ZNES Zentrum für nachhaltige Energiesysteme
Hochschule | Europa-Universität Flensburg

oemof user meeting

Modelling a compressed air energy storage (CAES) as a complex oemof component

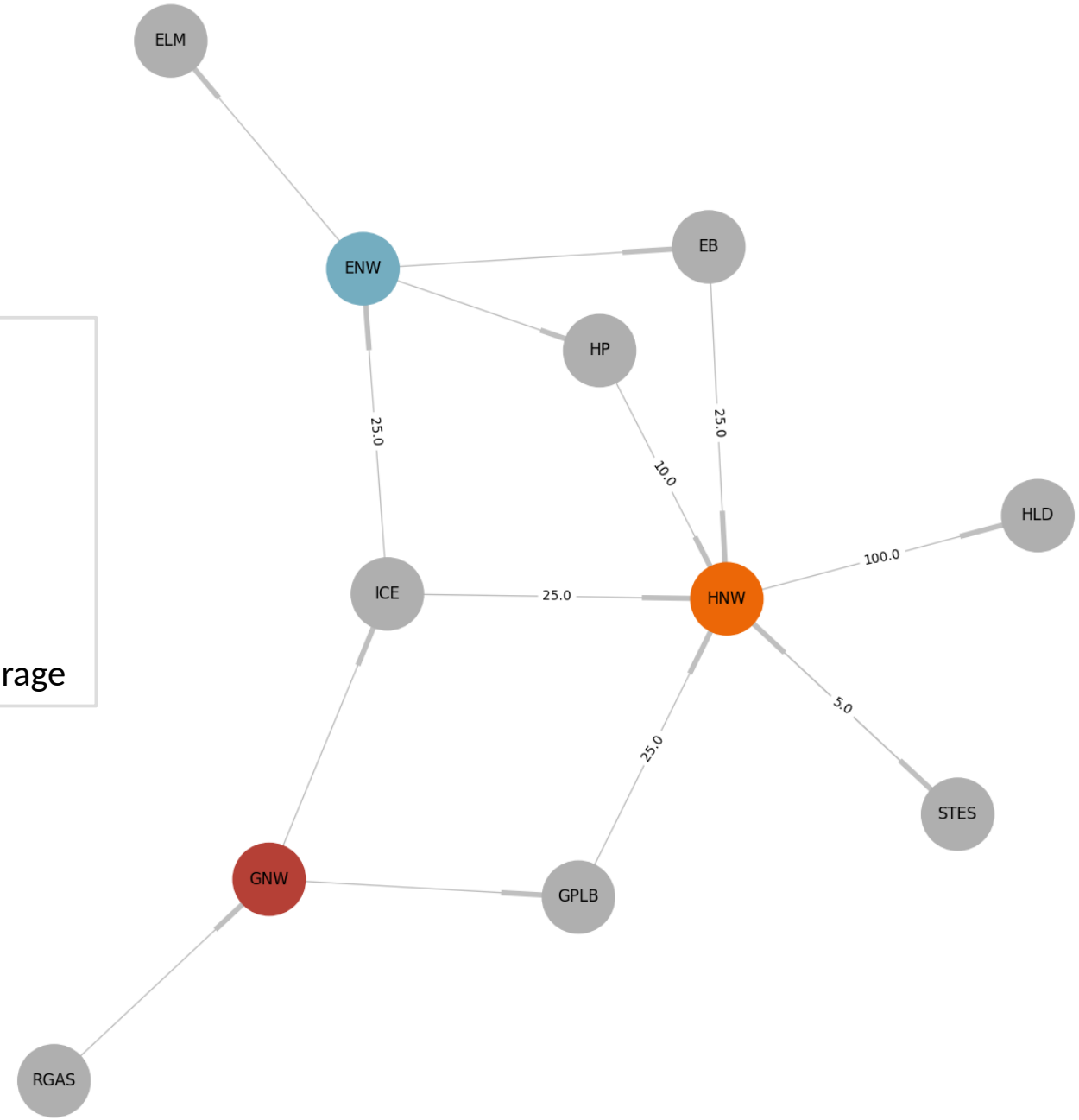
Berlin, 10.05.2017

Outline

- Basic component logic for investor-centered models
- When to create specific models?
- How to represent specific models?
- Modelling a compressed air energy storage in oemof
  - Basic principle
  - Plant characteristics
  - Abstract model
  - Considerations
  - Implementation
- Validation
  - Unit Commitment
  - Partial load behaviour
- Conclusion

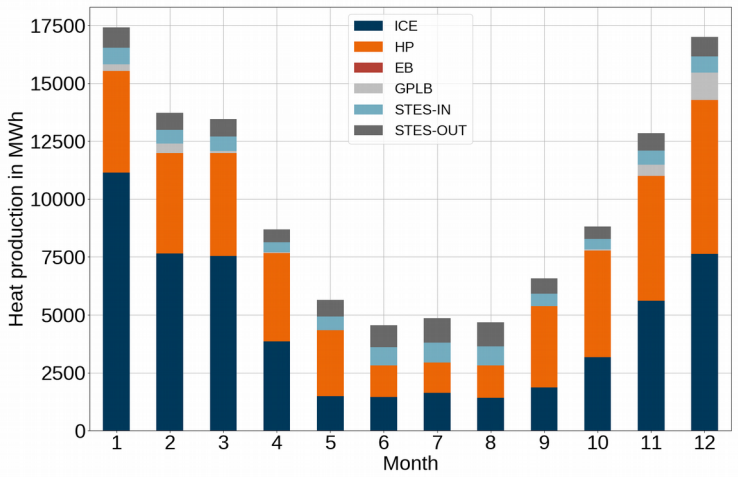
Basic component logic (investor perspective)

ELM: Electricity market
ENW: Electricity network
EB: Electric boiler
GNW: Gas network
GPLD: Gas peak load boiler
HP: Heat pump
ICE: Internal combustion engine
HNW: Heating network
HLD: Heat load
STES: Sensible thermal energy storage

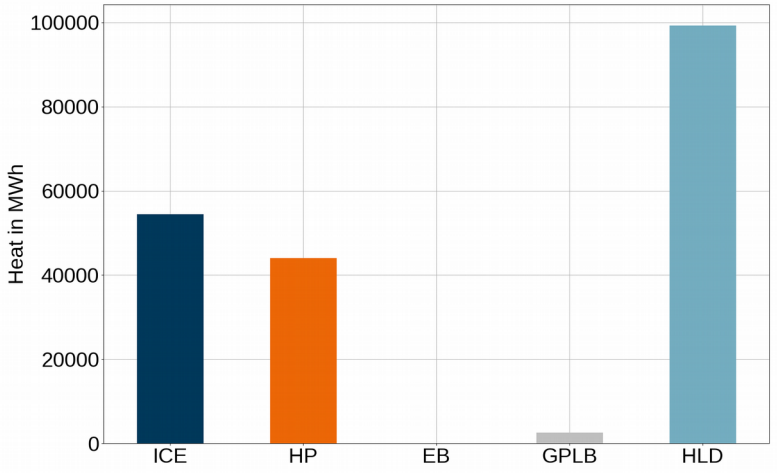


Basic component logic (investor perspective)

