oemof-flexmex documentation

Jann Launer

Jan 04, 2023

GETTING STARTED

CHAPTER

ONE

OVERVIEW

oemof-flexmex is a sector-integrated multi-node energy system model featuring a lot of flexibility options. Its region, interconnections and components can be adapted flexibly.

oemof-flexmex builds upon the open energy modeling framework oemof, which is an open source, modular toolbox for building energy system models. It hosts different libraries for different purposes. This model, oemof-flexmex, uses oemof.solph for linear optimisation and oemof.tabular for the handling of input data.

The model has been developed in the context of the model comparison project FlexMex. The project is now completed and the model is no longer maintained.

1.1 Energy system

In oemof-flexmex an energy system can be composed of

- demands and supplies
- a variety of energy transformers and storages (such as power plants, batteries, renewable energy plants)
- transmission lines, pipelines

Just as its core, oemof-solph, oemof-flexmex is flexible in modelling different energy carriers, such as electricity, heat, gas or hydrogen. It also allows for defining your own components with the help of oemof.tabular.facades.

1.2 Regions

Each energy system is separated into regions. Regions can be independent from each other (resulting in a number of isolated energy systems) or linked by transmission lines or pipelines (resulting in a network of energy systems). Timeseries for demand and supply can be applied to each region seperately.

1.3 Scenarios

In oemof-flexmex, each scenario defines its own energy system which can include different energy carriers (or sectors), primary energy sources, conversion, storage, transmission and demand. All scenarios are provided with the same set of input data, which consists out of parameters (e.g. capacities) and timeseries (e.g. energy demand or hourly capacity factors for renewable energies). Thus, the scenarios help to model different flexibility options within a given energy system.

CHAPTER

TWO

GETTING STARTED

Contents

2.1 Using oemof-flexmex

2.1.1 Installing the latest (dev) version

Clone oemof-flexmex from github:

```
git clone git@github.com:modex-flexmex/oemof-flexmex.git
```

Now you can install your local version of oemof-flexmex using pip:

```
pip install -e <path/to/oemof-flexmex/root/dir>
```

2.1.2 Requirements

1. To use *oemof-solph*, the core of oemof-flexmex, a LP/MILP solver must be installed. To use the CBC solver install the *coinor-cbc* package:

   ```
   apt-get install coinor-cbc
   ```

   cbc is the default solver. If you chose a different solver, you need to adapt it in oemof_flexmex/optimization.py.

2. oemof-flexmex needs *oemof-tabular* for data preprocessing. Please install the dev version from github rather than installing from PyPi/pip.

   ```
   git clone https://github.com/oemof/oemof-tabular.git
   cd oemof-tabular/
   git checkout dev
   pip install -e ./
   ```

(for further installing issues and their solution, see https://github.com/modex-flexmex/oemo-flex/issues/12)

2.1.3 Required data

Not provided with the github repository:

- Raw input data, see *Raw data*.
- Output template data, see *Postprocessing*.

This data is planned to be published at a later point in time by the project FlexMex.

2.2 Contributing to oemof-flexmex

The project FlexMex, for which this model has been built, is completed and the model is no longer maintained. You are welcome to contribute to a different project within the oemof community.

CHAPTER

THREE

MODEL PIPELINE

Data processing in oemof-flexmex is divided into 4 main steps:

- preprocessing
- inferring
- optimization
- postprocessing

The workflow is managed using the workflow management tool snakemake. Each of the 4 steps is represented by a snakemake rule in the Snakefile, which runs the script of the same name.

The data each step is provided with is held in different forms:

- raw data
- preprocessed data
- optimization results
- postprocessed results

3.1 Raw data

The raw data holds the energy system model definition for all scenarios. It consists of a parameter database (parameters are called *scalars*) and a bunch of timeseries (*sequences* or *profiles*).

The data is expected to be CSV-formatted and is read from `data/In`. The format of timeseries and scalars is described below.

Note: Raw data for FlexMex is not part of the oemof-flexmex github repository but can be provided by the FlexMex project partners.

3.1.1 Scalars

The scalars database defines parameters for all scenarios and regions. It is held in a CSV file called `Scalars.csv`.

