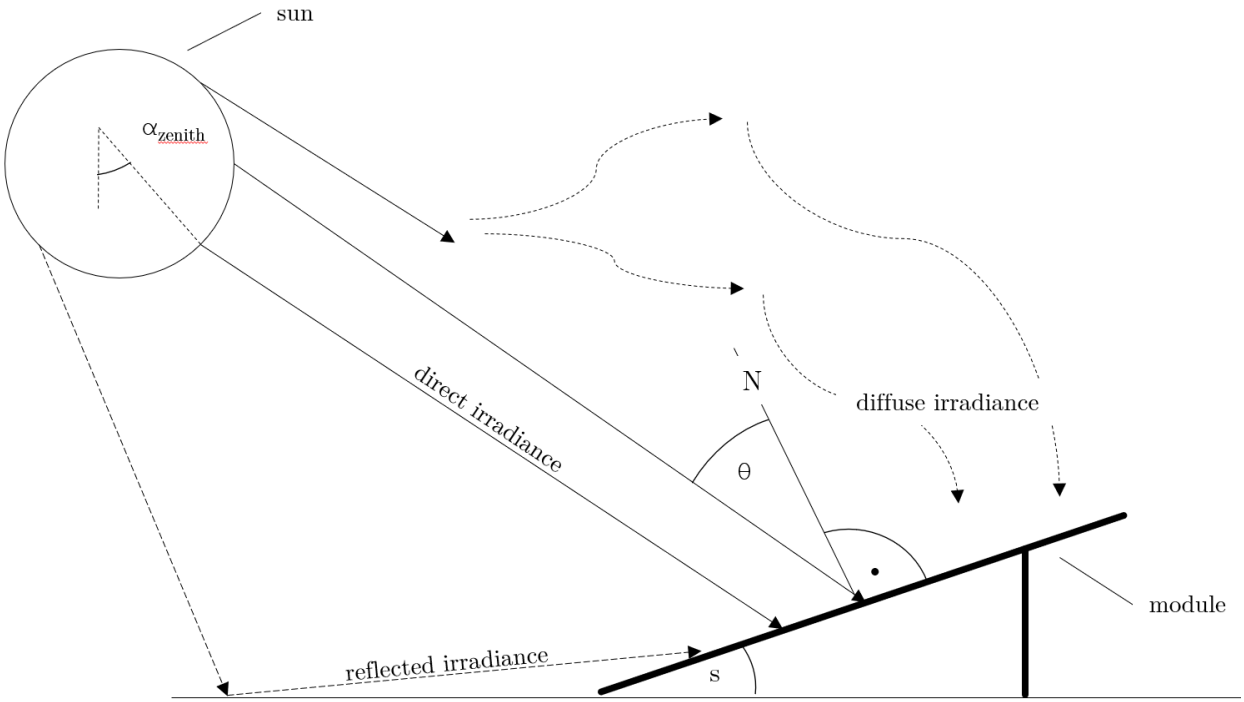

Fig. 2: Radiation on a photovoltaic module.

The global radiation is composed of direct and diffuse radiation. The "direct horizontal irradiance" dirhi is the amount of sun radiation as directly received by a horizontal surface. The "diffuse horizontal irradiance" dhi is the share of radiation, which arrives via scattering effects on the same surface. A part of the global radiation is reflected on the ground surface and can thus cause an additional radiation contribution on the photovoltaic module. The amount of the reflected part depends on the magnitude of the albedo of the ground material. Exemplary albedo values are listed in the following table.

Wind Turbines

For the modelilng of wind turbines, the weather data set must include wind speeds. The wind speeds must be available for a measurement height of 10 m in the unit m/s.

The system data of the wind turbine to be modelled are obtained from the "oedb" database.

1.1.3 Sinks

Sinks represent either energy demands within the energy system or energy exports to adjacent systems. Like sources, sinks can either have variable or fixed energy demands. Sinks with variable demands adjust their consumption to the amount of energy available. This could for example stand for the sale of surplus electricity. However, actual consumers usually have fixed energy demands, which do not respond to amount of energy available in the system. As with sources, the exact demands of sinks can be passed to the model with the help of time series.

In order to ensure a balance in the energy system at all times, it may be appropriate to add an "excess" sink to the energy system, which consumes energy in the event of energy surplus. In reality, this could be the sale of electricity or the give-away of heat to the atmosphere.

Standard Load Profiles

Oemof's sub-library demandlib can be used for the estimation of heat and electricity demands of different consumer groups, as based on German standard load profiles (SLP). The following electrical standard load profiles of the Association of the Electricity Industry (VDEW) can be used:

Profil	Consumer Group
H0	households
G0	commercial general
G1	commercial on weeks 8-18 h
G2	commercial with strong consumption in the evening
G3	commercial continuous
G4	shop/hairdresser
G5	bakery
G6	weekend operation
L0	agriculture general
L1	agriculture with dairy industry/animal breeding
L2	other agriculture

The following heat standard load profiles of the Association of Energy and Water Management (BDEW) can be used:

Profile	House Type
EFH	single family house
MFH	multi family house
GMK	metal and automotive
GHA	retail and wholesale
GKO	Local authorities, credit institutions and insurance companies
GBD	other operational services
GGA	restaurants
GBH	accommodation
GWA	laundries, dry cleaning
GGB	horticulture
GBA	bakery
GPD	paper and printing
GMF	household-like business enterprises
GHD	Total load profile Business/Commerce/Services

In addition, the location of the building and whether the building is located in a "windy" or "non-windy" area are taken into account for the application of heat standard load profiles. The following location classes may be considered:

Stochastic Load Profiles (Richardson Tool)

The use of standard load profiles has the disadvantage that they only represent the average of a larger number of households (> 200). Load peaks of individual households (e.g. through the use of hair dryers or electric kettles) are filtered out by this procedure. To counteract this, the Spreadsheet Energy System Model Generator offers the possibility to generate stochastic load profiles for residential buildings. These are generated on the basis of Richardsonpy. Thereby, an arbitrary number of different realistic load profiles is simulated under consideration of statistic rules. The mean value of a large-enough number of profiles should, again, result in the standard load profile. However, if calculations are continued using the individual values before averaging – as in the above calculation of costs – different values are obtained than when calculating with SLPs.

1.1.4 Transformers

Transformers are components with one ore more input flows, which are transformed to one or more output flows. Transformers may be power plants, energy transforming processes (e.g., electrolysis, heat pumps), as well as transport lines with losses. The transformers' efficiencies can be defined for every time step (e.g., the efficiency of a thermal powerplants in dependence of the ambient temperature).

Currently only Generic Transformers can be used within the Spreadsheet Energy System Model Generator. These may have one or more different outputs, e.g., heat and electricity. For the modelling, the nominal performance of a generic transformer with several outputs, the respective output ratios, and an efficiency for each output need to be known.

Heat Pumps

For the modelilng of heat pumps, different heat sources are considered so the weather data set must include different temperatures. The efficiency of the heat pump cycle process can be described by the Coefficient of Performance (COP). The heat pump automatically creates a heat source and a low temperature bus (see red bubble). So only a transformer and a electricity bus needs to be created. An example is shown in the following figure.

At the moment it is possible to use ground water, soil (vertical heat exchanger), surface water and ambient air as a heat source.

The heat pumps are implemnted by using "oemof.thermal" .

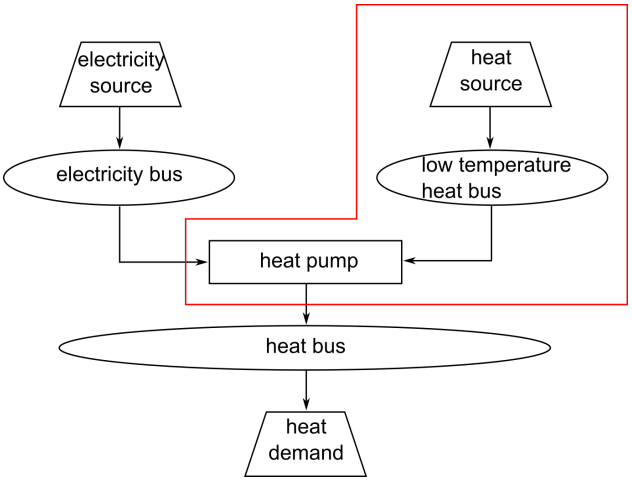

Fig. 3: Graph of a heat pump system.

1.1.5 Links

Links can be used to connect two buses or to display transport losses of networks. Links are not represented by a separate oemof class, they are rather represented by transformers. In order to map a loss-free connection between two buses, an efficiency of 1 is used. If a link is undirected, a separate transformer must be used for each direction. In an energy system, links can represent, for example, electrical powerlines, gas pipelines, district heating distribution networks or similar.

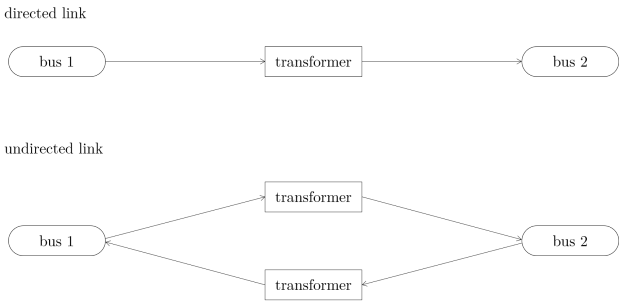

Fig. 4: Representation of directed and undirected links with oemof transformers

1.1.6 Storages

Storages are connected to a bus and can store energy from this bus and return it to a later point in time.

1.1.7 Investment

The investment costs help to compare the costs of building new components to the costs of further using existing components instead. The annual savings from building new capacities should compensate the investment costs. The investment method can be applied to any new component to be built. In addition to the usual component parameters, the maximum installable capacity needs to be known. Further, the periodic costs need to be assigned to the investment costs. The periodic costs refer to the defined time horizon. If the time horizon is one year, the periodical costs correspond to the annualized capital costs of an investment.

Non-Convex-Investments: While a linear programming approach is used for normal investment decisions, a mixed integer variable is defined for non-convex investment decisions. The model can thus decide, for example, whether a component should be implemented FULL or NOT. Mixed-integer variables increase the computational effort significantly and should be used with caution.

CHAPTER 2

Getting started

2.1 Installation

Warning: Warning! The installation has only been tested using Python 3.7.6 (64 bit)! Python 3.8 is currently not supported.

2.1.1 Windows

1. Install Python (version 3.5 or higher)

Note: Make sure to select "Add pyton to PATH" at the beginning of the Python installation.

- go to the Python download page
- chose a Python version (e.g., "Python 3.7.6") and click "download"
- download an installer (e.g., "Windows x86-64 executable installer")
- execute the installer on your computer

2. Download the Spreadsheet Energy System Model Generator from GIT as .zip folder.
3. Extract the .zip folder into any directory on the computer.