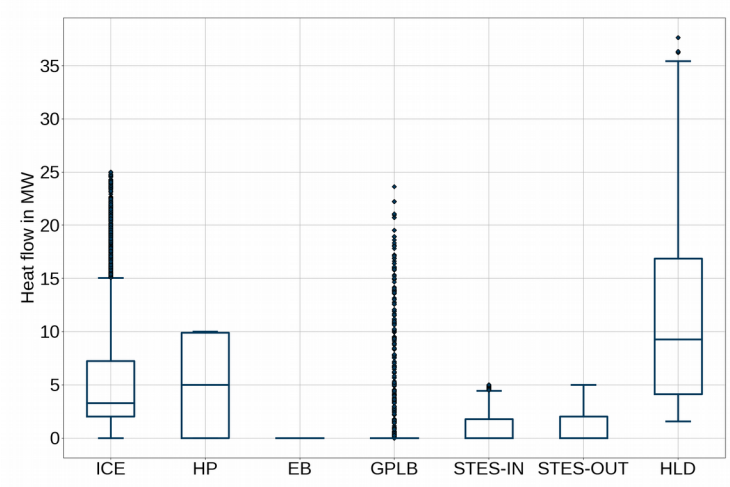
Monthly heat production



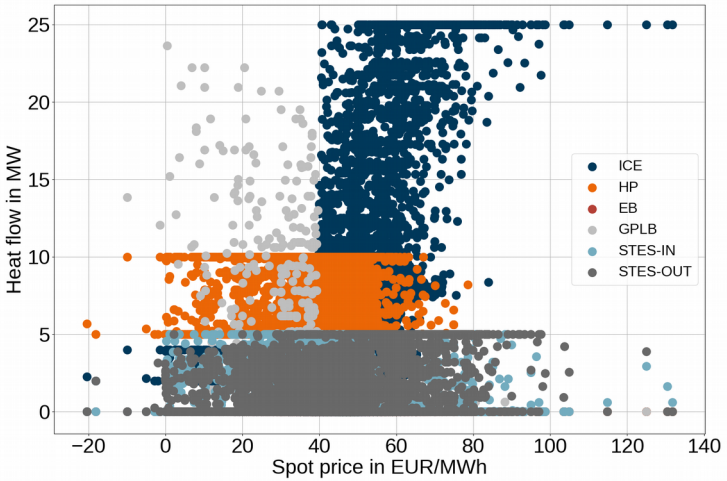
Annual heat production



General statistics



Unit commitment



When to create specific models?

- Modelling of detailed technical topologies

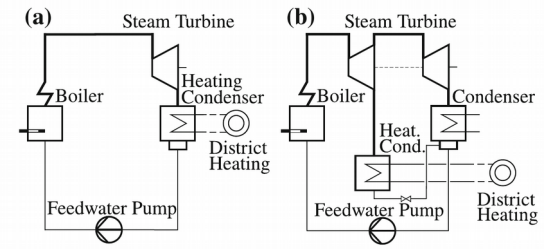
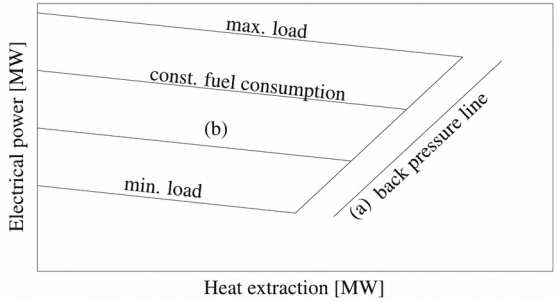


Fig. 1 **a** Back pressure turbine, **b** extraction condensing turbine

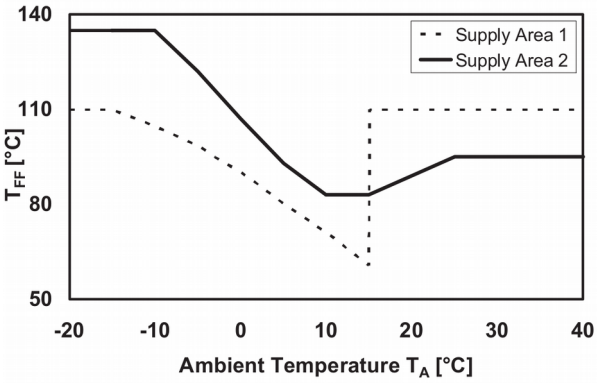
[1]

- Focus on component internal behaviour



[1]