The following table shows the first lines of an example `Scalars.csv` and its format:

Scenario	Region	Year	Parameter	Unit	Value
FlexMex1	ALL	ALL	Energy_SlackCost_Electricity	Eur/GWh	50000000
FlexMex1	ALL	ALL	Energy_SlackCost_Electricity	Eur/GWh	5000000
ALL	ALL	ALL	Energy_SlackCost_Heat	Eur/GWh	1000000
ALL	ALL	ALL	Energy_SlackCost_H2	Eur/GWh	1000000
FlexMex1	AT	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	248
FlexMex1	BE	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	584
FlexMex1	CH	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	183
FlexMex1	CZ	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	275
FlexMex1	DE	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	2950
FlexMex1	DK	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	391
FlexMex1	FR	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	3361
FlexMex1	IT	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	2304
FlexMex1	LU	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	58
FlexMex1	NL	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	947
FlexMex1	PL	2050	DemandResponse_Capacity_Electricity_Cooling	MW (el)	1497
FlexMex1	AT	2050	DemandResponse_Capacity_Electricity_HVAC	MW (el)	964
FlexMex1	BE	2050	DemandResponse_Capacity_Electricity_HVAC	MW (el)	2144
FlexMex1	CH	2050	DemandResponse_Capacity_Electricity_HVAC	MW (el)	1027
FlexMex1	CZ	2050	DemandResponse_Capacity_Electricity_HVAC	MW (el)	1117

Scenario: ***string***
Special identifier to address scenario or group of scenarios ('experiment')

Note: The keyword `ALL` can be used as a universal quantifier to avoid repetition.

Region: ***string***
Region identifier

Note: The keyword `ALL` can be used as a universal quantifier to avoid repetition.

Year: ***integer***
Year

Warning: Years support is not fully implemented!

Parameter: ***string***
The parameter name used in the FlexMex project. This is mapped via preprocessing to the components parameters.

Unit: ***string***
Unit of measurement of the given value

> **Warning:** Unit support is incomplete! Especially, there is no check for unit equivalence nor any automatic unit conversion!

Value: *float*
Value of the parameter

3.1.2 Timeseries

Timeseries in oemof-flexmex assign a value to every hour of the year (1…8760). They are held in CSV files with one time index-value pair per line and one timeseries per file.

> **Warning:** The time index is ignored at the moment. It will be overwritten by a `pandas datetimeindex`.

The paths to the timeseries are defined in `flexmex_config/mapping-input-timeseries.yml` per component. If a component has no timeseries defined here, an info line is added to the log output.

The found filenames are interpreted according to the following pattern:

```
{experiment name}_{region code}_{year}.csv
```

Note: `Experiment name` and `year` are ignored at the moment.

The following table shows the first lines of an exemplary time series csv file for heat demand in Austria, which is stored as `data/In/Energy/FinalEnergy/Heat/FlexMex1_AT_2050.csv`.

timeindex	load
1	0.000213222
2	0.000214263
3	0.0002161
4	0.000221314
5	0.000228666

And here is the corresponding entry in `mapping-input-timeseries.yml`:

```
heat-demand:
    profiles:
        heat-demand:
            input-path: Energy/FinalEnergy/Heat
```

3.2 Preprocessing

Preprocessing brings the raw data into the oemof.tabular format. In this step, scalars belonging to a component are mapped to the components model parameters and saved within an input CSV file. Timeseries are attached in a similar way. The so formed input data is held in a `datapackage` format comprising a JSON schema file (meta data) and the CSV files containing the actual data.

3.3 Inferring

3.4 Optimization

Optimization is performed by oemof-solph. Specifically, with the help of oemof.tabular, an `EnergySystem` is created from the data package created in preprocessing.

3.5 Postprocessing

Postprocessing translates the results into an exchange-friendly format defined by the FlexMex project partners. For that, a result template defines the output parameters for each scenario. The oemof-flexmex-internal parameters are recalculated and mapped to the FlexMex parameter names.

The results template is provided by the FlexMex project partners. It consists of an output directory structure and a scaffold Scalars.csv output file (with no values). It should be placed in the path:

```
flexmex_config/output_template/
```

The mapping is read from the two CSV files:

```
flexmex_config/mapping-output-scalars.csv
flexmex_config/mapping-output-timeseries.yml
```

CHAPTER

FOUR

MODEL STRUCTURE

Contents

4.1 Model structure

The model structure defines the format of the preprocessed data which is ready to be optimized by oemof.

4.1.1 Elements

All busses are defined in `oemof_flexmex/results/scenario/01_preprocessed/data/elements/bus.csv`.

The preprocessed component data is also stored in `oemof_flexmex/results/scenario/01_preprocessed/data/elements`

The filenames for the components are of the form

```
{carrier}-{tech}.csv
```

(e.g. `electricity-demand.csv`, `gas-bpchp.csv`).

