Practical Optimization of Energy Networks

Jagruti Thakur, Mohab Salem, Carolina, Fillipe

oemof.solph

s o l p h ['sɒlv]

open energy modelling framework

oemof.solph
Linear optimization library for energy systems

feedinlib
time series of pv or wind power plants

windpowerlib
Time series of wind plants and farms

demandlib
Create demand profiles

DHNx
District heating network optimization

oemof.solph

solph ['sɒlv]

open energy modelling framework

oemof.solph
Linear optimization library for energy systems

DHNx

feedinlib
time series of pv or wind power plants

windpowerlib
Time series of wind plants and farms

demandlib
Create demand profiles

DHNx
District heating network optimization

DHNx

Network design / investment: *Which pipes should be built? What diameters (sizes) should they have?*

- Cost optimization: *What is the least-cost configuration (investment + operation) of the network?*
- Thermal loss analysis: *How much heat is lost along the pipes, and how does this affect efficiency and cost?*
- Flow optimization: *How much heat should flow through each pipe to satisfy all consumer demands efficiently?*
- Scenario testing: *What happens to the system cost and performance if demand increases or supply temperature decreases?*

Example

Investigate how to design a cost-efficient district heating network that connects multiple heat producers to various heat consumers. The goal is to determine which pipelines should be built and at what capacities to minimize the total system cost, while ensuring all consumer heat demands are satisfied.

Input Data: The input data is provided in two folders:

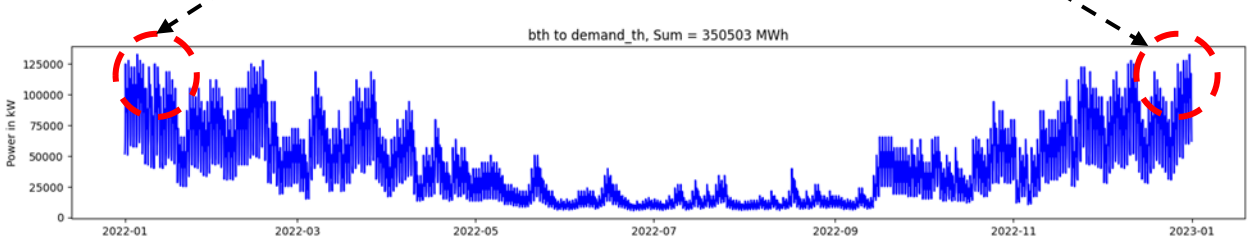
- *twn_data* includes:
  - *Locations (longitude, latitude) of producers, consumers, and forks.
  - *Possible connections (edges) between nodes.
  - *Technical parameters for each possible connection (length, losses, and maximum capacity).
- *invest_data* includes:
  - *Cost of installing different pipe types (euros/m).
  - *Operational costs.
  - *Thermal loss factors.

You must determine the optimal network configuration by solving an investment optimization problem that minimizes the total system cost while satisfying all technical and physical constraints.

Questions:

- What is the optimal configuration of the district heating network (which pipes are selected)?
- What are the capacities, directions, and thermal losses for each active pipe?
- What is the minimum total system cost (objective value) obtained from the optimization?
- How does the optimized network layout differ from the initial full network?

Peak demand




```
tree
├── pipes.csv                    # (required)
├── consumers.csv                # (required)
├── forks.csv                    # (required)
├── producers.csv                # (required)
└── sequences                    # (optional)
    └── consumers-heat_flow.csv
```

Relative thermal loss per length unit

Absolute thermal loss per length unit (e.g. [kW/m])

Pipes

- Id
- From node
- To node
- Length

Consumers

- id
- Active
- Lat
- Lon
- P_heat_max
- existing

Producers

- id
- active
- Lat
- Lon

Forks

- id
- Lat
- Lon

Sequences

- Consumer heat flow

Setting nonconvex=0 simplifies the model by ignoring fixed pipeline costs, making optimization faster but less realistic; nonconvex=1 is recommended for accurate DHS cost modeling.

Maximum installable capacity (e.g. [kW]).