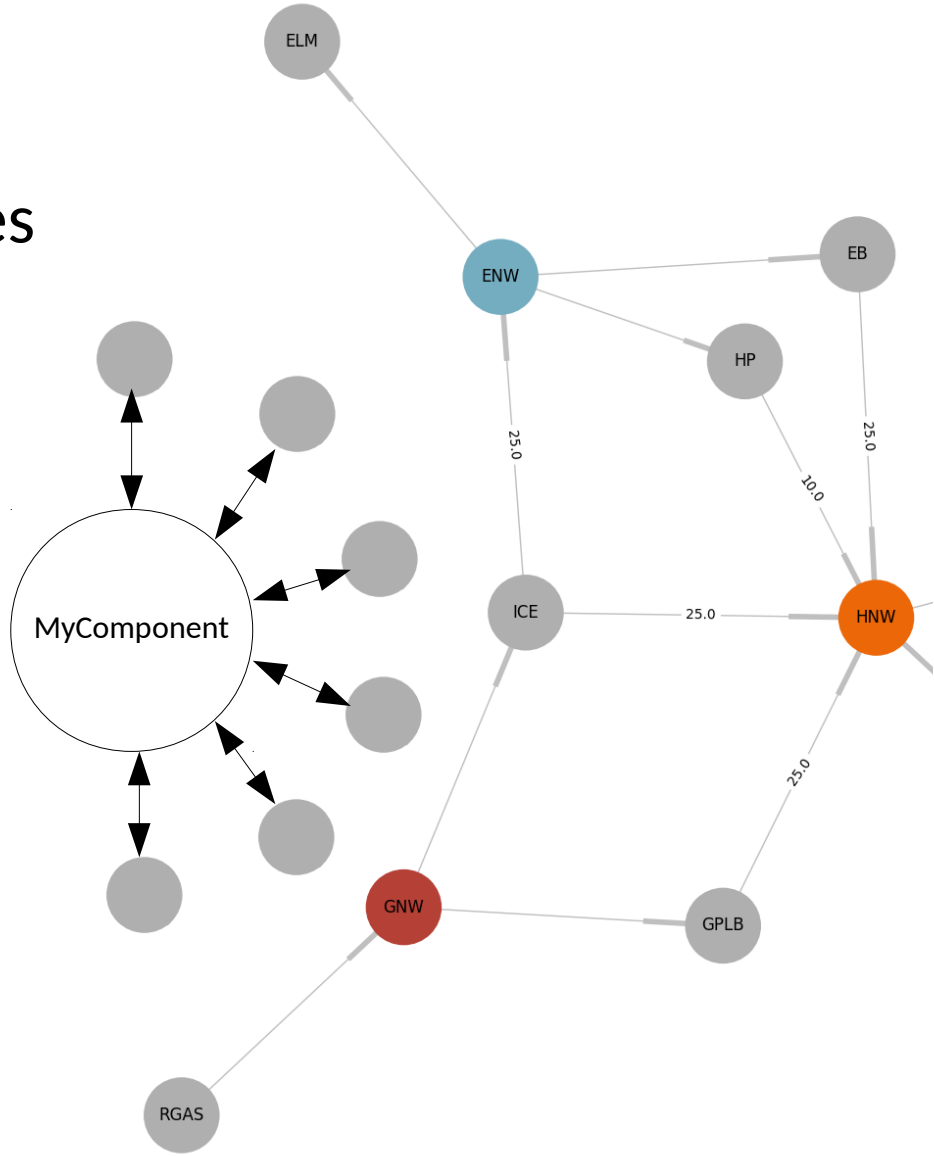
- Quantification of different metrics



[2]

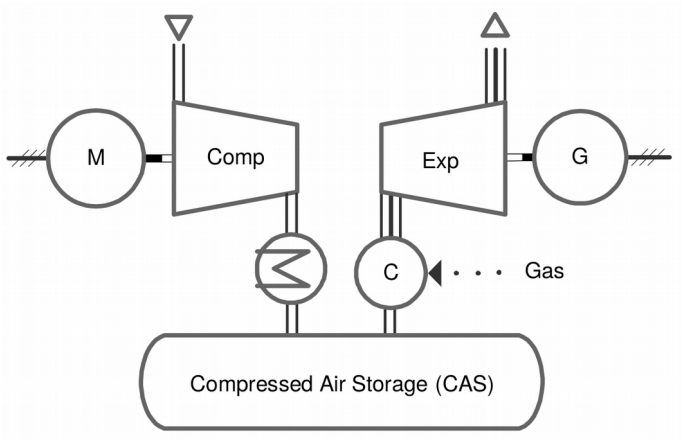
How to represent specific models?

- A complex component can
  - have N/M in- and outflows
  - have N+M connections to busses
  - have detailed internal logic
- Representation is possible
  - through basic components
  - within specific models
- When to apply which form of representation?



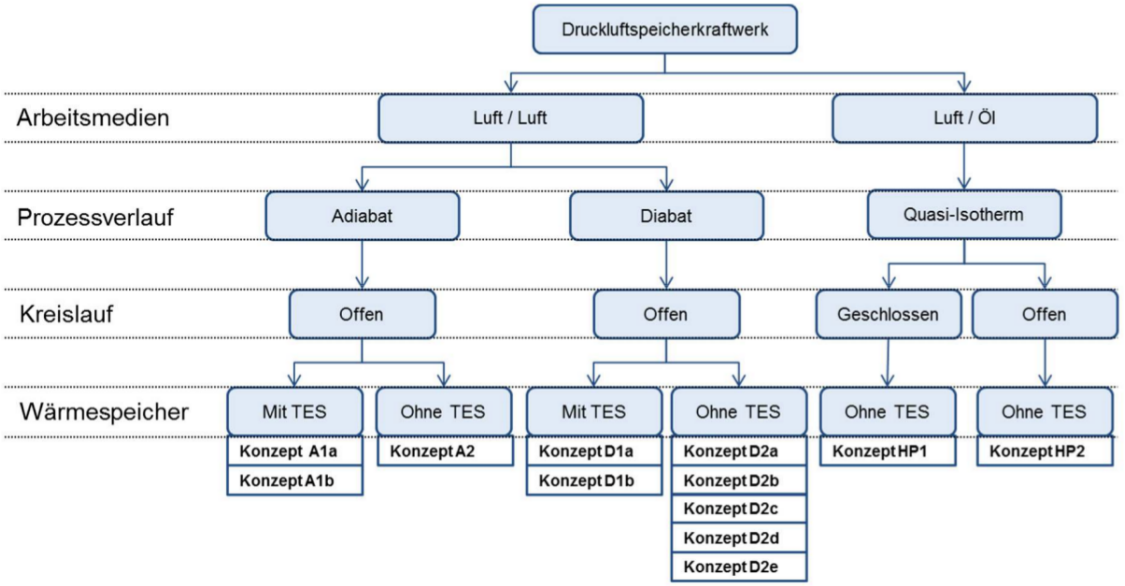
Modelling of compressed air energy storage

Basic principle (adiabatic)



[3]