The first columns of the component scalars file are similar in all of the files. They contain the following information:

- **region**: Region of a component. Modelled *regions* are defined here
- **name**: Unique name (`'region-carrier-tech'`, eg. `'LU-gas-bpchp'`, `'AT-electricity-airsourcehp'`)
- **type**: Type of oemof.tabular.facade
- **carrier**: Energy sector according to carrier (e.g. solar, wind, biomass, coal, lignite, uranium, oil, gas, methane, hydro, waste, electricity, heat).
- **tech**: Specification of the technology (e.g. pv, onshore, offshore, battery, demand, curtailment, shortage, transmission, ror, st, ocgt, ccgt, extchp, bpchp)

Following these columns, the attributes for the respective components are defined. The number and kind of attributes varies between components.

4.1.2 Sequences

The input timeseries are combined into a new set of CSV files, with one file per technology. The preprocessed sequences are stored in

```
results/{scenario name}/01_preprocessed/data/sequences/{technology}_profile.csv
```

The filenames are of the form

```
<carrier>-<tech>_<profile>.csv
```

(e.g. `wind-offshore_profile.csv`, `electricity-demand_profile.csv`).

Each sequence file contains the hourly profile of all the regions, organized in rows. They are indexed by a pandas datetimeindex. The column names have the structure `{region}-{technology}-profile`.

4.2 Available components

These components are available in oemof-flexmex.

name	path	description
electricity-demand	component_attrs/electricity-demand.csv	Electricity demand
heat-shortage	component_attrs/heat-shortage.csv	Shortage backup to keep model solvable
electricity-shortage	component_attrs/electricity-shortage.csv	Shortage backup to keep model solvable
electricity-transmission	component_attrs/electricity-transmission.csv	Electrical transmission between regions
solar-pv	component_attrs/solar-pv.csv	Solar pv capacities
wind-offshore	component_attrs/wind-offshore.csv	Offshore wind capacities
wind-onshore	component_attrs/wind-onshore.csv	Onshore wind capacities
electricity-curtailment	component_attrs/electricity-curtailment.csv	Curtailment of non-dispatchable electricity
ch4-gt	component_attrs/ch4-gt.csv	Open cycle gas turbine gas turbine capacities (natural gas driven)
uranium-nuclear-st	component_attrs/uranium-nuclear-st.csv	Nuclear power plant capacities (steam turbine)
ch4-bpchp	component_attrs/ch4-bpchp.csv	Backpressure turbine CHP
ch4-boiler-small	component_attrs/ch4-boiler.csv	Peak load boiler
ch4-boiler-large	component_attrs/ch4-boiler.csv	Peak load boiler
ch4-extchp	component_attrs/ch4-extchp.csv	Extraction turbine CHP
electricity-pth	component_attrs/electricity-pth.csv	Resistive heater (Power-to-Heat)
heat-demand	component_attrs/heat-demand.csv	Heat demand
electricity-h2_cavern	component_attrs/electricity-h2_cavern.csv	Large scale H2 storage
electricity-liion_battery	component_attrs/electricity-liion_battery.csv	Li-ion battery storage
electricity-heatpump-small	component_attrs/electricity-heatpump.csv	Airsource compression heatpump
electricity-heatpump-large	component_attrs/electricity-heatpump.csv	Airsource compression heatpump
heat-storage-small	component_attrs/heat-storage.csv	Small sensible heat storage
heat-storage-large	component_attrs/heat-storage.csv	Large scale sensible heat storage
heat-excess	component_attrs/heat-excess.csv	Heat excess
hydro-reservoir	component_attrs/hydro-reservoir.csv	Hydro reservoir
electricity-bev	component_attrs/electricity-bev.csv	Battery electric vehicle

4.3 Component attributes

The component's attributes are defined in separate csv files contained in `oemof-flexmex/model_structure/component_attrs/`

4.3.1 Extra parameters

tabular supports handing over extra `output_parameters` and `input_parameters` to the components' classes. These have to be given as `dict`'s in the corresponding CSV field. If you want to pass more than two parameters:

A) Enclose the `dict` with quotes and use double-quotes in it (*less readable*).

OR

B) Make the CSV file semicolon-separated and separate the output_parameters and/or input_parameters with commas (*better readable*).

   More over, all component `read_csv()` function calls in `preprocessing.csv` must be adapted to the new separator (`sep=';'`).

   See https://github.com/modex-flexmex/oemo-flex/issues/57 for details.

CHAPTER

FIVE

WHAT'S NEW

These are new features and improvements of note in each release

Releases

5.1 v0.0.1 (2020)

First release.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search