4. Download the CBC-solver from here
5. Extract the CBC solver into the folder of the Spreadsheet Energy System Model Generator
6. Install "Graphviz"
   - go to Graphviz download page
   - select and download the graphviz version for your device (e.g. graphviz-2.38.msi for Windows)

Note: Make sure to select the correct installation location for Graphviz!!

   - Execute the installation manager you just downloaded. Choose the following directory for the installation: "C:\Program Files (x86)\Graphviz2.38" (should be the default settings)
7. Execute the "Windows_installation.cmd" file.

Note: If you receive a "Your computer has been protected by Windows" error message, click "More Information," and then "Run Anyway".

8. The Spreadsheet Energy System Model Generator has been installed.

2.1.2 MacOS

1. Install Python (version 3.5 or higher)

Note: Make sure to select "Add pyton to PATH" at the beginning of the Python installation.

   - go to the Python download page
   - chose a Python version (e.g., "Python 3.7.6") and click "download"
   - download an installer (e.g., "Python 3.7.6 macOS 64-bit installer")
   - execute the installer on your computer
2. Download the Spreadsheet Energy System Model Generator from GIT as .zip folder.
3. Extract the .zip folder into any directory on the computer.
4. Excecute the "MacOS_installation.command" file.
5. The Spreadsheet Energy System Model Generator has been installed.

2.1.3 Linux

1. Install Python (version 3.5 or higher)
   - go to https://phoenixnap.com/kb/how-to-install-python-3-ubuntu/
   - use Python3.7 instead of Python3.8

Note: Make sure that the alias python3 is set to Python3.7.x. If not use update-alternatives to change it.

2. Download the Spreadsheet Energy System Model Generator from GIT as .zip folder.

3. Extract the .zip folder into any directory on the computer.
4. Install PIP

```
$ sudo apt-get install python3-pip
```

5. Install tkinter

```
$ sudo apt-get install python3.7-tk
```

6. Install Graphviz

```
$ sudo apt-get install graphviz
```

7. Install the CBC-Solver

```
$ sudo apt-get install coinor-cbc
```

8. Execute the “Linux_installtion.sh” file.
9. The Spreadsheet Energy System Model Generator has been installed.

2.2 Application of the Model Generator

1. Fill in the spreadsheet document according to the instructions in the “Using th Scenario File” chapter.

2. Execute the “SESMG_EXE.py” file in the main folder of the program. .. note:

```
If you receive a "Your computer has been protected by Windows" error message, click
↪"More Information," and then "Run Anyway".
```

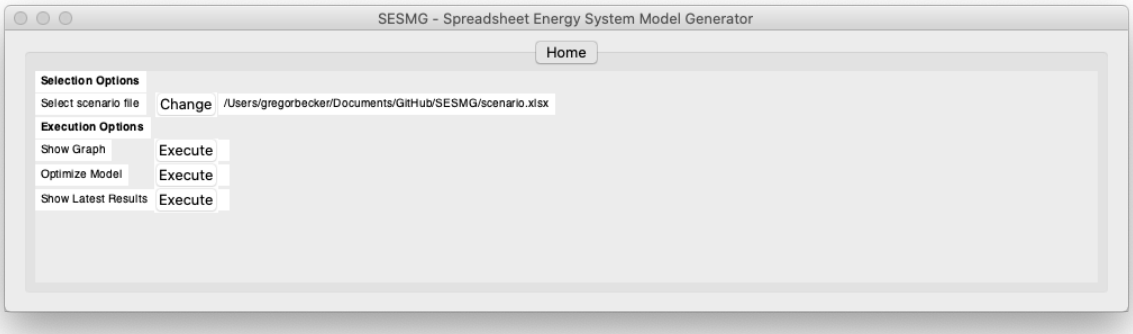

Fig. 1: The display may vary depending on the operating system.

Description of the GUI functions

Select the xlsx-scenario to be executed.

Now there are three functions:

1. Displays the currently selected xlsx-scenario as graph.
2. Modeling and optimization of the selected xlsx-scenario with subsequent output of results.
3. Display of the latest optimized scenario. .. note:

```
The detailed modelling results are also stored within the "results" folder.
```

2.3 Using the Scenario File

For the modeling and optimization of an energy system, parameters for all system components must be given in the model generator using the enclosed .xlsx file (editable with Excel, LibreOffice, ...). The .xlsx file is divided into nine input sheets. In the "timesystem" sheet, general parameters are defined for the time horizon to be examined, in the sheets "buses", "sinks", "sources", "transformers", "storages" and "links" corresponding components are defined. In the sheet "time series", the performance of individual components can be stored. In the "weather data" sheet, the required weather data is stored. When completing the input file, it is recommended to enter the energy system step by step and to perform test runs in between, so that potential input errors are detected early and can be localized more easily. In addition to the explanation of the individual input sheets, an example energy system is built step by step in the following subchapters. The input file for this example is stored in the program folder "examples" and viewed on GIT. The following units are used throughout:

- capacity/performance in kW,
- energy in kWh,
- angles in degrees, and
- costs in cost units (CU).

Cost units are any scalable quantity used to optimize the energy system, such as euros or grams of carbon dioxide emissions.

2.3.1 Timesystem

Within this sheet, the time horizon and the temporal resolution of the model is defined. The following parameters have to be entered:

- **start date**: start of the modelling time horizon. Format: "YYYY-MM-DD hh:mm:ss";
- **end date**: end date of the modelling time horizon. Format: "YYYY-MM-DD hh:mm:ss"; and
- **temporal resolution**: for the modelling considered temporal resolution. Possible inputs: "a" (years), "d" (days), "h" (hours) "min" (minutes), "s" (seconds), "ms" (mili seconds).
- **periods**: Number of periods within the time horizon (one year with hourly resolution equals 8760 periods).

start date	end date	temporal resolution
2012-01-01 00:00:00	2012-12-30 23:00:00	h

Fig. 2: Exemplary input for the time system

2.3.2 Buses

Within this sheet, the buses of the energy system are defined. The following parameters need to be entered:

- **label**: Unique designation of the bus. The following format is recommended: "ID_energy sector_bus".
- **comment**: Space for an individual comment, e.g. an indication of which measure this component belongs to.
- **active**: Specifies whether the bus shall be included to the model. 0 = inactive, 1 = active.

- **excess**: Specifies whether a sink is to be generated, which consumes excess energy. 0 = no excess sink will be generated; 1 = excess sink will be generated.
- **shortage**: Specifies whether to generate a shortage source that can compensate energy deficits or not. 0 = no shortage source will be generated; 1 = shortage source will be generated.
- **shortage costs/(CU/kWh)**: Assigns a price per kWh to the purchase of energy from the shortage source. If the shortage source was deactivated, the fill character "x" is used.
- **excess costs/(CU/kWh)**: Assigns a price per kWh to the release of energy to the excess sink. If the excess sink was deactivated, the fill character "x" is used.

label	comments	active	excess	shortage	shortage costs [CU/kWh]	excess costs [CU/kWh]
bus001_electricity_bus	Maßnahme A	1	0	1	0.0100	x
bus002_electricity_bus	Maßnahme B	1	1	0	x	0.0100

Fig. 3: Exemplary input for the buses sheet

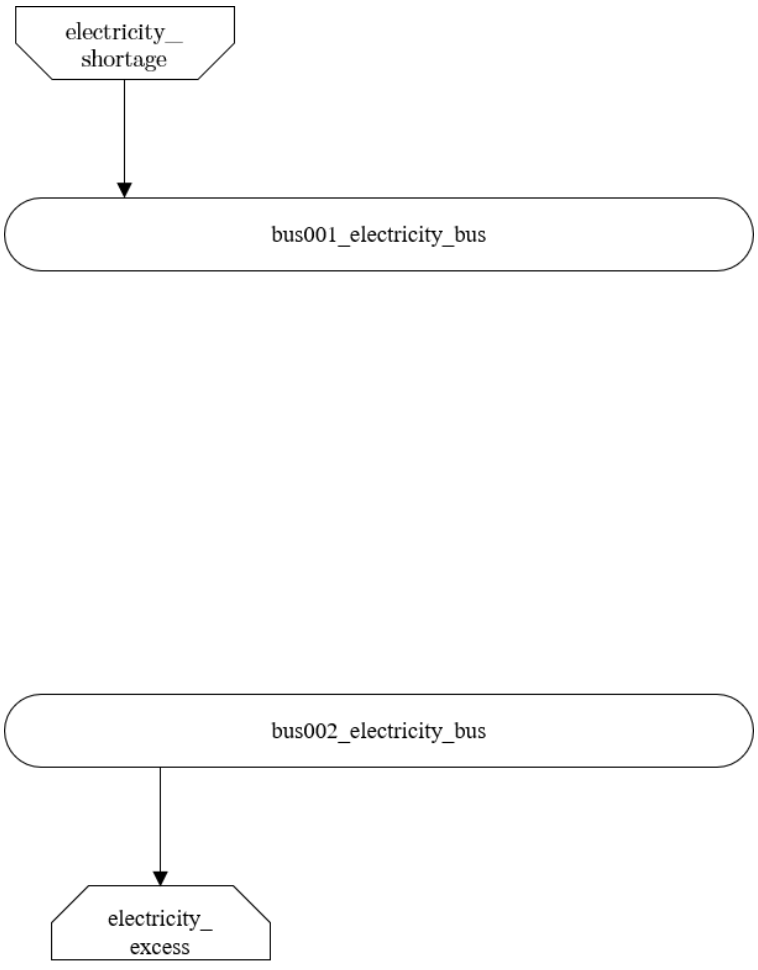

Fig. 4: Graph of the energy system, which is created by entering the example components. Two buses, a shortage source, and an excess sink were created by the input.