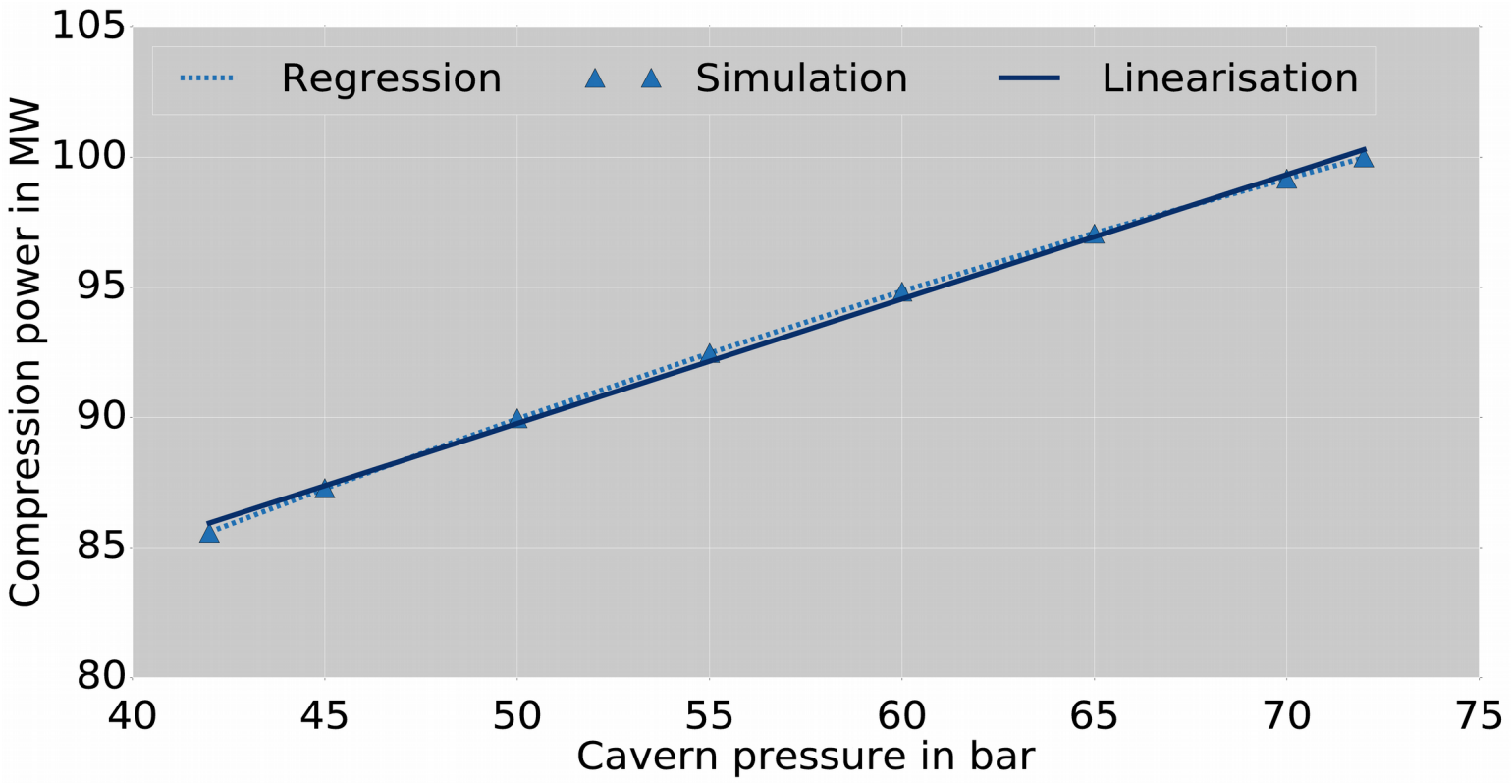
Different concepts (to be modelled as MILP)



[4]

Modelling of compressed air energy storage

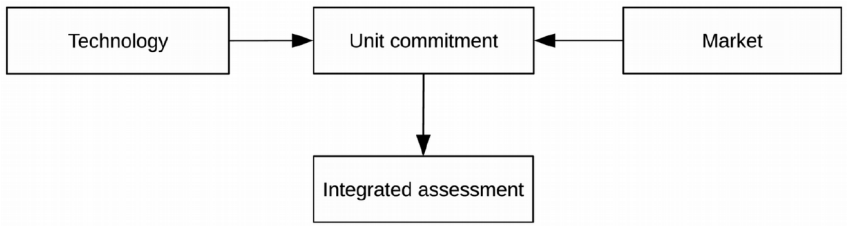
Characteristics based on thermal modelling (extract)

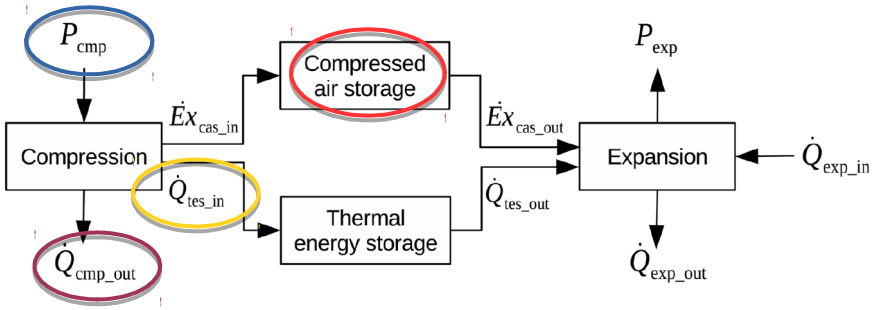


[5]

Modelling of compressed air energy storage

Description as MILP (extract)





3.1 Elektrische Leistung

Die realisierbare Maximalleistung ist abhängig vom Füllstand der Kaverne.

$$P_{cmp\_max}(t) = m_{cmp\_max} CAS_{fil}(t-1) + b_{cmp\_max} \; \forall t \in [1, t_{max}] \tag{3.1}$$

$$P_{cmp\_max}(t) = b_{cmp\_max} \; \forall t \notin [1, t_{max}] \tag{3.2}$$

Die Leistung darf nur Werte zwischen der Minimalleistung und Maximalleistung annehmen.

$$P_{cmp\_min} ST_{cmp}(t) \leq P_{cmp}(t) \leq P_{cmp\_max}(t) \; \forall t \in T \tag{3.3}$$

3.2 Wärmeströme

Der insgesamt austretende Wärmestrom hängt von der elektrischen Leistung ab.

$$\dot{Q}_{cmp}(t) = m_{cmp\_q} P_{cmp}(t) + b_{cmp\_q} ST_{cmp}(t) \; \forall t \in T \tag{3.4}$$

Der insgesamt austretende Wärmestrom teilt sich in einen Abwärmestrom und einen Eingangswärmestrom des thermischen Energiespeichers auf.

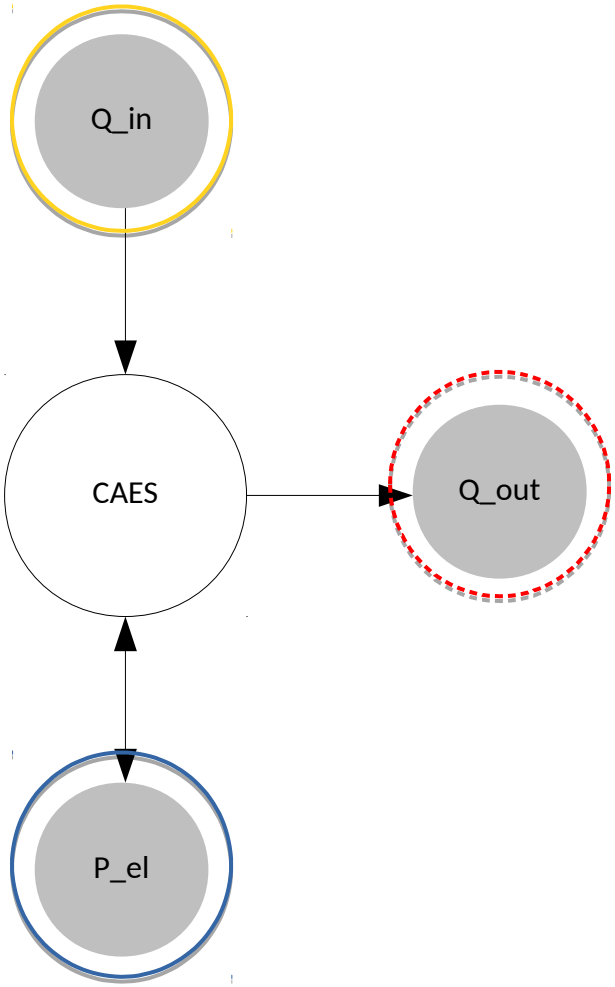
$$\dot{Q}_{cmp}(t) = \dot{Q}_{cmp\_out}(t) + \dot{Q}_{tes\_in}(t) \; \forall t \in T \tag{3.5}$$

