oemof

Release 0.4.0.beta0

Apr 02, 2020

Contents

CHAPTER 1

Overview

docs	
tests	
package	

Open Energy Modelling Framework - Python toolbox for energy system modelling and optimisation.

The omeof project aims to be a loose organisational frame for tools in the wide field of (energy) system modelling. Every project is managed by their own developer team but we share some developer and design rules to make it easier to understand each others tools.

All projects are in different states and some even may not have a stable release but all projects are open to join. We do not belong to a specific institution and everybody is free to join the developer teams and will have the same rights. There is no higher decission level.

This repository is also used to organise everything for the oemof community.

- Webconference dates
- Real life meetings
- Website and Mailinglist
- General communication

1.1 Projects with stable releases

- **oemof-solph** A model generator for energy system modelling and optimisation (LP/MILP).

- **TESPy** Thermal Engineering Systems in Python (TESPy). This package provides a powerful simulation toolkit for thermal engineering plants such as power plants, district heating systems or heat pumps.

1.2 Projects in an early state

- **DHNx** District heating system optimisation and simulation models

All project libraries are free software licenced under the MIT license.

1.3 Installation

To install the *oemof.solph* package (previously just *oemof*), please use

```
pip install https://github.com/oemof/oemof-solph/archive/master.zip"
```

1.4 Documentation

https://oemof.readthedocs.io/

1.5 Development

To run the all tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER 2

Installation

To install the *oemof.solph* package (previously just *oemof*), please use

```
pip install https://github.com/oemof/oemof-solph/archive/master.zip"
```

CHAPTER 3

Usage

Open Energy Modelling Framework - Python toolbox for energy system modelling and optimisation.

The omeof project aims to be a loose organisational frame for tools in the wide field of (energy) system modelling.

Current oemof libraries

- *oemof-solph*
- *oemof-thermal*
- *cydets*
- *demandlib*
- *feedinlib*
- *tespy*

3.1 oemof-solph

The solph library is designed to create and solve linear or mixed-integer linear optimization problems. It is based on optimization modelling language pyomo.

To use solph at least one linear solver has to be installed on your system. See the pyomo installation guide to learn which solvers are supported. Solph is tested with the open source solver *cbc* and the *gurobi* solver (free for academic use). The open *glpk* solver recently showed some odd behaviour.

The formulation of the energy system is based on the oemof-network library but contains additional components such as storages. Furthermore the network class are enhanced with additional parameters such as efficiencies, bounds, cost and more. See the API documentation for more details. Try the examples to learn how to build a linear energy system.

3.2 oemof-thermal

Coming soon...

3.3 cydets

Cycle Detection in Time Series (CyDeTS). An algorithm to detect cycles in times series along with their respective depth-of-cycle (DoC) and duration.

3.4 demandlib

The demandlib library can be used to create load profiles for elctricity and heat knowing the annual demand. See the documentation of the demandlib for examples and a full description of the library.

3.5 feedinlib

The feedinlib library serves as an interface between Open Data weather data and libraries to calculate feedin timeseries for fluctuating renewable energy sources.

It is currently under revision (see here for further information). To begin with it will provide an interface to the pvlib and windpowerlib and functions to download MERRA2 weather data and open_FRED weather data. See documentation of the feedinlib for a full description of the library.

3.6 tespy

Coming soon...

CHAPTER 4

Reference

4.1 oemof

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.2 Documentation improvements

oemof could always use more documentation, whether as part of the official oemof docs, in docstrings, or even on the web in blog posts, articles, and such.

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/oemof/oemof/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

5.4 Development

To set up *oemof* for local development:

1. Fork oemof (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:oemof/oemof.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 6

Authors

- oemof developer group - https://oemof.org

CHAPTER 7

Changelog

7.1 0.4.0.beta0 (2020-04-04)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

o

Index

O