The oemof.thermal package

How to use the new thermal components

Caroline Möller, Christoph Pels Leusden, Franziska Pleißner, Jakob Wolf, Jann Launer, Marie-Claire Gering, Silke Köhler

Reiner Lemoine Institut, Beuth University of Applied Sciences

May 13, 2020

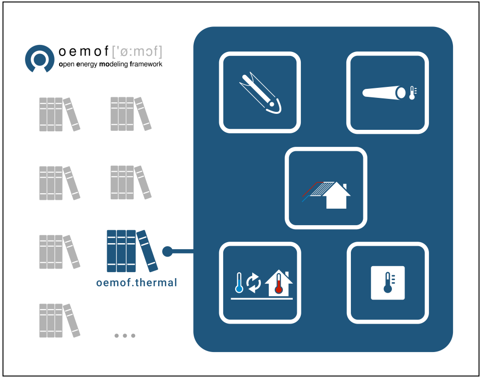
Some background information

- oemof.thermal was developed within the project "oemof_heat" by Reiner Lemoine Institut and Beuth University of Applied Sciences
- Project duration: 07/2017 - 09/2020
- Currently released: v0.0.2
- Last release from our side (v0.0.3) will be in Summer 2020

oemof_heat
['ø:mɔf_'hēt]

The oemof.thermal package

- Provides tools to model thermal energy components in energy system optimization as an extension of oemof.solph
  - Stratified thermal storage
  - Solar thermal collector
  - Concentrating solar power
  - Compression heat pump and chiller
  - To be released soon: absorption heat pump - PR #74



How to install the oemof.thermal package

Install oemof.thermal by running

```
pip install oemof.thermal
```

in your virtualenv.

In your code, you can import modules like:

```
from oemof.thermal import concentrating_solar_power
```

Find the examples here:
https://github.com/oemof/oemof-thermal/tree/master/examples

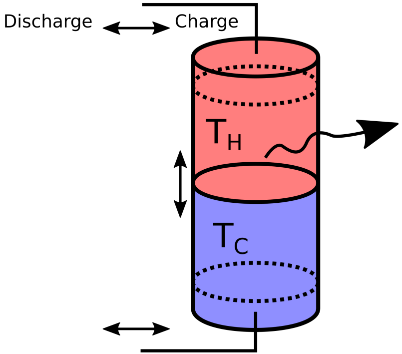
Find the documentation at https://oemof-thermal.readthedocs.io.

How to use the oemof.thermal package

- There are two ways:
  - Use the oemof.thermal facades, which are based on the oemof.tabular.facades module, for a simple way to set up an oemof.solph component for your energy system - Facades are provided for the three technologies "stratified thermal storage", "solar thermal collector" and "concentrating solar power".
  - Use the collection of functions for each technology independently to perform pre-calculations of an optimization model or postprocess optimization results

Stratified thermal storage

- Large-scale sensible heat storage with perfect stratification
- Two zones with cold (T_C) and hot(T_H) temperature
- When charging/discharging the storage the thermocline moves down and up.
- Losses through the surface depend on the size of the hot and cold zone.
- For the storage investment mode, you provide a diameter, but leave height and capacity open.

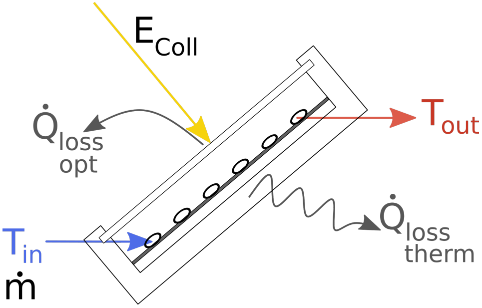


Stratified thermal storage

```python
from oemof import solph
from oemof.thermal.facades import StratifiedThermalStorage

bth = solph.Bus('heat')

thermal_storage = StratifiedThermalStorage(
    label='thermal_storage',
    bus=bth,
    diameter=2,
    temp_h=95,
    temp_c=60,
    temp_env=10,
    u_value=u_value,
    expandable=True,
    capacity_cost=50,
    storage_capacity_cost=400,
    min_storage_level=0.05,
    max_storage_level=0.95,
    efficiency=0.98,
    marginal_cost=0.0001
)
```

Solar thermal collector

- Provides the heat of a flat plate collector while considering collector and ambient temperatures
- The processing of the irradiance data is done by the pvlib which calculates the total in-plane irradiance (E_{Coll}) according to the location and the azimuth and tilt angle of the collector
- The optical efficiency and thermal loss parameters (usually part of the technical data sheet) must be provided.
- Further efficiencies (e.g. for electrical consumptions of pumps or peripheral thermal losses) can be provided.