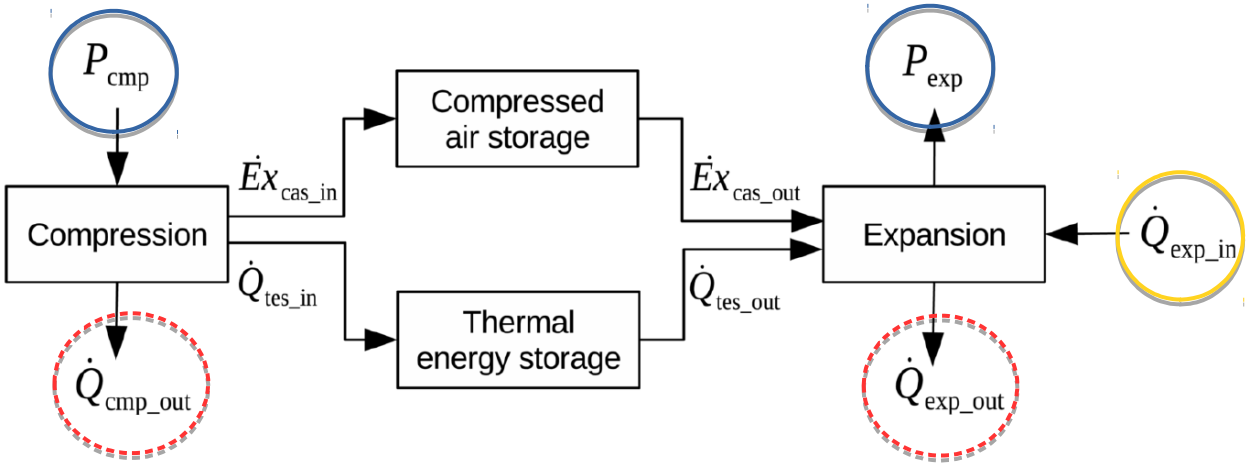
Der Abwärmestrom und Eingangswärmestrom des thermischen Energiespeichers stehen in einem fixen Verhältnis zueinander.

$$r_{cmp\_tes} \dot{Q}_{cmp\_out}(t) = (1 - r_{cmp\_tes}) \dot{Q}_{tes\_in}(t) \; \forall t \in T \tag{3.6}$$

Modelling of compressed air energy storage

- Multiple flows of different types
  - Electricity
  - Heat (omitted)
  - Fuel (optional)
- Connections to different busses
- Implementation as opt. module (caes.py)





Modelling of compressed air energy storage

Connecting flows

```
37    def connections(es=None, electrical_balance=None, gas_balance=None):
38        """
39        Create the basic connections for a CAES plant.
40
41        The connections are added as `oemof.solph.network.Sink` and
42        `oemof.solph.network.Source` to the passed
43        `oemof.solph.network.EnergySystem` which is returned.
44
45        Parameters
46        ----------
47        es : `oemof.solph.network.EnergySystem`
48
49        electrical_balance : `string` Label of connected `oemof.solph.network.Bus`
50
51        gas_balance : `string` Label of connected `oemof.solph.network.Bus`
52
53        params : `dict` with specific plant parameters
54
55        """
56        if es is not None and electrical_balance is not None:
57            solph.Sink(label='caes_p_in',
58                       inputs={es.groups[electrical_balance]:
59                               solph.Flow()})
60            solph.Source(label='caes_p_out',
61                         outputs={es.groups[electrical_balance]: solph.Flow()})
62            if gas_balance is not None:
63                solph.Sink(label='caes_q_in',
64                           inputs={es.groups[gas_balance]: solph.Flow()})
65            logging.info('CAES connections have been constructed successfully.')
66        else:
67            logging.warning('An EnergySystem and el. balance must be passed!')
68
69        return es
```

Modelling of compressed air energy storage

Abstract plant model

```
72    def model(es=None, om=None, electrical_balance=None, gas_balance=None,
73              params=params):
74        """
75        Create a detailed model for a CAES plant based on a passed parameter set.
76
77        The model is added as a pyomo.Block to the passed
78        `oemof.solph.models.OperationalModel` which is returned.
79
80        Parameters
81        ----------
82        es : `oemof.solph.network.EnergySystem`
83
84        om : `oemof.solph.models.OperationalModel`
85
86        electrical_balance : `string` Label of connected `oemof.solph.network.Bus`
87
88        gas_balance : `string` Label of connected `oemof.solph.network.Bus`
89
90        params : `dict` with specific plant parameters
91
92        """
```

Specific parameter set

```
10    # dictionary with parameters for a specific CAES plant
11    # based on thermal modelling and linearization techniques
12    params = {
13        "cav_e_in_b": 0,
14        "cav_e_in_m": 0.6457267578,
15        "cav_e_out_b": 0,
16        "cav_e_out_m": 0.3739636077,
17        "cav_eta_temp": 1.0,
18        "cav_level_max": 211.11,
19        "cmp_p_max_b": 86.0918959849,
20        "cmp_p_max_m": 0.0679999932,
21        "cmp_p_min": 1,
22        "cmp_q_out_b": -19.3996965679,
23        "cmp_q_out_m": 1.1066036114,
24        "cmp_q_tes_share": 0,
25        "exp_p_max_b": 46.1294016678,
26        "exp_p_max_m": 0.2528340303,
27        "exp_p_min": 1,
28        "exp_q_in_b": -2.2073411014,
29        "exp_q_in_m": 1.129249765,
30        "exp_q_tes_share": 0,
31        "tau": 0.25,
32        "tes_eta_temp": 1.0,
33        "tes_level_max": 1000000.0
34    }
```

Modelling of compressed air energy storage

Compressor power

```
157             # Compression: Capacity on markets
158             def cmp_p_constr_rule(m, t):
159                 return (caes.cmp_p[t] == om.flow[es.groups[electrical_balance],
160                                                  es.groups['caes_p_in'], t])
161             caes.cmp_p_constr = po.Constraint(
162                 om.TIMESTEPS, rule=cmp_p_constr_rule)
```