Pipes

label_3	active	non-convex	l_factor	l_factor_fix	cap_max	cap_min	capex_pipes	fix_costs
pipe-typ-A	1	0	0	0	100000	0	0.5	0

Example for table of *Buses*

Consumers

label_2	active	excess	shortage	shortage costs	excess costs
heat	1	0	0	99999	99999

demand.csv

label_2	active	nominal_value
heat	1	1

```
tree
├── network
│   └── pipes.csv            # (required)
│
├── consumers
│   ├── bus.csv              # (required)
│   ├── demand.csv           # (required)
│   ├── source.csv           # (optional)
│   ├── storages.csv         # (optional)
│   └── transformer.csv      # (optional)
│
└── producers
    ├── bus.csv              # (required)
    ├── demand.csv           # (optional)
    ├── source.csv           # (required)
    ├── storages.csv         # (optional)
    └── transformer.csv      # (optional)
```

Producers

Example for table of *Buses*

label_2	active	excess	shortage	shortage costs	excess costs
heat	1	0	0	99999	99999

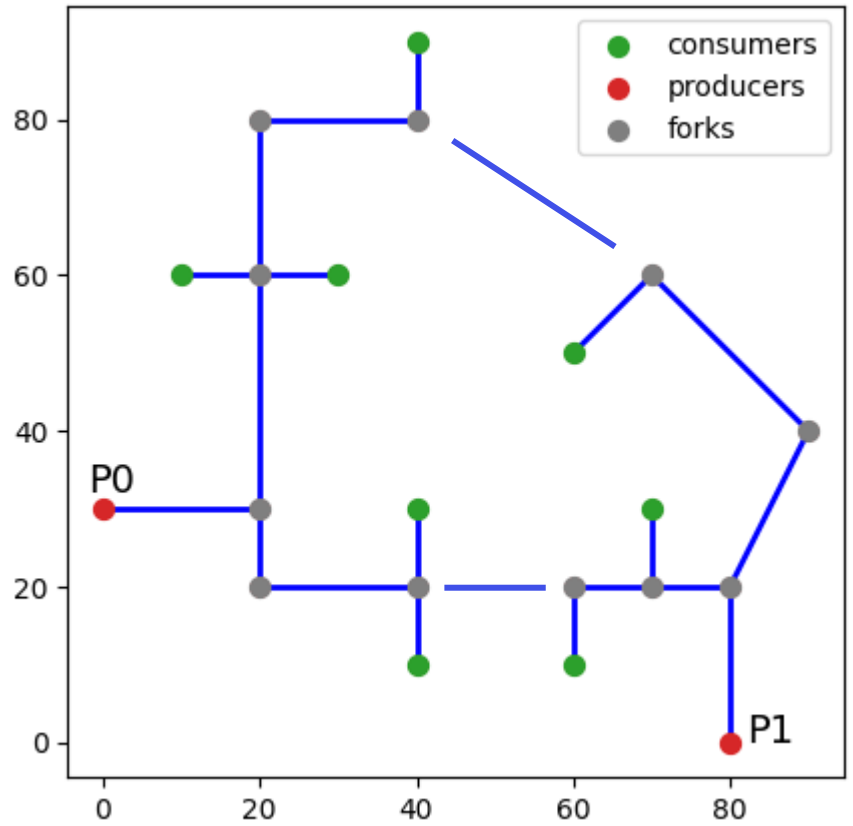
source.csv

label_2	active
heat	1

```python
import matplotlib.pyplot as plt
import dhnx
import os
from pyomo.environ import SolverFactory

# Initialize thermal network
network = dhnx.network.ThermalNetwork()
network = network.from_csv_folder(r"E:\Jagruti\Mohab\twn_data")

# Load investment parameter
invest_opt = dhnx.input_output.load_invest_options(r"E:\Jagruti\Mohab\invest_data")
network.optimize_investment(invest_options=invest_opt,  solver='glpk')

# ####### Postprocessing and Plotting ###########
# get results
results_edges = network.results.optimization['components']['pipes']
print(results_edges[['from_node', 'to_node', 'hp_type', 'capacity',
                     'direction', 'costs', 'losses']])

# print(results_edges[['invest_costs[€]']].sum())
print('Objective value: ', network.results.optimization['oemof_meta']['objective'])

# assign new ThermalNetwork with invested pipes
twn_results = network
twn_results.components['pipes'] = results_edges[results_edges['capacity'] > 0.001]

# plot invested edges
static_map_2 = dhnx.plot.static_map(twn_results, node_color='tab:red')
static_map_2 = dhnx.plot.static_map(twn_results, node_color='tab:red')
plt.title('Result network')
plt.scatter(network.components.consumers['lon'], network.components.consumers['lat'],
            color='tab:green', label='consumers', zorder=2.5, s=50)
plt.scatter(network.components.producers['lon'], network.components.producers['lat'],
            color='tab:red', label='producers', zorder=2.5, s=50)
plt.scatter(network.components.forks['lon'], network.components.forks['lat'],
            color='tab:grey', label='forks', zorder=2.5, s=50)
# plt.text(-2, -32, 'P0', fontsize=14)
# plt.text(82, 0, 'P1', fontsize=14)
plt.legend()
plt.show()
```

Variable investment costs depending on the installed heat transport capacity (e.g. [€/kW]).