2.3.3 Sinks

Within this sheet, the sinks of the energy system are defined. The following parameters need to be entered:

- **label**: Unique designation of the sink. The following format is recommended: "ID_energy sector_sinks".
- **comment**: Space for an individual comment, e.g. an indication of which measure this component belongs to.
- **active**: Specifies whether the sink shall be included to the model. 0 = inactive, 1 = active.
- **input**: Space for an individual comment, e.g. an indication of which measure this component belongs to.
- **load profile**: Specifies the basis onto which the load profile of the sink is to be created. If the Richardson tool is to be used, "richardson" has to be inserted. For standard load profiles, its acronym is used. If a time series is used, "timeseries" must be entered. If the source is not fixed, the fill character "x" has to be used.
- **nominal value/(kW)**: Nominal performance of the sink. Required when "time series" has been entered into the "load profile". When SLP or Richardson is used, use the fill character "x" here.
- **annual demand/(kWh/a)**: Annual energy demand of the sink. Required when using the Richardson Tool or standard load profiles. When using time series, the fill character "x" is used.
- **occupants [RICHARDSON]**: Number of occupants living in the respective building. Only required when using the Richardson tool, use fill character "x" for other load profiles.
- **building class [HEAT SLP ONLY]**: BDEW-building class.
- **wind class [HEAT SLP ONLY]**: wind classification for building location (0=not windy, 1=windy)
- **fixed**: Indicates whether it is a fixed sink or not. 0 = not fixed; 1 = fixed.

label	comment	active	input	load profile
building001_electricity_sink	example sink	1	bus001_electricity_bus	richardson

nominal value [kW]	annual demand [kWh/a]	occupants [RICHARDSON]	building class [HEAT SLP ONLY]	wind class [HEAT SLP ONLY]	fixed
x	5000.0	2	x	x	1

Fig. 5: Exemplary input for the sinks sheet

2.3.4 Sources

Within this sheet, the sources of the energy system are defined. Properties with the addition "PV ONLY" have only to be defined if the parameter "technology" is set on "photovoltaic". The following parameters have to be entered:

- **label**: Unique designation of the source. The following format is recommended: "ID_energy sector_source".
- **comment**: Space for an individual comment, e.g. an indication of which measure this component belongs to.
- **active**: Specifies whether the source shall be included to the model. 0 = inactive, 1 = active.
- **output**: Specifies which bus the source is connected to.
- **technology**: Technology type of source. Input options: "photovoltaic", "windpower", "timeseries". Time series are automatically generated for photovoltaic systems and wind turbines. If "timeseries" is selected, a time series must be provided in the "time_series" sheet.
- **Turbine Model (Windpower ONLY)**: Reference wind turbine model. Possible turbine types are listed here.
- **Hub Height (Windpower ONLY)**: Hub height of the wind turbine. Which hub heights are possible for the selected reference turbine can be viewed here.
- **variable costs/(CU/kWh)**: Defines the variable costs incurred for a kWh of energy drawn from the source.
- **existing capacity/(kW)**: Existing capacity of the source before possible investments.

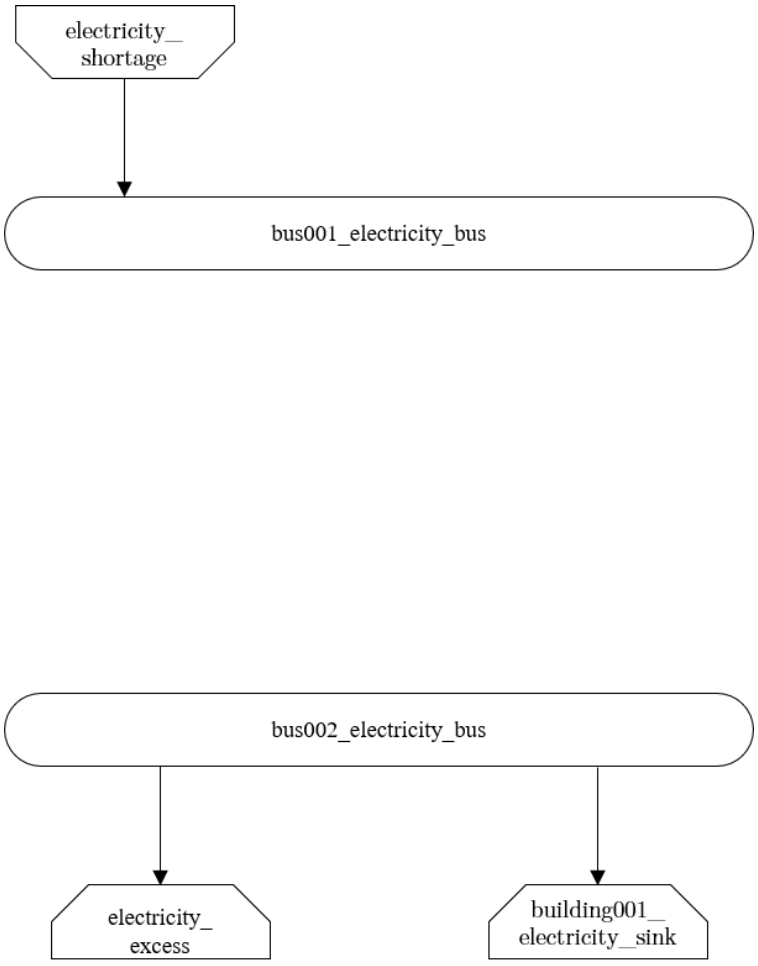

Fig. 6: Graph of the energy system, which is created by entering the example components. By the input in the sinks sheet, a photovoltaic source has been created.

- **min. investment capacity/(kW)**: Minimum capacity to be installed in case of an investment.
- **max. investment capacity/(kW)**: Maximum capacity that can be added in the case of an investment. If no investment is possible, enter the value "0" here.
- **Non-Convex Investment**: Specifies whether the investment capacity should be defined as a mixed-integer variable, i.e. whether the model can decide whether NOTHING OR THE INVESTMENT should be implemented.
- **Fix Investment Costs /(CU/a)**: Fixed costs of non-convex investments (in addition to the periodic costs)
- **periodical costs/(CU/(kW a))**: Costs incurred per kW for investments within the time horizon
- **technology database (PV ONLY)**: Database, from where module parameters are to be obtained. Recommended Database: "SandiaMod".
- **inverter database (PV ONLY)**: Database, from where inverter parameters are to be obtained. Recommended Database: "sandiainverter".
- **Modul Model (PV ONLY)**: Module name, according to the database used.
- **Inverter Model (PV ONLY)**: Inverter name, according to the database used.
- **Azimuth (PV ONLY)**: Specifies the orientation of the PV module in degrees. Values between 0 and 360 are permissible (0 = north, 90 = east, 180 = south, 270 = west). Only required for photovoltaic sources, use fill character "x" for other technologies.
- **Surface Tilt (PV ONLY)**: Specifies the inclination of the module in degrees (0 = flat). Only required for photovoltaic sources, use fill character "x" for other technologies.
- **Albedo (PV ONLY)**: Specifies the albedo value of the reflecting floor surface. Only required for photovoltaic sources, use fill character "x" for other technologies.
- **Altitude (PV ONLY)**: Height (above mean sea level) in meters of the photovoltaic module. Only required for photovoltaic sources, use fill character "x" for other technologies.
- **Latitude (PV ONLY)**: Geographic latitude (decimal number) of the photovoltaic module. Only required for photovoltaic sources, use fill character "x" for other technologies.
- **Longitude (PV ONLY)**: Geographic longitude (decimal number) of the photovoltaic module. Only required for photovoltaic sources, use fill character "x" for other technologies.
- **fixed**: Indicates whether it is a fixed source or not. 0 = not fixed; 1 = fixed.

label	Comment	active	output	technology
pv001_electricity_source	test	1	bus001_electricity_bus	photovoltaic

variable costs [CU/kWh]	existing capacity [kW]	min. investment capacity [kW]	max. investment capacity [kW]	periodical costs [CU/(kW a)]
0	10	0	10	10.00

technology database (PV ONLY)	inverter database (PV ONLY)	Modul Model (PV ONLY)	Inverter Model (PV ONLY)	Azimuth (PV ONLY)
SandiaMod	sandiainverter	Panasonic_VBHN235SA06B__2	ABB__MICRO_0_25_I_OUTD_U	153

Surface Tilt (PV ONLY)	Albedo (PV ONLY)	Altitude (PV ONLY)	Latitude (PV ONLY)	Longitude (PV ONLY)
32	0.20	60	50	10.000

Fig. 7: Exemplary input for the sources sheet

2.3.5 Transformers

Within this sheet, the transformers of the energy system are defined. Properties with the addition "HP ONLY" have only to be defined if the parameter "transformer type" is set on "HeatPump". With other transformers, these fields can be left empty or filled with any placeholder.

The following parameters have to be entered:

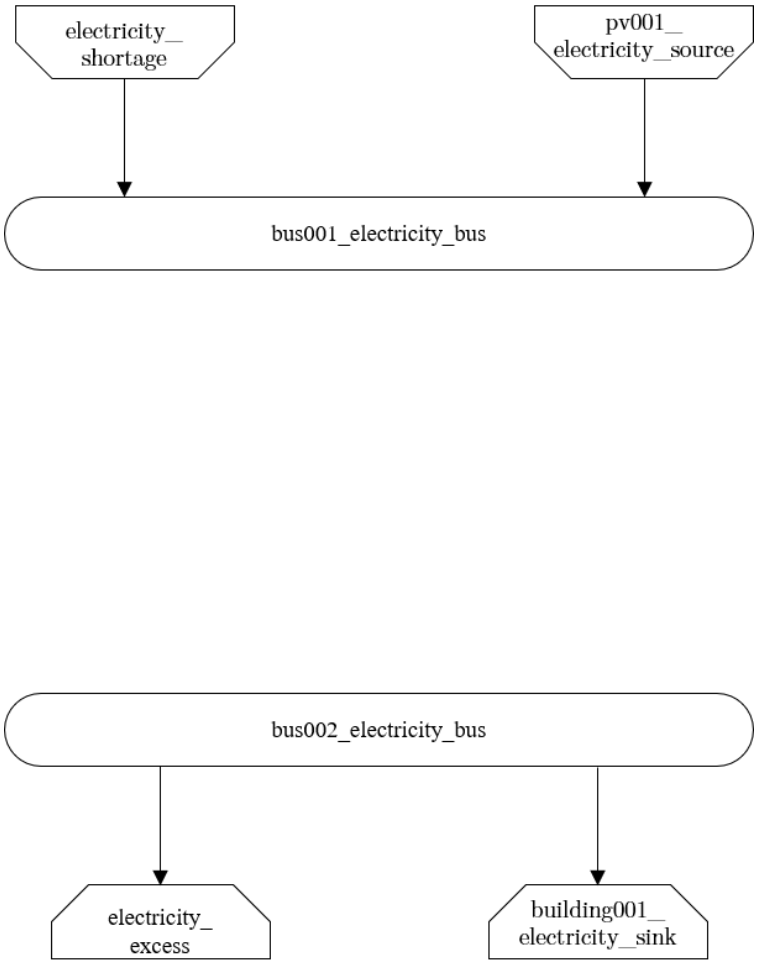

Fig. 8: Graph of the energy system, which is created by entering the example components. By the input in the sources-sheet one sink has been created.