Turbine power

```
215             # Expansion: Capacity on markets
216             def exp_p_constr_rule(m, t):
217                 return (caes.exp_p[t] ==
218                         om.flow[es.groups['caes_p_out'],
219                                 es.groups[electrical_balance], t])
220             caes.exp_p_constr = po.Constraint(
221                 om.TIMESTEPS, rule=exp_p_constr_rule)
```

Fuel consumption (optional)

```
260             # Expansion: Fuel allocation
261             if gas_balance is not None:
262                 def exp_q_fuel_constr_rule(m, t):
263                     return (caes.exp_q_fuel_in[t] ==
264                             om.flow[es.groups[gas_balance],
265                                     es.groups['caes_q_in'], t])
266                 caes.exp_q_fuel_constr = po.Constraint(
267                     om.TIMESTEPS, rule=exp_q_fuel_constr_rule)
```

„linking equalities“

Modelling of compressed air energy storage

Result processing

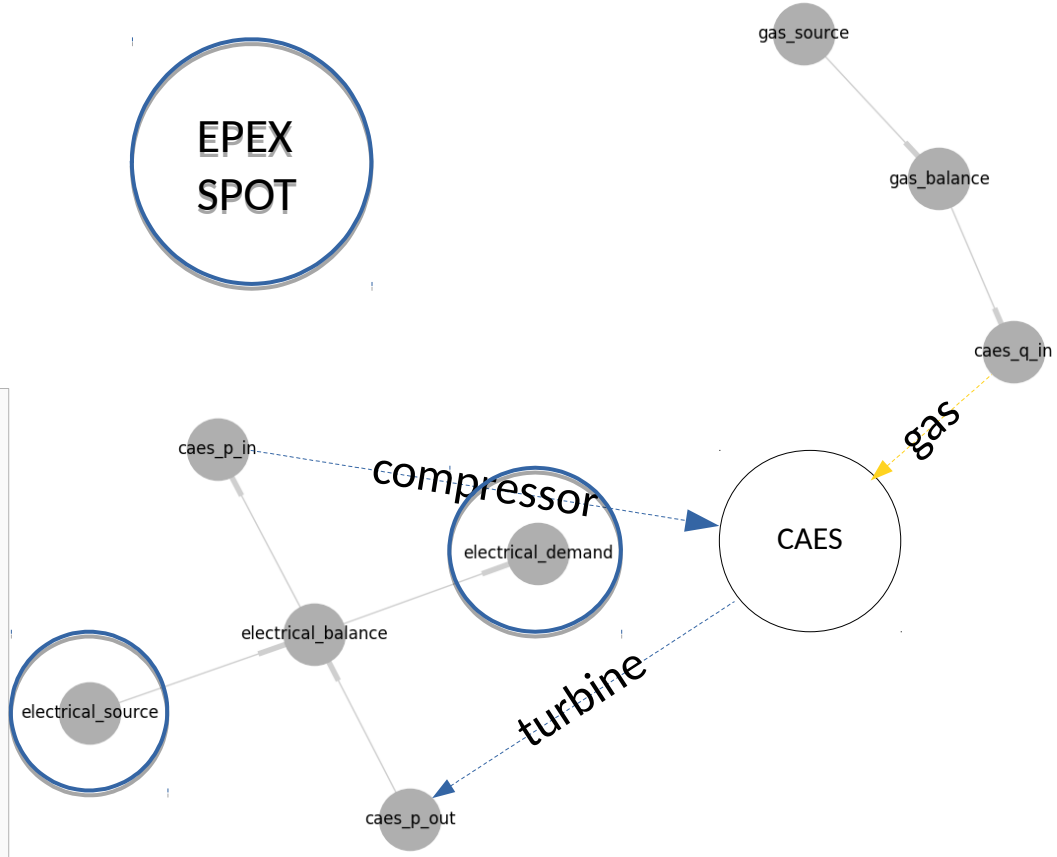
```
335    def results(es=None, om=None):
336        """
337        Create the specific internal results for a CAES plant (exemplary).
338
339        Parameters
340        ----------
341        es : `oemof.solph.network.EnergySystem`
342
343        om : `oemof.solph.models.OperationalModel`
344
345        """
346        if es is not None and om is not None:
347            df = pd.DataFrame(index=es.timeindex)
348            df['exp_p'] = [om.CAESBlock.exp_p[t].value
349                           for t in om.TIMESTEPS]
350            df['cmp_p'] = [om.CAESBlock.cmp_p[t].value
351                           for t in om.TIMESTEPS]
352            df['exp_q_fuel'] = [om.CAESBlock.exp_q_fuel_in[t].value
353                                for t in om.TIMESTEPS]
354            df['cav_level'] = [om.CAESBlock.cav_level[t].value
355                               for t in om.TIMESTEPS]
356            df['tes_level'] = [om.CAESBlock.tes_level[t].value
357                               for t in om.TIMESTEPS]
358            logging.info('CAES results have been created successfully.')
359        else:
360            logging.warning('An EnergySystem and OperationalModel must be passed!')
361
362        return df
```