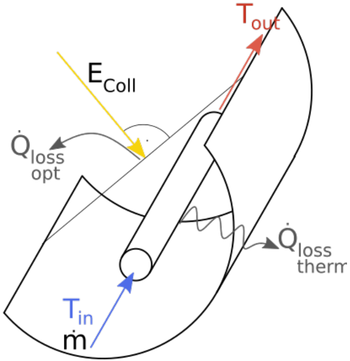


Solar thermal collector

```
from oemof import solph
from oemof.thermal.facades import SolarThermalCollector

bth = solph.Bus(label='thermal')
bel = solph.Bus(label='electricity')

collector = SolarThermalCollector(
    label='solar_collector',
    heat_out_bus=bth,
    electricity_in_bus=bel,
    electrical_consumption=0.02,
    peripheral_losses=0.05,
    aperture_area=1000,
    latitude=52.2443,
    longitude=10.5594,
    collector_tilt=10,
    collector_azimuth=20,
    eta_0=0.73,
    a_1=1.7,
    a_2=0.016,
    temp_collector_inlet=20,
    delta_temp_n=10,
    irradiance_global=input_data['global_horizontal_W_m2'],
    irradiance_diffuse=input_data['diffuse_horizontal_W_m2'],
    temp_amb_col=input_data['temp_amb'],
)
```

Concentrating solar power

- Differences to the solar thermal collector:
  - Consideration of cleanliness
  - Consideration of an incidence angle modifier which adapts the optical efficiency
  - Implementation of a second method to determine the heat losses
  - Only the direct irradiation is considered

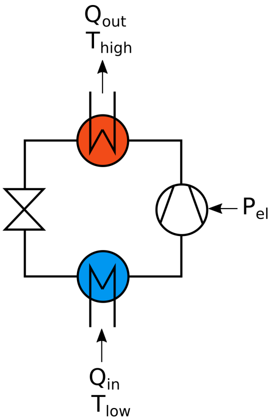


Concentrating solar power

```
from oemof import solph
from oemof.thermal.facades import ParabolicTroughCollector

bth = solph.Bus(label='thermal_bus')
bel = solph.Bus(label='electrical_bus')

collector = ParabolicTroughCollector(
    label='solar_collector',
    heat_bus=bth,
    electrical_bus=bel,
    electrical_consumption=0.05,
    additional_losses=0.2,
    aperture_area=1000,
    loss_method='Janotte',
    irradiance_method='horizontal',
    latitude=23.614328,
    longitude=58.545284,
    collector_tilt=10,
    collector_azimuth=180,
    x=0.9,
    a_1=-0.00159,
    a_2=0.0000977,
    eta_0=0.816,
    c_1=0.0622,
    c_2=0.00023,
    temp_collector_inlet=435,
    temp_collector_outlet=500,
    temp_amb=input_data['t_amb'],
    irradiance=input_data['E_dir_hor']
)
```

Compression heat pump and chiller

- Calculation of the COP based on the temperatures
- Define a quality grade to reduce the carnot efficiency
- Icing can be considered when using the ambient temperature as heat source.
- This component does not exist as facade.
- The COP of a compression heat pump is precalculated and then used as an input of a transformer (oemof.solph component).

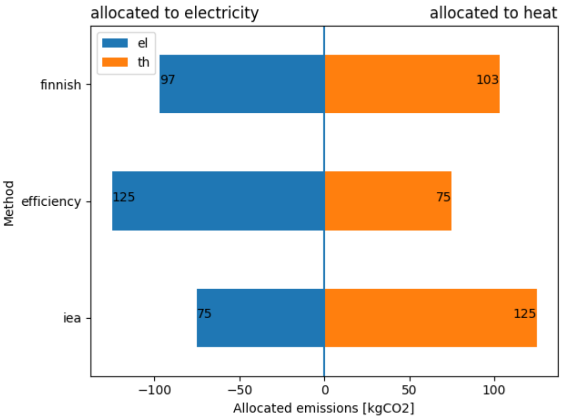


Compression heat pump and chiller

```
from oemof.thermal.compression_heatpumps_and_chillers import calc_cops
from oemof.solph import Transformer

bel = solph.Bus(label='electricity')
bth = solph.Bus(label='thermal')

COP = calc_cops(
    temp_high=40,
    temp_low=data['ambient_temperature'],
    quality_grade=0.4,
    temp_threshold_icing=2,
    factor_icing=0.8,
    mode='heat_pump'
    )

energysystem.add(solph.Transformer(
    label='heat_pump',
    inputs={bel: solph.Flow()},
    outputs={bth: solph.Flow(nominal=25, variable_costs=5)},
    conversion_factors={b_heat: COP}))
```

Cogeneration: Emission allocation

- The module is designed to hold functions that are helpful when modeling components that generate more than one type of output.
- Currently there are three different methods that can be used to allocate the emissions to the two outputs of a unit that produces electricity and heat.



Questions?

- If you have further questions:
  - Use the oemof forum at the openmod initiative: https://forum.openmod-initiative.org/tags/oemof
  - Contact: caroline.moeller@rl-institut.de