- **label**: Unique designation of the transformer. The following format is recommended: "ID_energy sector_transformer".
- **comment**: Space for an individual comment, e.g. an indication of which measure this component belongs to.
- **active**: Specifies whether the transformer shall be included to the model. 0 = inactive, 1 = active.
- **transformer type**: Indicates what kind of transformer it is. Possible entries: "GenericTransformer" for linear transformers with constant efficiencies; "GenericCHP" for transformers with varying efficiencies.
- **input**: Specifies the bus from which the input to the transformer comes from.
- **output**: Specifies bus to which the output of the transformer is forwarded to.
- **output2**: Specifies the bus to which the output of the transformer is forwarded to, if there are several outputs. If there is no second output, the fill character "x" must be entered here.
- **efficiency**: Specifies the efficiency of the first output. Values between 0 and 1 are allowed entries.
- **efficiency2**: Specifies the efficiency of the second output, if there is one. Values between 0 and 1 are entered. If there is no second output, the fill character "x" must be entered here.
- **variable input costs**: Variable costs incurred per kWh of input energy supplied.
- **existing capacity/(kW)**: Already installed capacity of the transformer.
- **max investment capacity/(kW)**: Maximum installable transformer capacity in addition to the previously existing one.
- **min investment capacity/(kW)**: Minimum transformer capacity to be installed.
- **periodical costs /(CU/a)**: Costs incurred per kW for investments within the time horizon.
- **Non-Convex Investment**: Specifies whether the investment capacity should be defined as a mixed-integer variable, i.e. whether the model can decide whether NOTHING OR THE INVESTMENT should be implemented.
- **Fix Investment Costs /(CU/a)**: Fixed costs of non-convex investments (in addition to the periodic costs)
- **heat source (HP ONLY)**: Specifies the heat source. At the moment are "GroundWater", "Ground", "Air" and "Water" possible.
- **temperature high /(deg C) (HP ONLY)**: Temperature of the high temperature heat reservoir
- **quality grade (HP ONLY)**: To determine the COP of a real machine a scale-down factor (the quality grade) is applied on the Carnot efficiency (see oemof.thermal).
- **area /(sq m) (HP ONLY)**: Open spaces for ground-coupled heat pumps (GCHP).
- **length of the geoth. probe (m) (HP ONLY)**: Length of the vertical heat exchanger, only for GCHP.
- **heat extraction (kW/(m*a)) (HP ONLY)**: Heat extraction for the heat exchanger referring to the location, only for GCHP.
- **min. borehole area (sq m) (HP ONLY)**: Limited space due to the regeneation of the ground source, only for GCHP.
- **temp threshold icing (HP ONLY)**: Temperature below which icing occurs (see oemof.thermal).
- **factor icing (HP ONLY)**: COP reduction caused by icing (see oemof.thermal).

2.3.6 Storages

Within this sheet, the sinks of the energy system are defined. The following parameters have to be entered:

- **label**: Unique designation of the storage. The following format is recommended: "ID_energy sector_storage".

label	comment	active	transformer type	input
tr001_electricity_transformer	example	1	GenericTransformer	bus002_electricity_bus

output	output2	efficiency	efficiency2	variable input costs [CU/kWh]
bus001_electricity_bus	x	0.8	x	0.01

variable output costs [CU/kWh]	existing capacity [kW]	max. investment capacity [kW]	min. investment capacity [kW]	periodical costs [CU/(kW a)]
0	10	10	0	10

Fig. 9: Exemplary input for the transformers sheet

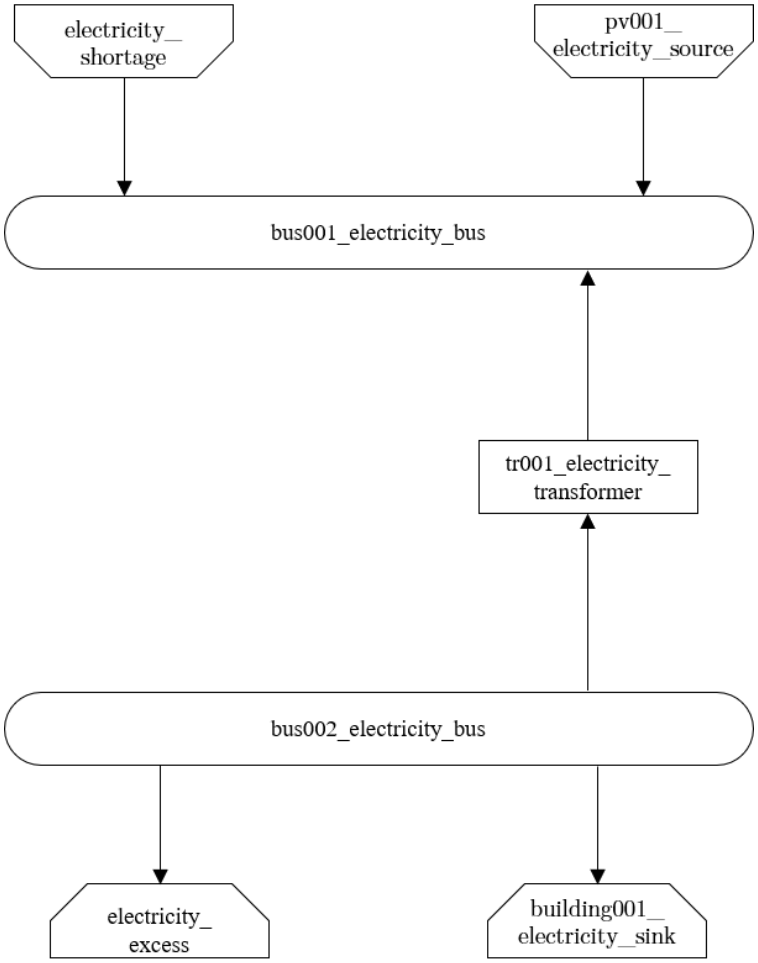

Fig. 10: Graph of the energy system, which is created by entering the example components. One transformer has been created by including the transformers sheet

- **comment**: Space for an individual comment, e.g. an indication of which measure this component belongs to.
- **active**: Specifies whether the storage shall be included to the model. 0 = inactive, 1 = active.
- **bus**: Specifies which bus the storage is connected to.
- **input/capacity ratio (invest)**: Indicates the performance with which the memory can be charged.
- **output/capacity ratio (invest)**: Indicates the performance with which the memory can be discharged.
- **capacity loss**: Indicates the storage loss per time unit.
- **efficiency inflow**: Specifies the charging efficiency.
- **efficiency outflow**: Specifies the discharging efficiency.
- **initial capacity**: Specifies how far the memory is loaded at time 0 of the simulation. Value must be between 0 and 1.
- **capacity min**: Specifies the minimum amount of memory that must be loaded at any given time. Value must be between 0 and 1.
- **capacity max**: Specifies the maximum amount of memory that can be loaded at any given time. Value must be between 0 and 1.
- **variable input costs**: Indicates how many costs arise for charging with one kWh.
- **variable output costs**: Indicates how many costs arise for charging with one kWh.
- **existing capacity/(kW)**: Previously installed capacity of the storage.
- **periodical costs /(CU/a)**: Costs incurred per kW for investments within the time horizon.
- **max. investment capacity/(kW)**: Maximum in addition to existing capacity, installable storage capacity.
- **min. investment capacity/(kW)**: Minimum storage capacity to be installed.
- **Non-Convex Investment**: Specifies whether the investment capacity should be defined as a mixed-integer variable, i.e. whether the model can decide whether NOTHING OR THE INVESTMENT should be implemented.
- **Fix Investment Costs /(CU/a)**: Fixed costs of non-convex investments (in addition to the periodic costs)

label	comment	active	bus	existing capacity /(kWh)	min. investment capacity /(kWh)	max. investment capacity /(kWh)
battery001_electricity_storage	test	1	bus001_electricity_bus	0	0	19.5

periodical costs /(CU/(kWh a))	input/capacity ratio (invest)	output/capacity ratio (invest)	capacity loss	efficiency inflow	efficiency outflow	initial capacity
70.97	0.17	0.17	0	1	0.98	0

capacity min	capacity max	variable input costs	variable output costs	Non-Convex Investment	Fix Investment Costs /(CU/a)
0.1	1	0	0	0	0

Fig. 11: Exemplary input for the storages sheet

2.3.7 Links

Within this sheet, the links of the energy system are defined. The following parameters have to be entered:

- **label**: Unique designation of the link. The following format is recommended: “ID_energy sector_transformer”
- **comment**: Space for an individual comment, e.g. an indication of which measure this component belongs to.
- **active**: Specifies whether the link shall be included to the model. 0 = inactive, 1 = active.
- **bus_1**: First bus to which the link is connected. If it is a directed link, this is the input bus.
- **bus_2**: Second bus to which the link is connected. If it is a directed link, this is the output bus.

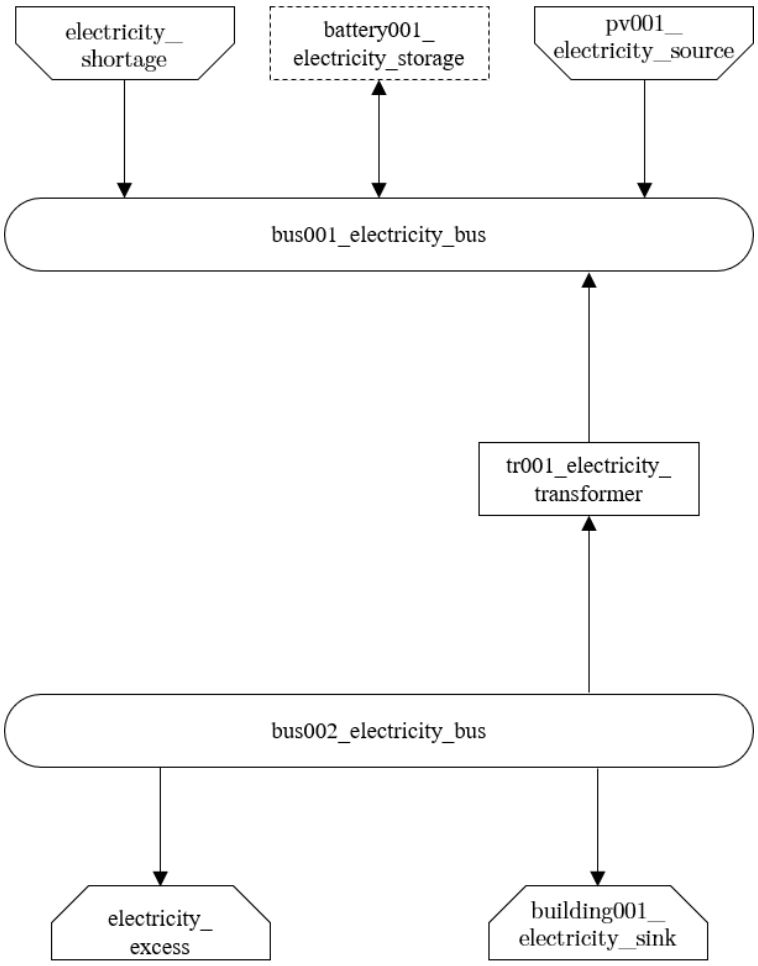

Fig. 12: Graph of the energy system, which is created after entering the example components. One storage has been created by the storage sheet.