Modelling of compressed air energy storage

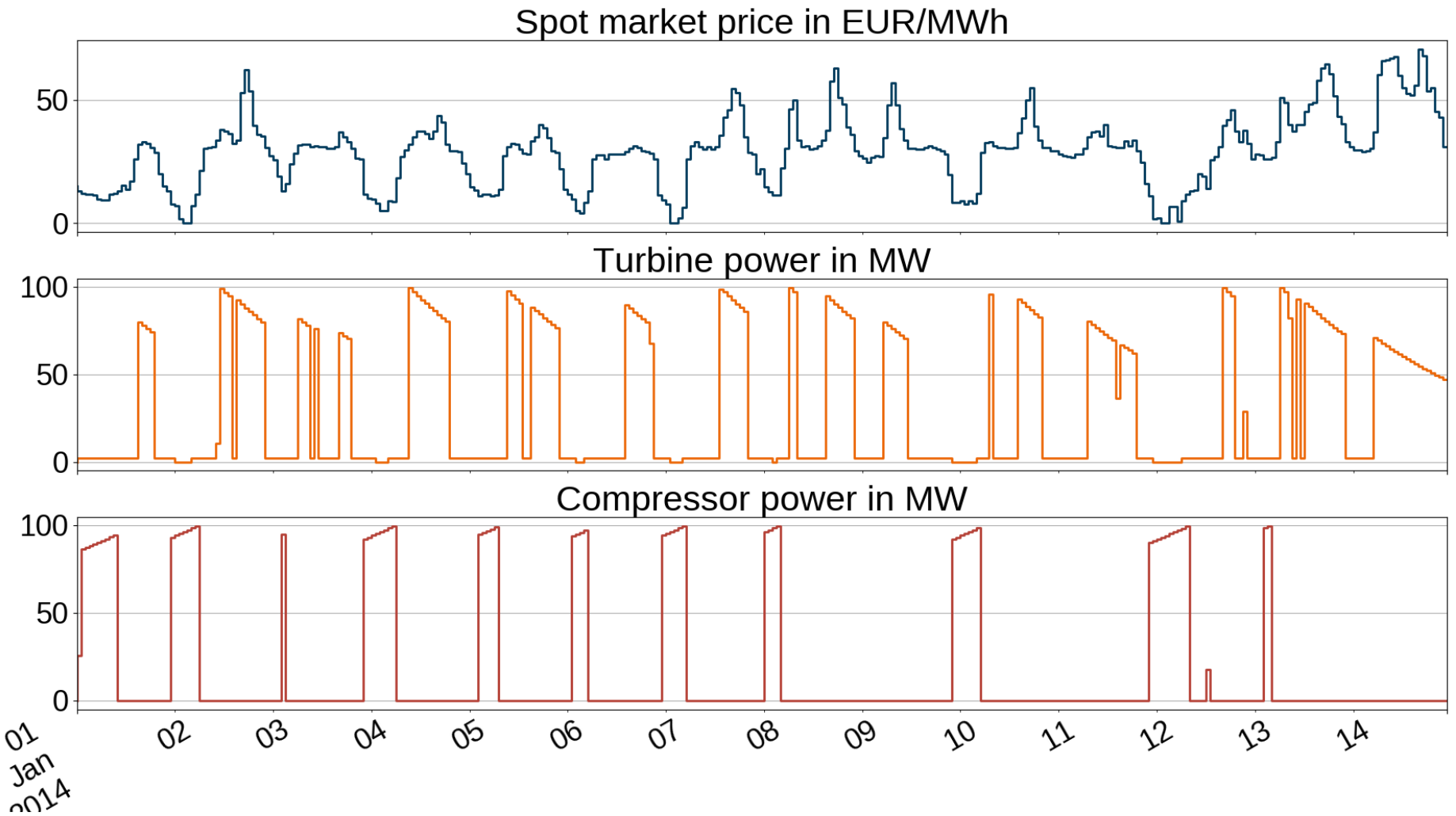
Import and usage in main.py

```python
# -*- coding: utf-8 -*-

import os
import logging
import pandas as pd
import caes

from datetime import datetime
from oemof.tools import logger
from oemof.solph import OperationalModel, EnergySystem
from oemof.solph import NodesFromCSV
from oemof.outputlib import ResultsDataFrame
```

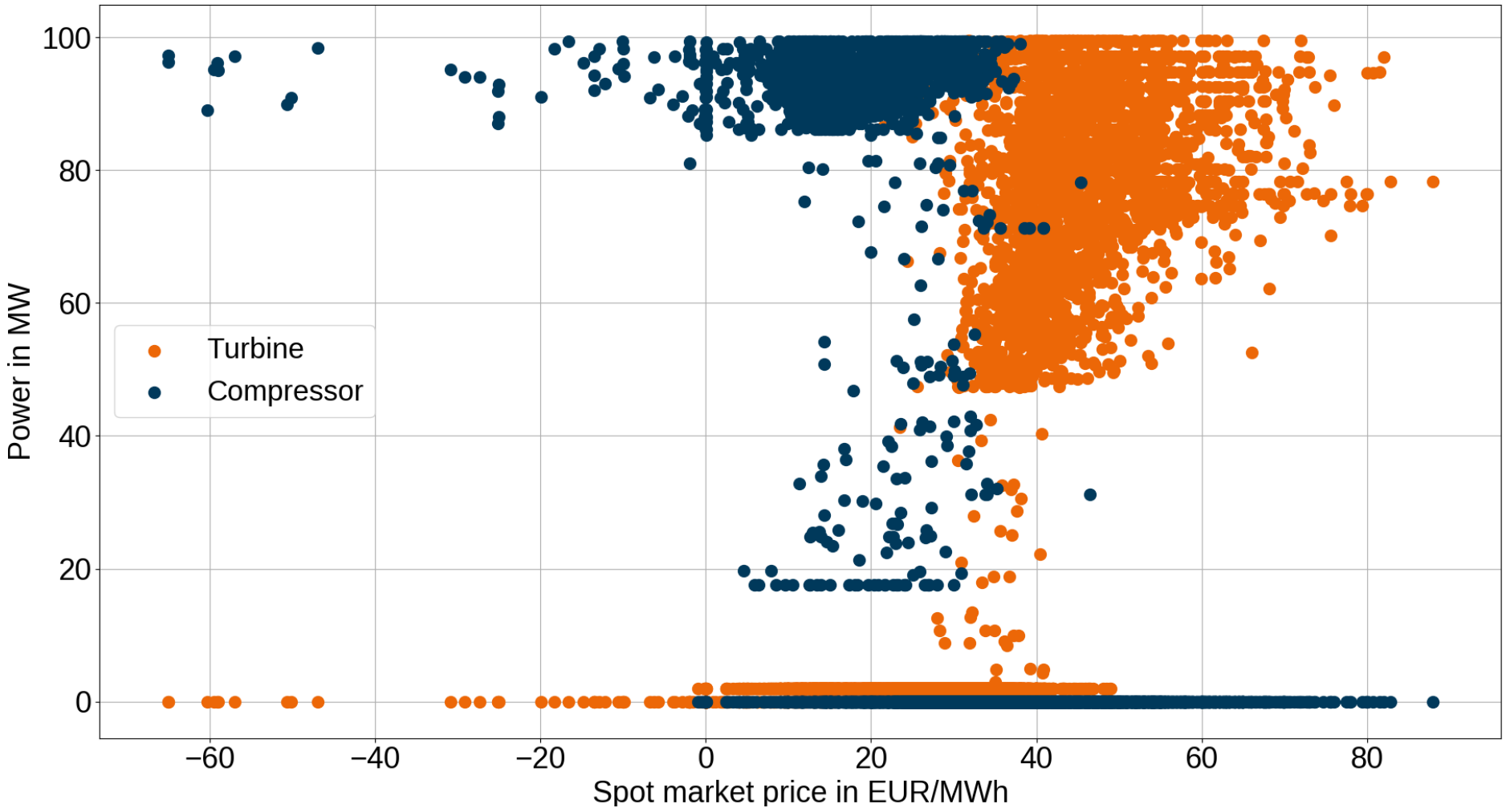
```python
    logger.define_logging()

    datetime_index = pd.date_range(date_from, date_to, freq='60min')

    es = EnergySystem(timeindex=datetime_index)

    NodesFromCSV(file_nodes_flows=os.path.join(scenario_path, nodes_flows),
                 file_nodes_flows_sequences=os.path.join(
                     scenario_path, nodes_flows_sequences), delimiter=',')

    # create connections for CAES plant
    caes.connections(es, electrical_balance='electrical_balance',
                     gas_balance='gas_balance')

    om = OperationalModel(es)

    # add model for CAES plant
    om = caes.model(es, om, electrical_balance='electrical_balance',
                    gas_balance='gas_balance')

    om.solve(solver='gurobi', solve_kwargs={'tee': True})
```