Minimum installable capacity (e.g. [kW]).




```python
import matplotlib.pyplot as plt
import dhnx
import os
from pyomo.environ import SolverFactory

# Initialize thermal network
network = dhnx.network.ThermalNetwork()
network = network.from_csv_folder(r"E:\Jagruti\Mohab\twn_data")

# Load investment parameter
invest_opt = dhnx.input_output.load_invest_options(r"E:\Jagruti\Mohab\invest_data")
network.optimize_investment(invest_options=invest_opt,  solver='glpk')
```

```
1 network.results.optimization.keys()
```

```
dict_keys(['oemof', 'oemof_meta', 'components'])
```

```python
# ####### Postprocessing and Plotting ###########
# get results
results_edges = network.results.optimization['components']['pipes']
print(results_edges[['from_node', 'to_node', 'hp_type', 'capacity',
                     'direction', 'costs', 'losses']])

# print(results_edges[['invest_costs[€]']].sum())
print('Objective value: ', network.results.optimization['oemof_meta']['objective'])

# assign new ThermalNetwork with invested pipes
twn_results = network
twn_results.components['pipes'] = results_edges[results_edges['capacity'] > 0.001]

# plot invested edges
static_map_2 = dhnx.plotting.StaticMap(twn_results)
static_map_2.draw(background_map=False)
plt.title('Result network')
plt.scatter(network.components.consumers['lon'], network.components.consumers['lat'],
            color='tab:green', label='consumers', zorder=2.5, s=50)
plt.scatter(network.components.producers['lon'], network.components.producers['lat'],
            color='tab:red', label='producers', zorder=2.5, s=50)
plt.scatter(network.components.forks['lon'], network.components.forks['lat'],
            color='tab:grey', label='forks', zorder=2.5, s=50)
# plt.text(-2, 32, 'P0', fontsize=14)
# plt.text(82, 0, 'P1', fontsize=14)
plt.legend()
plt.show()
```

Example

In this task, you will design, optimize, and analyze a small district heating network for a town. Your goal is to determine an investment-optimal configuration of pipes that satisfies all consumer heat demands while minimizing total system costs.

Steps to follow:

Initialize the network

Create a new thermal network object to represent your system.
This network will include all consumers, producers, and potential pipe connections.

Load town and component data

Import the parameters for the town, including the locations of consumers and producers, heat demands, and other relevant data.
Import investment-related parameters such as pipe types, costs, capacities, and efficiencies.

Analyze the results

Examine the selected pipes, their capacities, and associated costs.
Identify which pipes were actually invested in (capacity > 0).
Check the overall objective value to ensure the solution is reasonable.

Visualize the network

Plot the network before and after optimization to compare the initial design with the optimized solution.
Highlight the locations of consumers, producers, and network forks to help interpret the network structure.

Discussion and reflection

Why did the optimizer choose certain pipe types or locations?
How do the investment decisions relate to cost, thermal losses, and capacities?

Exercise

In this exercise, you will work with a district heating network optimization model using the District Heating Network Optimization (DHNx) library. The goal is to determine the most cost-effective pipe diameters (DN numbers) to install in a thermal network that connects heat producers and consumers.