- **(un)directed**: Specifies whether it is a directed or an undirected link. Input options: “directed”, “undirected”.
- **efficiency**: Specifies the efficiency of the link. Values between 0 and 1 are allowed entries.
- **existing capacity/(kW)**: Already installed capacity of the link.
- **min. investment capacity/(kW)**: Minimum, in addition to existing capacity, installable capacity.
- **max. investment capacity/(kW)**: Maximum capacity to be installed.
- **variable costs/(CU/kWh)**: Specifies the efficiency of the first output. Values between 0 and 1 are allowed entries.
- **periodical costs/(CU/(kW a))**: Costs incurred per kW for investments within the time horizon.
- **Non-Convex Investment**: Specifies whether the investment capacity should be defined as a mixed-integer variable, i.e. whether the model can decide whether NOTHING OR THE INVESTMENT should be implemented.
- **Fix Investment Costs /(CU/a)**: Fixed costs of non-convex investments (in addition to the periodic costs)

label	Comment	active	bus_1	bus_2	(un)directed
pl001_electricity_link	example	1	bus001_electricity_bus	bus002_electricity_bus	directed

efficiency	existing capacity [kW]	min. investment capacity [kW]	max. investment capacity [kW]	variable costs [CU/kWh]	periodical costs [CU/(kW a)]
1	100	0	0	0	0

Fig. 13: Exemplary input for the input in the storages sheet

2.3.8 Time Series

Within this sheet, time series of components of which no automatically created time series exist, are stored. More specifically, these are sinks to which the property “load profile” have been assigned as “timeseries” and sources with the “technology” property “timeseries”. The following parameters have to be entered:

- **timestamp**: Points in time to which the stored time series are related. Should be within the time horizon defined in the sheet “timesystem”.
- **timeseries**: Time series of a sink or a source which has been assigned the property “timeseries” under the attribute “load profile” or “technology. Time series contain a value between 0 and 1 for each point in time, which indicates the proportion of installed capacity accounted for by the capacity produced at that point in time. In the header line, the name must rather be entered in the format “componentID.fix” if the component enters the power system as a fixed component or it requires two columns in the format “componentID.min” and “componentID.max” if it is an unfixed component. The columns “componentID.min/.max” define the range that the solver can use for its optimisation.

2.3.9 Weather Data

If electrical load profiles are simulated with the Richardson tool, heating load profiles with the demandlib or photovoltaic systems with the feedinlib, weather data must be stored here. The weather data time system should be in conformity with the model’s time system, defined in the sheet “timesystem”.

- **timestamp**: Points in time to which the stored weather data are related.
- **dhi**: diffuse horizontal irradiance in W/m^2
- **dirhi**: direct horizontal irradiance in W/m^2
- **pressure**: air pressure in Pa
- **windspeed**: Wind speed, measured at 10 m height, in unit m/s

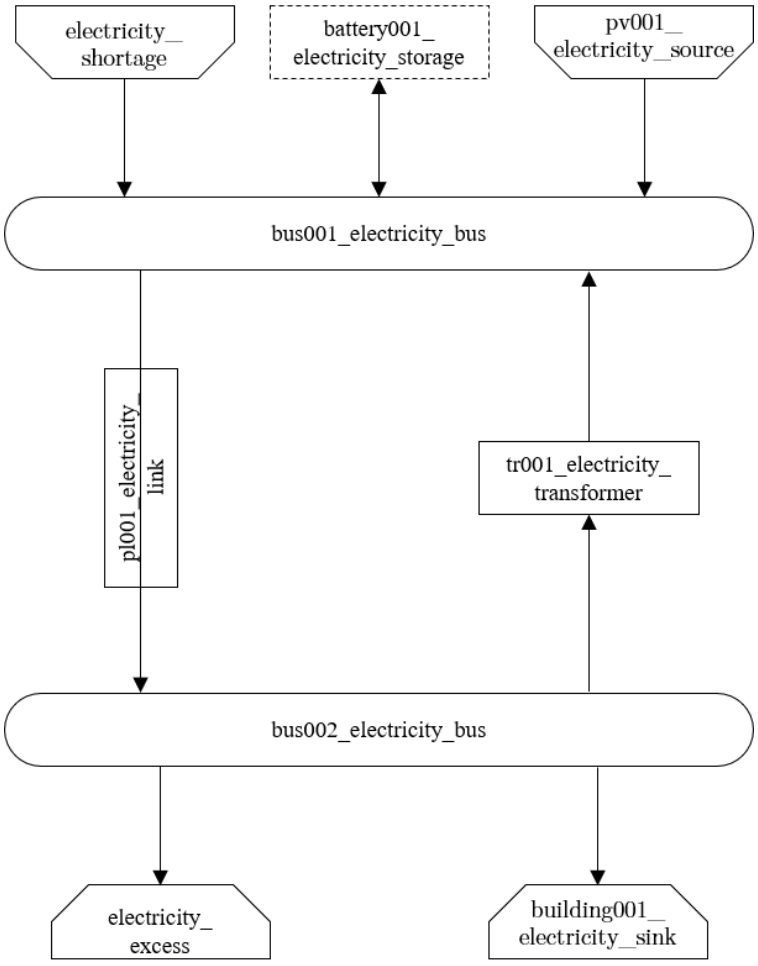

Fig. 14: Graph of the energy system, which is created by entering the example components. One link has been created by the addition of the links sheet

timestamp	unfixed_timeseries_source.min	unfixed_timeseries_source.max	fixed_timeseries_sink.fix
2012-01-01 00:00:00	0,000000	1,000000	0,000000
2012-01-01 01:00:00	0,000000	0,500000	0,041667
2012-01-01 02:00:00	0,000000	0,333333	0,083333
2012-01-01 03:00:00	0,000000	0,250000	0,125000
2012-01-01 04:00:00	0,000000	0,200000	0,166667
2012-01-01 05:00:00	0,000000	0,166667	0,208333
2012-01-01 06:00:00	0,000000	0,142857	0,250000
2012-01-01 07:00:00	0,000000	0,125000	0,291667
2012-01-01 08:00:00	0,000000	0,111111	0,333333
2012-01-01 09:00:00	0,000000	0,100000	0,375000
2012-01-01 10:00:00	0,000000	0,090909	0,416667
2012-01-01 11:00:00	0,000000	0,083333	0,458333
2012-01-01 12:00:00	0,000000	0,076923	0,500000
2012-01-01 13:00:00	0,000000	0,071429	0,541667
2012-01-01 14:00:00	0,000000	0,066667	0,583333
2012-01-01 15:00:00	0,000000	0,062500	0,625000
2012-01-01 16:00:00	0,000000	0,058824	0,666667
2012-01-01 17:00:00	0,000000	0,055556	0,708333
2012-01-01 18:00:00	0,000000	0,052632	0,750000
2012-01-01 19:00:00	0,000000	0,050000	0,791667
2012-01-01 20:00:00	0,000000	0,047619	0,833333
2012-01-01 21:00:00	0,000000	0,045455	0,875000
2012-01-01 22:00:00	0,000000	0,043478	0,916667
2012-01-01 23:00:00	0,000000	0,041667	0,958333

Fig. 15: Exemplary input for time series sheet

- **z0**: roughness length of the environment in units m

	dhi	dirhi	pressure	temperature	windspeed	z0
2012-01-01 00:00:00	0.00	0.00	98405.70	10.33	2.00	0.15
2012-01-01 01:00:00	0.00	0.00	98405.70	10.33	2.00	0.15
2012-01-01 02:00:00	0.00	0.00	98405.70	10.48	2.00	0.15
2012-01-01 03:00:00	0.00	0.00	98405.70	10.55	2.00	0.15
2012-01-01 04:00:00	0.00	0.00	98405.70	10.93	2.00	0.15
2012-01-01 05:00:00	0.00	0.00	98405.70	10.90	2.00	0.15
2012-01-01 06:00:00	0.00	0.00	98405.70	10.88	2.00	0.15
2012-01-01 07:00:00	0.00	0.00	98405.70	11.22	2.00	0.15
2012-01-01 08:00:00	0.00	0.00	98405.70	11.68	2.00	0.15
2012-01-01 09:00:00	0.56	0.56	98405.70	11.87	2.00	0.15
2012-01-01 10:00:00	13.06	13.06	98405.70	11.65	2.00	0.15
2012-01-01 11:00:00	27.50	27.50	98405.70	11.82	2.00	0.15
2012-01-01 12:00:00	38.89	38.89	98405.70	12.05	2.00	0.15
2012-01-01 13:00:00	35.83	35.83	98405.70	12.35	2.00	0.15
2012-01-01 14:00:00	23.61	23.61	98405.70	12.37	2.00	0.15
2012-01-01 15:00:00	6.94	6.94	98405.70	12.02	2.00	0.15
2012-01-01 16:00:00	0.00	0.00	98405.70	12.30	2.00	0.15
2012-01-01 17:00:00	0.00	0.00	98405.70	12.53	2.00	0.15
2012-01-01 18:00:00	0.00	0.00	98405.70	12.87	2.00	0.15
2012-01-01 19:00:00	0.00	0.00	98405.70	13.10	2.00	0.15
2012-01-01 20:00:00	0.00	0.00	98405.70	13.58	2.00	0.15
2012-01-01 21:00:00	0.00	0.00	98405.70	13.42	2.00	0.15
2012-01-01 22:00:00	0.00	0.00	98405.70	14.00	2.00	0.15
2012-01-01 23:00:00	0.00	0.00	98405.70	14.23	2.00	0.15

Fig. 16: Exemplary input for weather data

2.4 Analyzing the Results

2.4.1 Interactive Results

If the Spreadsheet Energy System Model Generator was executed via the exe.cmd-file, a browser window with interactive results will be opened automatically after successful modeling. Alternatively, the results of the last modelling run can be accessed by executing the Interactive_Results.py file.

The results interface has the following elements:

- **Table with a summary of the modelling (1).**
- **Graph of the energy system (2).**
- **Table with information about every component (3):** The entries can be filtered and sorted according to their content.
- **Plot, where time series of the different components can be shown (5).** With the help of a selection window (drop-down menu and search function) (4) time series to be shown can be selected. With the help of a number of tools (6) the graphs can be scaled, sections can be displayed and images can be saved.