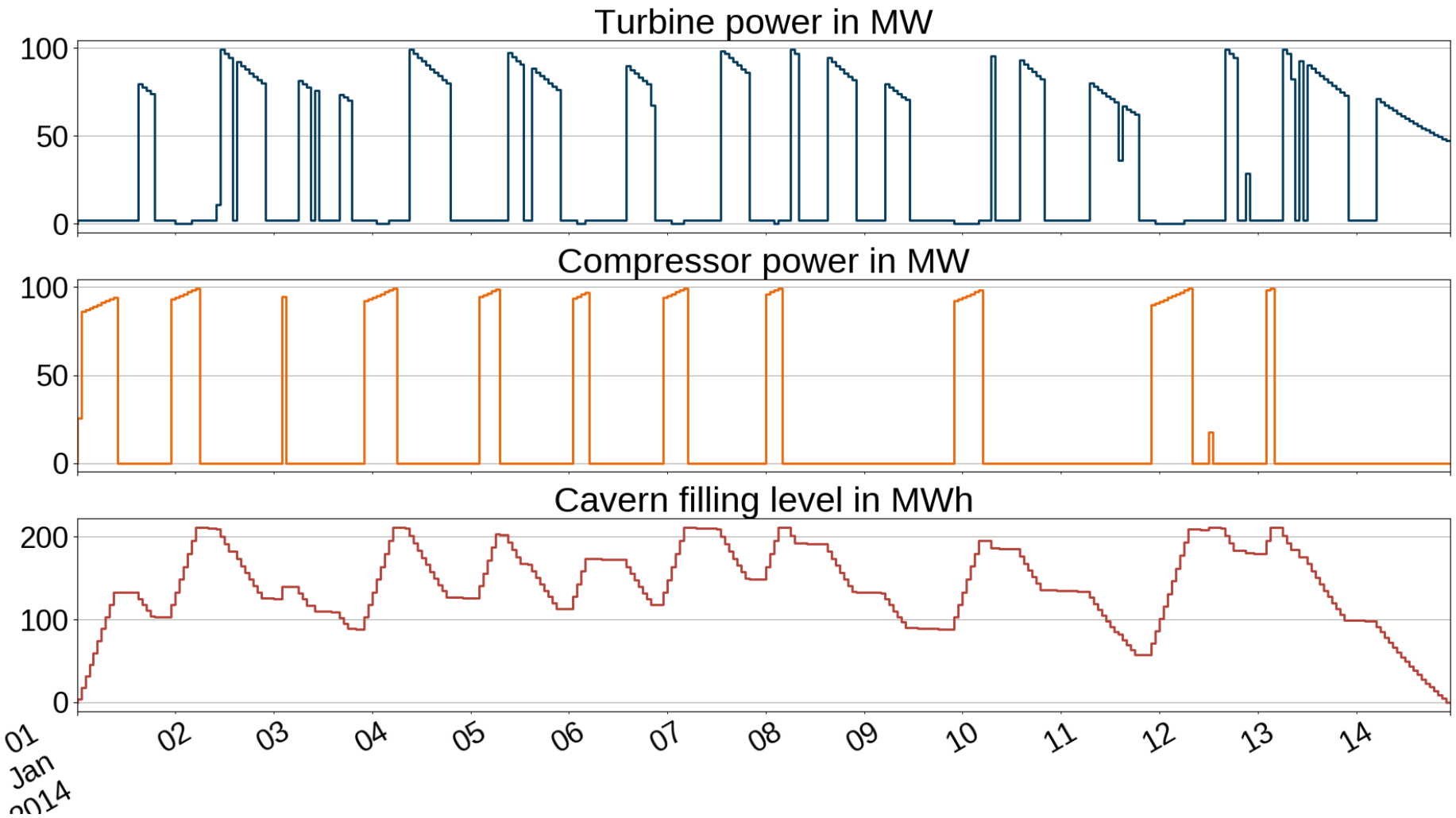
Validation


Spot market price in EUR/MWh
50
0
Turbine power in MW
100
50
0
Compressor power in MW
100
50
0
01
Jan
2014
02
03
04
05
06
07
08
09
10
11
12
13
14

Validation


Power in MW
100
80
60
40
20
0
Turbine
Compressor
−60
−40
−20
0
20
40
60
80
Spot market price in EUR/MWh

Validation


Turbine power in MW
100
50
0
Compressor power in MW
100
50
0
Cavern filling level in MWh
200
100
0
01
Jan
2014
02
03
04
05
06
07
08
09
10
11
12
13
14

Conclusion

- Specific components can be implemented in various ways
- Single modules on application level are only one possibility
  - Connections to busses as basic components (Sink, Source)
  - Internal variables mapped via „linking equalities“
  - Pyomo model uses sets of OperationalModel and can be created straightforward
  - Results have to be processed individually
- Component modules can be used within arbitrary systems
- Adding a base class allows for usage of multiple instances
- The process of adding specific components is planned to be simplified within the next releases

Questions?

Cord Kaldemeyer, M.Eng
Research Assistant, ZNES Flensburg
PhD Student, Europa-Universität Flensburg
cord.kaldemeyer@hs-flensburg.de
+49 (0)461 1260

Official website and contact to all developers:
http://www.oemof.org/contact

github repositories:
http://github.com/oemof

Sources

[1] Mollenhauer, E.; Christidis, A. & Tsatsaronis, G. Evaluation of an energy- and exergy-based generic modeling approach of combined heat and power plants International Journal of Energy and Environmental Engineering, 2016, 7, 167-176

[2] Christidis, A.; Koch, C.; Pottel, L. & Tsatsaronis, G. The contribution of heat storage to the profitable operation of combined heat and power plants in liberalized electricity markets, Energy, Volume 41, Issue 1, 2012, 75-82

[3] Wolf, D., Methods for design and application of adiabatic compressed air energy storage based on dynamic modeling, Dissertation, Fraunhofer UMSICHT, 2011

[4] Boysen, C., Grotlüschen, H., Großer, H., Kaldemeyer, C., Tuschy, I., Druckluftspeicherkraftwerk Schleswig-Holstein - Untersuchung zur Eignung Schleswig-Holsteins als Modellstandort für die Energiewende., ZNES-Forschungsergebnisse 5, 2017, ISSN: 2195-4925

[5] Kaldemeyer, C., Boysen, C., Tuschy, I., Compressed Air Energy Storage in the German Energy System – Status Quo & Perspectives, Energy Procedia, Volume 99, November 2016, Pages 298-313, ISSN 1876-6102