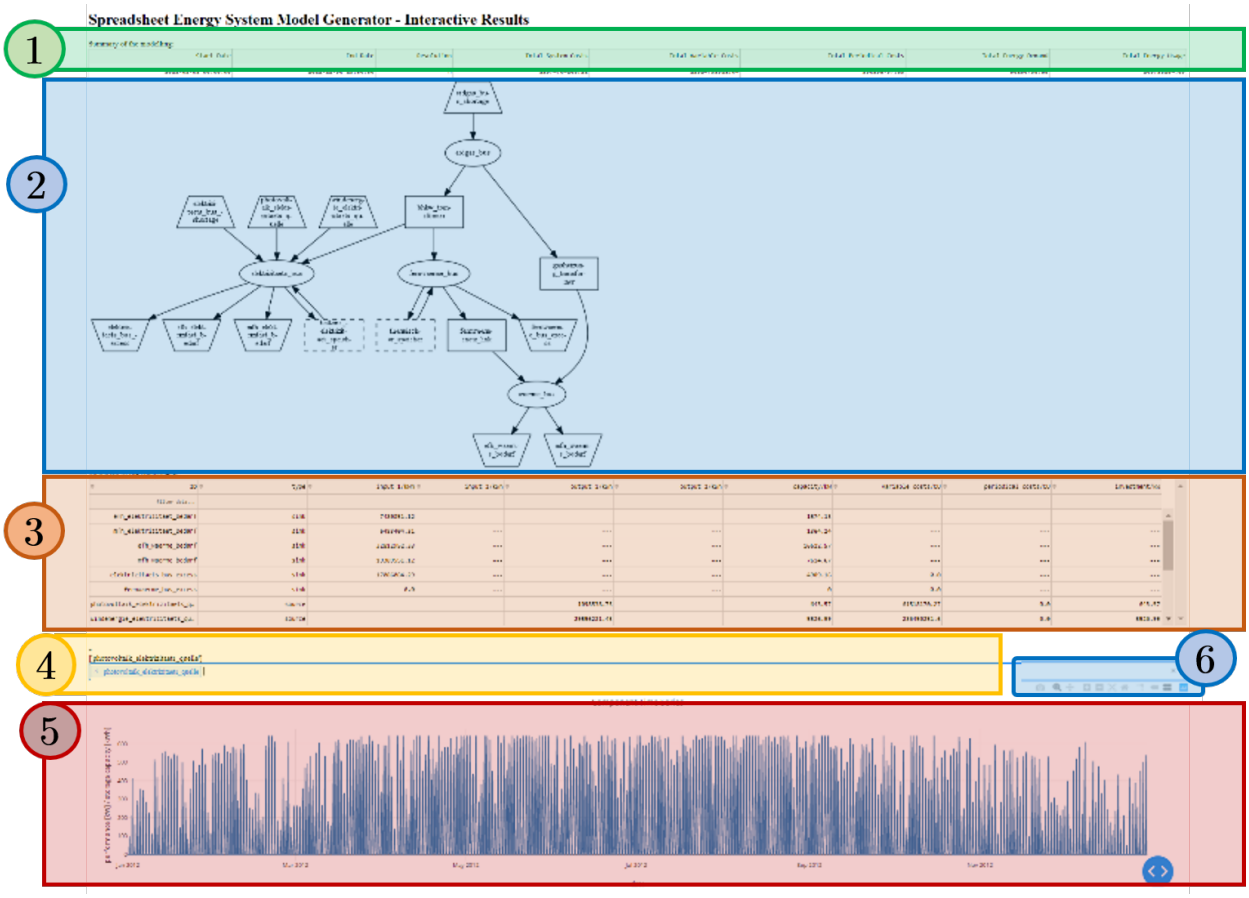

Fig. 17: Screenshots of the interactive results browser interface

2.4.2 Results as Spreadsheets and Log-Files

The results of the modeling are stored in the “results” folder in two formats: - as summarizing log files, under - as detailed xlsx-files.

The log-file gives an overview of which components are created and which of the investment options should be implemented. In addition, it is indicated which costs for the supply of the energy system are incurred in the optimized case. For each implemented bus, an xlsx-file is created in which incoming and outgoing energy flows are specified for each time step of the model are.

2.5 Demo Tool

With the help of the demo tool a descriptive introduction to energy system modeling can be given. The demo tool has been implemented for training purposes and is not intended for the actual analysis of a real energy system.

Users have the possibility to integrate different energy supply technologies into a defined municipal energy system. If possible, the technologies should be dimensioned in such a way that the monetary costs of the entire system and/or the carbon dioxide emissions are reduced.

With the help of the demo tool the chosen energy supply scenario can be simulated by simply entering performance values (monetary costs and CO2 emissions) and then compared graphically with the stored optimized scenarios.

2.5.1 Instruction

1: Tab to select the demo tool

2: Input of the supply scenario to be simulated

3: Start of the simulation

4: Output of results

5: Storage of the intermediate results shown in (4) for display in the results output (8 and 10)

6: Input of already known simulation results (e.g. from previous simulation runs). By using this function, computing time can be saved.

7: Storage of the values from (6) for display in the result output

8: Display of a scatter plot for all saved scenarios

9: Adoption of optimized and status-quo scenarios in the output of results

10: Display of a bar plot for all saved scenarios

11: Overview of the stored system parameters

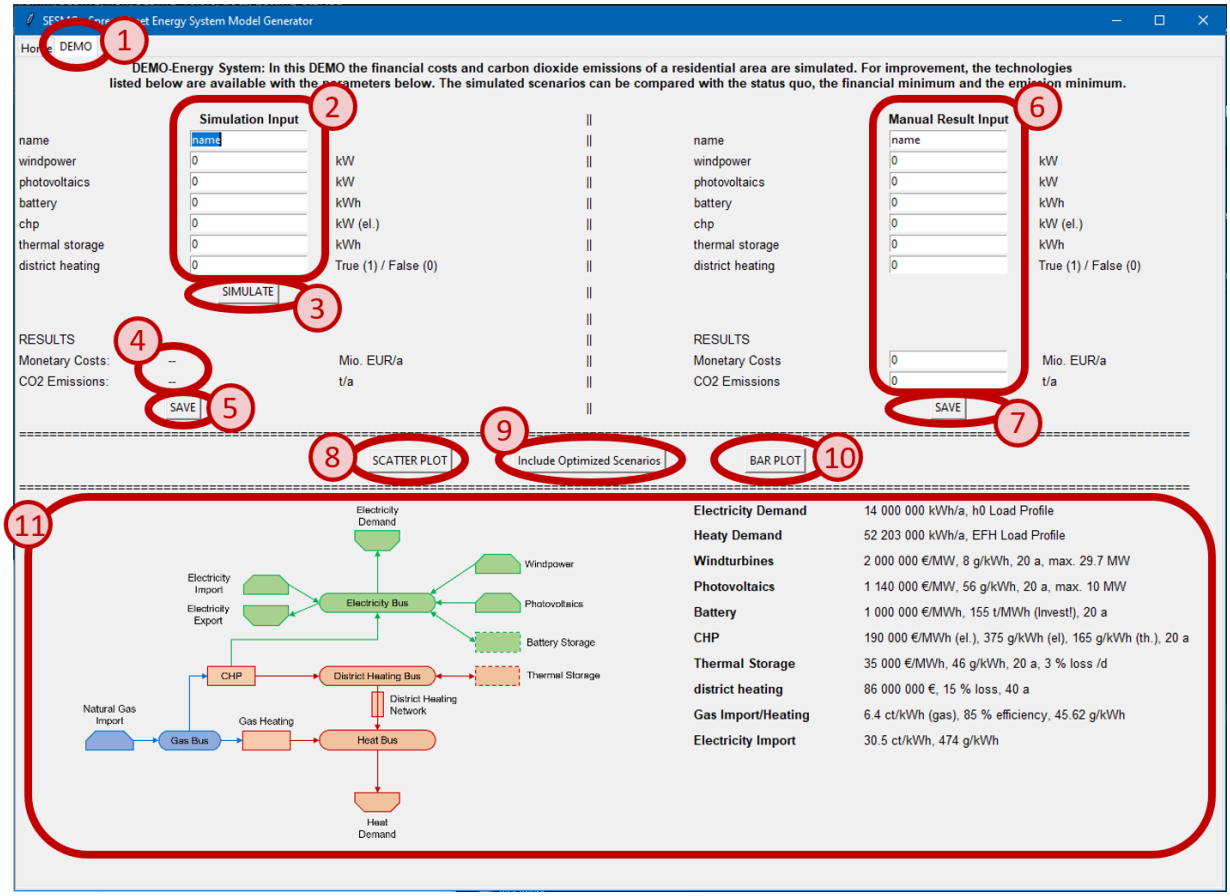

Fig. 18: Interface of the Demo Tool

CHAPTER 3

Troubleshooting

3.1 Troubleshooting

During execution of the model generator, error messages and associated program aborts may occur.

General debugging:

Pay attention to the correct spelling:

- Pay attention to correct upper and lower case.
- Do not use spaces in the entire spreadsheet (except for the “comment” columns).
- Make sure that every column of the used lines is filled. Columns that are not used can be filled with an “x”.

Make sure that the displayed system can stay in balance. - It must always be possible to take off all of the supplied energy and vice versa. - The use of excess-sinks and shortage-sources can help to keep the system in balance.

Known error messages:

> **Warning:** flowsum = source[‘sequences’].sum()
>
> KeyError: ‘sequences’

or

> **Warning:** ApplicationError: Solver (cbc) did not exit normally

- **Possible Error Cause**: A system component was entered incorrectly in the input file.

- **Debugging**: For all components, make sure that 1) each column is filled correctly, and 2) the first component of a sheet is entered in the row directly below the header row, and that there are no blank rows between the individual components of a sheet

Warning: df = node_results['sequences']

KeyError: 'sequences'

- **Possible Error Cause**: The implemented model probably has an circuit. For example, the excess sink of a bus could achieve higher selling prices than buying from a shortage source. In theory, this could generate an infinitely large profit. Such a model cannot be solved.
- **Debugging**: Make sure, there are no circuits within the model.

Warning: Memory Error

- **Possible Error Cause**: The available memory is not sufficient to solve the model.
- **Debugging**: Take the following measures gradually until the error no longer occurs:
  - Restart the used Python interpreter
  - Close unnecessary programs on the computer
  - Make sure that python 64 bit version is used (Python 32 bit can manage only 2 GB of memory).
  - Start the program on a more powerful computer.

Warning: AttributeError: module 'time' has no attribute 'clock'

- **Possible Error Cause**: You are using a Python version not compatible with oemof.
- **Debugging**: Use Pyhton 3.7

Warning: ValueError: operands could not be broadcast together with shapes (8784,) (8760,)

- **Possible Error Cause**: The weather dataset contains the wrong number of data points for using feedinlib.
- **Debugging**: Make sure that the number of weather data points corresponds to the time steps of the model (At hourly resolution, one year has 8760 time steps). When simulating a leap year, it is recommended limiting the time horizon to 8760 hours.

Warning: ValueError: pyutilib.common._exceptions.ApplicationError: Solver (cbc) did not exit normally

- **Possible Error Cause**: A value for the use of the investment module (e.g., "min Investment Capacity") was not filled in.
- **Debugging**: Make sure, that all necessary cells of the spreadsheet have been filled in.

Warning: KeyError: '*__any component name__*'

- **Possible Error Cause**: Incorrectly assigned bus name for the input or output of a component
- **Debugging**: Check that all bus references are correct. Also check for typos.

Warning: TypeError: ufunc 'true_divide' not supported for the input types, and the inputs could not be safely coerced to any supported types according to the casting rule ''safe"

- **Possible Error Cause**: The column "annual demand" was not filled in correctly for a sink.
- **Debugging**: Make sure to use the "annual demand" column for SLP and Richardson sinks and the "nominal value" column for time series sinks.

Warning: AttributeError: 'str' object has no attribute 'is_variable_type'

- **Possible Error Cause**: The cost value for an activated excess sink or shortage source was not correctly specified in the bus sheet
- **Debugging**: Make sure that all excess/sortage prices consist of real numbers. Also check for typos.

Warning: "exe.cmd" is not executed (no errror messages)

- **Possible Error Cause**: The used Python environment does not work correctly.
- **Debugging**: Reinstall python (see "Getting Started", step 1 "Install Python").

Your error message is not included? Do not hesitate to contact the developers.

CHAPTER 4

Sourcecode

- *Python code documentation*

4.1 Sourcecode documentation

The Spreadsheet Energy System Model Generator has a hierarchical structure and consists of a total of four work blocks, which in turn consist of various functions and subfunctions. The individual (sub-)functions are documented with docstrings according to the PEP 257 standard. Thus, the descriptions of functions, any information about input and output variables and further details can be easily accessed via the python help function. The model generator's flow chart is shown in the following figure, including all input and output data, used functions and Python libraries.

Create Energy System. In the first block, the Python library Pandas is used to read the input xlsx-spreadsheet file. Subsequently, an oemof time index (time steps for a time horizon with a resolution defined in the input file) is created on the basis of the parameters imported. This block is the basis for creating the model. The model does not yet contain any system components, these must be added in the following blocks.

Create Objects. In the second block, the system components defined in the xlsxscenario file are created according to the oemof specifications, and added to the model. At first, the buses are initialized, followed by the sources, sinks, transformers, storages and links. With the creation of sources, commodity sources are created first and photovoltaic sources second. The creation of sinks is divided into six sub-functions, one for each type of sinks: unfixed sinks, sinks with a given time series, sinks using standard load profiles (residential heat, commercial heat, electricity) as well as sinks using load profiles that were created with the Richardson tool. Although it is untypical to convert a function into a single sub-function, this alternative was chosen for the creation of transformers and storages. This offers the option to add further sub-functions such as additional types of transformers and storages lateron. Lastly, the creation of links is divided into the creation of undirected and directed links.

Optimize Model. Within the third block, the CBC solver is utilized to solve the energy system for minimum costs. It returns the "best" scenario. This block only contains one function. Again, further functions may be added lateron, for example the combination of more than one assessment criterion.

Create Results. In the last block, the scenario as returned from the CBC solver is analyzed and prepared for further processing. With the first function of this block, the results are saved within xlsx-files. It contains ingoing and outgoing energy flows for every time step of the entire time horizon. With the second function, a set of statistics for every component is returned into a log-file. Finally, the results are illustrated as shown in the chapters above.

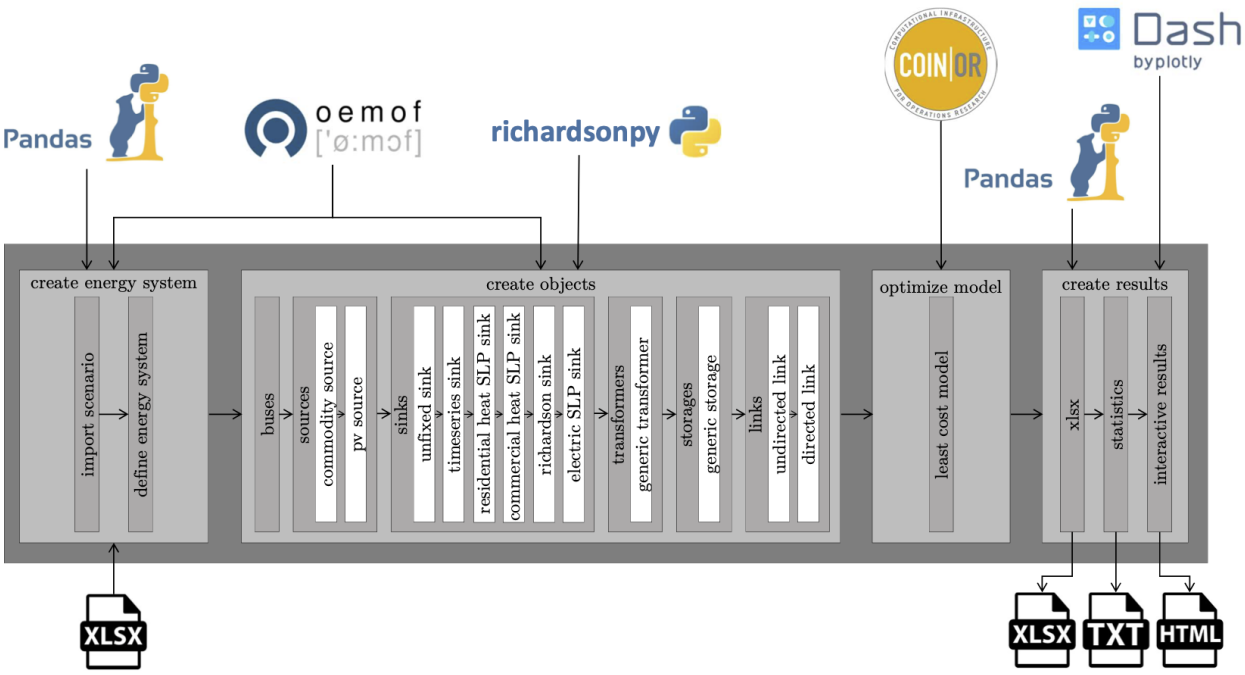

Fig. 1: Program flow of the Spreadsheet Energy System Model Generator (grey, center), as well as local inputs and outputs (bottom) and used Python libraries (top).

4.1.1 create energy system

def import_scenario(filepath)

Imports Data from a Spreadsheet Scenario File. | | | | The excel sheet has to contain the following sheets:| | |

- timesystem |
- buses |
- transformers |
- sinks |
- sources |
- storages |
- powerlines |
- time_series | | |

Parameters |

|

- String filename : path to excel scenario file|

Return values

- dict nodes_data : dictionary containing data from excel scenario file

def define_energy_system(nodes_data)

Creates an energy system.

Creates an energy system with the parameters defined in the given .xlsx-file. The file has to contain a sheet called “timesystem”, which must have the following structure:

start_date	end_date	temporal resolution
YYYY-MM-DD hh:mm:ss	YYYY-MM-DD hh:mm:ss	h

Parameters

- dict nodes_data : dictionary containing data from excel scenario file

Return values

- dict esys : oemof energy system

4.1.2 create objects

def buses(nodes_data, nodes)

Creates bus objects.

Creates bus objects with the parameters given in 'nodes_data' and adds them to the list of components 'nodes'.

Parameters

- **dict nodes_data** [dictionary containing parameters]

  the buses to be created. The following parameters have to be provided:

  - label,
  - active,
  - excess,
  - shortage,
  - shortage costs /(CU/kWh),
  - excess costs /(CU/kWh)

- list nodes : list of components created before (can be empty)

Return values

- dict busd : dictionary containing all buses created

class Sources

Creates source objects.

There are four options for labeling source objects to be created:

- 'commodity' : a source with flexible time series
- 'timeseries' : a source with predefined time series
- 'photovoltaic' : a photovoltaic component
- 'wind power' : a wind power component

Creates an oemof source with fixed / unfixed timeseries

Paramters

- dict so: dictionary containing all information for the

creation of an oemof source. At least the following key-value-pairs have to be included: - 'label' - 'output' - 'periodical costs /(CU/(kW a))' - 'min. investment capacity /(kW)' - 'max. investment capacity /(kW)' - 'existing capacity /(kW)' - 'Non-Convex Investment' - 'Fix Investment Costs /(CU/a)' - 'variable costs /(CU/kWh)'

- **dict timeseries_args: dictionary rather containing the 'fix-attribute' or the** 'min-' and 'max-attribute' of a source

Creates an oemof source with flexible time series (no maximum or minimum) with the use of the create_source method.

Parameters

- dict so: dictionary containing all information for the

creation of an oemof source. At least the following key-value-pairs have to be included: - 'label'

Creates an oemof source object from a pre-defined timeseries with the use of the create_source method.

Parameters

- dict so: dictionary containing all information for the

creation of an oemof source. At least the following key-value-pairs have to be included: - 'label' - 'fixed'

- String filepath: path to .xlsx scenario-file containing a "time_series" sheet

Creates an oemof photovoltaic source object.

Simulates the yield of a photovoltaic system using feedinlib and creates a source object with the yield as time series and the use of the create_source method.

Parameters

- dict so: dictionary containing all information for the

creation of an oemof source. At least the following key-value-pairs have to be included:

- 'label'
- 'fixed'
- 'Azimuth (PV ONLY)'
- 'Surface Tilt (PV ONLY)'
- 'Modul Model (PV ONLY)'
- 'Inverter Model (PV ONLY)'
- 'Albedo (PV ONLY)'
- 'Latitude (PV ONLY)'
- 'Longitude (PV ONLY)'

Creates an oemof windpower source object.

Simulates the yield of a windturbine using feedinlib and creates a source object with the yield as time series and the use of the create_source method.

Parameters

- dict so: dictionary containing all information for the

creation of an oemof source. At least the following key-value-pairs have to be included:

- 'label'
- 'fixed'
- 'Turbine Model (Windpower ONLY)'
- 'Hub Height (Windpower ONLY)'

Inits the source class.

Parameters

- **dict nodes_data** [dictionary containing parameters of sources to be created.]

  The following data have to be provided:

  - 'label'
  - **'active'**
    - 'fixed'
  - 'output'
  - 'technology'
  - 'variable costs / (CU / kWh)'
  - 'existing capacity / (kW)'
  - 'min.investment capacity / (kW)'
  - 'max.investment capacity / (kW)'
  - 'periodical costs / (CU / (kW a))'
  - 'Non-Convex Investment'
  - 'Fix Investment Cost / (CU/a)'
  - 'Turbine Model (Windpower ONLY)'
  - 'Hub Height (Windpower ONLY)'
  - 'technology database(PV ONLY)'
  - 'inverter database(PV ONLY)'
  - 'Modul Model(PV ONLY)'
  - 'Inverter Model(PV ONLY)'
  - 'Azimuth(PV ONLY)'
  - 'Surface Tilt(PV ONLY)'
  - 'Albedo(PV ONLY)'
  - 'Altitude(PV ONLY)'
  - 'Latitude(PV ONLY)'
  - 'Longitude(PV ONLY)'

  - dict bus : dictionary containing the buses of the energy system
  - list nodes : list of components created before (can be empty)
  - **String filepath** [path to .xlsx scenario-file containing a "weather data" sheet] with timeseries for - "dhi" (diffuse horizontal irradiation) W/m^2 - "dirhi" (direct horizontal irradiance) W/m^2 - "pressure" in Pa - "temperature" in °C - "windspeed" in m/s - "z0" (roughness length) in m

Other Variables

- list nodes_sources : class intern list of sources that are already created

class Sinks

Creates sink objects.

There are four options for labeling source objects to be created:

- ‘unfixed’ : a source with flexible time series
- ‘timeseries’ : a source with predefined time series
- SLP : a VDEW standard load profile component
- ‘richardson’ : a component with stochatical generated timeseries

Creates an oemof sink with fixed or unfixed timeseries.

Parameters

- **dict de: dictionary containing all information for the** creation of an oemof sink. At least the following key-value-pairs have to be included: - ‘label’
  - ‘input’
- dict timeseries_args: dictionary rather containing the ‘fix-attribute’

or the ‘min-‘ and ‘max-attribute’ of a sink

Creates a sink object with an unfixed energy input and the use of the create_sink method.

Parameters:

- **dict de: dictionary containing all information for the** creation of an oemof sink. At least the following key-value-pairs have to be included: - ‘label’ - ‘nominal value /(kW)’

Creates a sink object with fixed input. The input must be given as a time series in the scenario file. In this context the method uses the create_sink method.

Parameters:

- **dict de: dictionary containing all information for the** creation of an oemof sink. At least the following key-value-pairs have to be included: - ‘label’ - ‘nominal value /(kW)’
- String filepath: path to .xlsx scenario-file containing a “time_series” sheet

  Creates a sink with a residential or commercial SLP time series.

  Creates a sink with inputs according to VDEW standard load profiles, using oemofs demandlib. Used for the modelling of residential or commercial

electricity demand. In this context the method uses the create_sink method.

Parameters:

- **dict de: dictionary containing all information for the** creation of an oemof sink. At least the following key-value-pairs have to be included:
  - ‘label’
  - ‘load profile’
  - ‘annual demand /(kWh/a)’
  - ‘building class [HEAT SLP ONLY]’
  - ‘wind class [HEAT SLP ONLY]’

- dict nd: dictionary containing the whole scenario file

Creates a sink with stochastical timeseries.

Creates a sink with stochastical input, using richardson.py. Used for the modelling of residential electricity demands. In this context the method uses the create_sink method.

Parameters:

- **dict de: dictionary containing all information for the** creation of an oemof sink. At least the following key-value-pairs have to be included:
  - 'label'
  - 'fixed'
  - 'annual demand /(kWh/a)'
  - 'occupants [RICHARDSON]'
- String filepath: path to the .xlsx scenario-file containing a "timesysten" sheet

Creates a sink with stochastical timeseries.

Creates a sink with stochastical input, using richardson.py. Used for the modelling of residential electricity demands. In this context the method uses the create_sink method.

Parameters:

- **dict de: dictionary containing all information for the** creation of an oemof sink. At least the following key-value-pairs have to be included:
  - 'label'
  - 'active'
  - 'fixed'
  - 'input'
  - 'load profile'
  - 'nominal value /(kW)'
  - 'annual demand /(kWh/a)'
  - 'occupants [Richardson]'
  - 'building class [HEAT SLP ONLY]'
  - 'wind class [HEAT SLP ONLY]'
- dict busd: dicitionary containing the buses of the energy system
- list nodes: list of components created before (can be empty)
- String filepath: path to .xlsx scenario-file containing a

"weather data" sheet with timeseries for

- "dhi"(diffuse horizontal irradiation) W / m ^ 2
- "dirhi"(direct horizontal irradiance) W / m ^ 2
- "pressure" in Pa
- "temperature" in °C
- "windspeed" in m / s

- "z0"(roughness length) in m

Other variables:

- list nodes_sinks: class intern list of sinks that are already created

class Transformers

Creates a Generic Transformer object.

Creates a generic transformer with the paramters given in 'nodes_data' and adds it to the list of components 'nodes'.

Parameters:

- **dict de: dictionary containing all information for the** creation of an oemof transformer. At least the following key-value-pairs have to be included:
  - 'label'
  - 'input'
  - 'output'
  - 'output2'
  - 'efficiency'
  - 'efficiency2'
  - 'variable input costs / (CU/kWh)'
  - 'variable output costs / (CU/kWh)'
  - 'variable output costs 2 / (CU/kWh)'
  - 'periodical costs / (CU/kWh)'
  - 'min. investment capacity / (kW)'
  - 'max. investment capacacity / (kW)'
  - 'existing capacity / (kW)'
  - 'Non-Convex Investment'
  - 'Fix Investment Costs / (CU/a)'

transformers()

Creates a transformer object.

Creates transformers objects as defined in 'nodes_data' and adds them to the list of components 'nodes'.

Parameters

- dict nodes_data : dictionary containing data from excel scenario file. The following data have to be provided: label, active, transformer type, input, output, output2, efficiency, efficency2, variable input costs /(CU/kWh), variable output costs /(CU/kWh), existing capacity /(kW), max. investment capacity /(kW), min. investment capacity /(kW), periodical costs /(CU/(kW a))
- dict bus : dictionary containing the busses of the energy system
- list nodes : list of components

genericchp_transformer()

Creates a Generic CHP transformer object.

Creates a generic chp transformer with the paramters given in 'nodes_data' and adds it to the list of components 'nodes'.

class Storages

Creates oemof storage objects as defined in 'nodes_data' and adds them to the list of components 'nodes'.

Inits the storage class.

Parameters:

- dict nodes_data: dictionary containing parameters of storages to be

created.The following data have to be provided:

- 'label'
- 'active'
- 'bus'
- 'existing capacity / (kWh)'
- 'min.investment capacity / (kWh)'
- 'max.investment capacity / (kWh)'
- 'Non-Convex Investments'
- 'Fix Investment Costs /(CU/a)'
- 'input/capacity ratio (invest)'
- 'output/capacity ratio (invest)'
- 'capacity loss'
- 'efficiency inflow'
- 'efficiency outflow'
- 'initial capacity'
- 'capacity min'
- 'capacity max'
- 'variable input costs'
- 'variable output costs'
  - dict busd: dicitionary containing the buses of the energy system
  - list nodes: list of components created before (can be empty)

class Links:

Creates an oemof link object with the given parameters and returns it.

Parameters:

- dict link: dictionary containing parameters of link to be

created.The following data have to be provided:

- 'bus_1'
- 'bus_2'
- 'efficiency'
- 'variable costs /(CU/kWh)'
- 'existing capacity /(kW)'
- 'min. investment capacity /(kW)'
- 'max. investment capacity /(kW)'
- 'periodical costs /(CU/(kW a))'
- 'Non-Convex Investment'
- 'Fix Investment Costs /(CU/a)'

- String label: separate transmission of the label, because there are two

labels for an undirected link

- int outnum: Defines in which direction the link is directed.

Inits the Links class.

Parameters:

- dict nodes_data: dictionary containing data from excel scenario file. The

following data have to be provided:

- 'active'
- 'label'
- '(un)directed'

- dict bus: dictionary containing the buses of the energy system
- list nodes: list of components created before (can be empty)

4.1.3 optimize model

def least_cost_model(energy_system)

Solves a given energy system model.

Solves a given energy system for least costs and returns the optimized energy system.

Parameters

- energy_system : energy system consisting a number of components

Return values

- Model om: solved oemof model

4.1.4 create_graph()

Visualizes the energy system as graph.

Creates, using the library Graphviz, a graph containing all components and connections from "nodes_data" and returns this as a PNG file.

Parameters

- String filepath : path, where the PNG-result shall be saved
- dict nodes_data : dictionary containing data from excel scenario file.
- bool legend : specifies, whether a legend will be added to the graph or not

4.1.5 charts()

Plots model results.

Plots the in- and outgoing flows of every bus of a given, optimized energy system

Parameters

- dict nodes_data : dictionary containing data from excel scenario file
- oemof.solph.models.Model optimization_model: optimized energy system
- energy_system: original (unoptimized) energy system

Return values

- plots plots displaying in and outgoing flows of the energy systems' buses.

4.1.6 prepare_plotly_results()

def program_files.create_results.prepare_plotly_results (*nodes_data*, *optimization_model*, *energy_system*, *result_path*)

Function which prepares the results for the creation of a HTML page.

Creates three pandas data frames and saves them, which are required for creating an interactive HTML result page:

- df_list_of_components: Consists all components with several properties
- df_result_table: Consists timeseries of al components
- df_summary: Consists summarizing results of the modelling

Parameters

- dict nodes_data: dictionary containing data from excel scenario file
- **oemof.solph.models.Model optimization_model: optimized energy system**
  - energy_system: original (unoptimized) energy system
- String result_path: path, where the data frames shall be saved as csv-file

4.1.7 create results

xlsx()

def program_files.create_results.xlsx (*nodes_data*, *optimization_model*, *energy_system*, *filepath*)

Returns model results as xlsx-files.

Saves the in- and outgoing flows of every bus of a given, optimized energy system as .xlsx file

Parameters

- dict nodes_data : dictionary containing data from excel scenario file
- oemof.solph.models.Model optimization_model: optimized energy system
- energy_system: original (unoptimized) energy system
- String filepath: path, where the results will be stored

Return values

- obj '.xlsx' results: xlsx files containing in and outgoing flows of the energy systems' buses.

statistics()

def program_files.create_results.statistics (*nodes_data*, *optimization_model*, *energy_system*)

Returns a list of all defined components with the following information:

component	information
sinks	Total Energy Demand
sources	Total Energy Input, Max. Capacity, Variable Costs, Periodical Costs
transformers	Total Energy Output, Max. Capacity, Variable Costs, Investment Capacity, Periodical Costs
storages	Energy Output, Energy Input, Max. Capacity, Total variable costs, Investment Capacity, Periodical Costs
links	Total Energy Output

Furthermore, a list of recommended investments is printed.

Parameters

- dict nodes_data: dictionary containing data from excel scenario file
- oemof.solph.models.Model optimization_model: optimized energy system
- energy_system: original (unoptimized) energy system

CHAPTER 5

Further Information

- information/links
- information/license
- information/contact
- information/acknowledgements