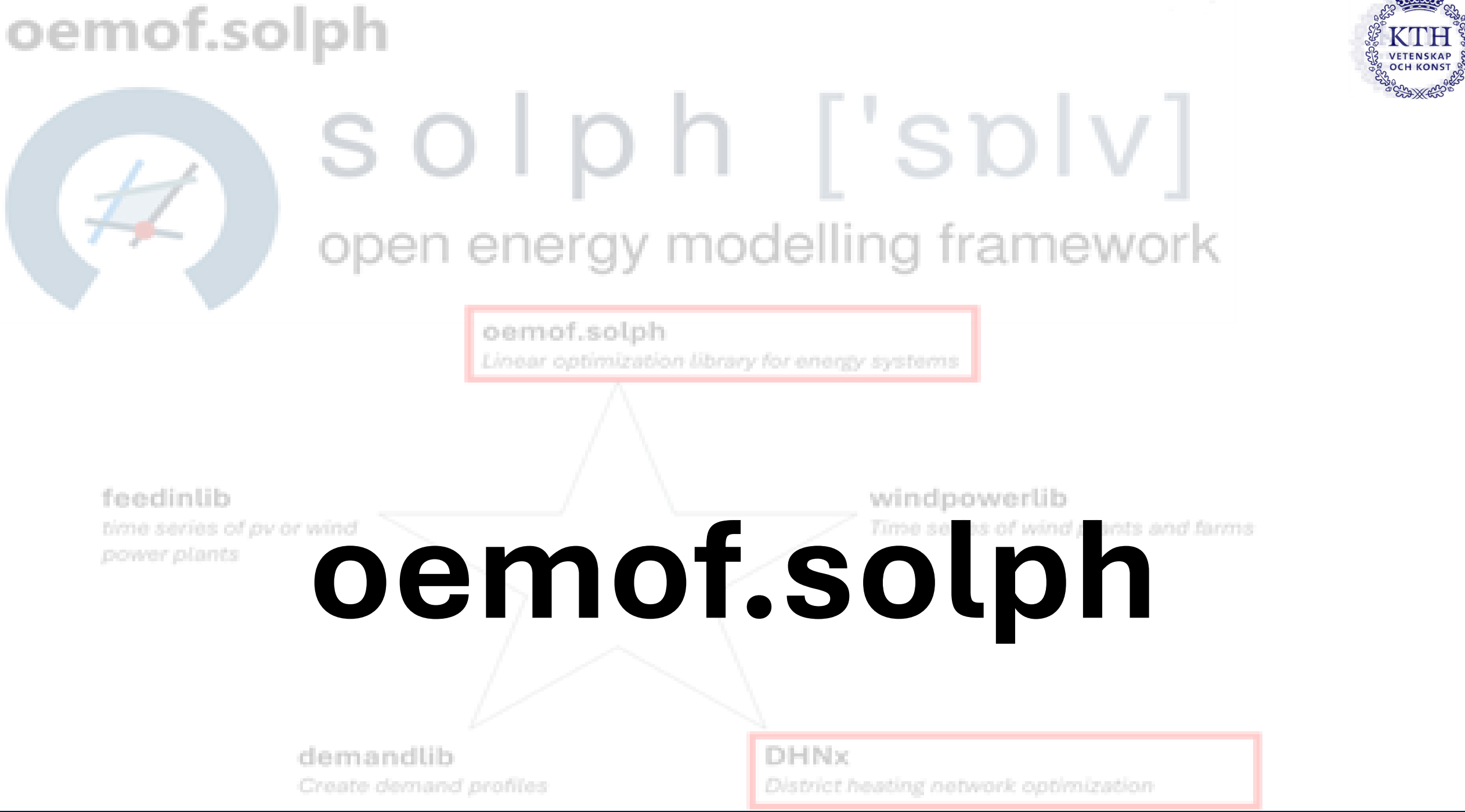
- Load and visualize the network.
- Perform an investment optimization to determine the best pipe sizes.
- Analyze the resulting network and observe which DN numbers were chosen.
- Plot both the given and optimized networks for comparison.

Pipes	Absolute thermal losses	Fixed costs	Maximum capacity	Minimum capacity
DN-25	0.0076	466	28	27
DN-32	0.0086	491	54	53
DN-40	0.0095	522	98	97
DN-50	0.0091	563	179	178
DN-63	0.011	620	335	334

Exercise

In a district heating network, some pipes are already installed, but the city wants to expand or upgrade the network to meet growing demand or improve efficiency. Building new pipes costs money, so you need to decide which pipes to invest in and which existing ones to keep.

- Which pipes should be built or expanded to meet the heat demand efficiently.
- What capacity each pipe should have to transport the required heat.
- How costs and heat losses are distributed across the network.
- How the existing infrastructure affects new investment decisions, i.e., which existing pipes can be used and which new ones are necessary.


oemof.solph
solph ['sɒlv]
open energy modelling framework
oemof.solph
Linear optimization library for energy systems
feedinlib
time series of pv or wind power plants
windpowerlib
demandlib
Create demand profiles
DHNx
District heating network optimization

oemof.solph

- ***OEMOF Answers:***

- Which technologies should operate at what time?

Should electricity come from solar, wind, or gas at 2 PM?
Should heat be provided from CHP or Geothermal plant at 1 PM?

- How much energy should flow between components?

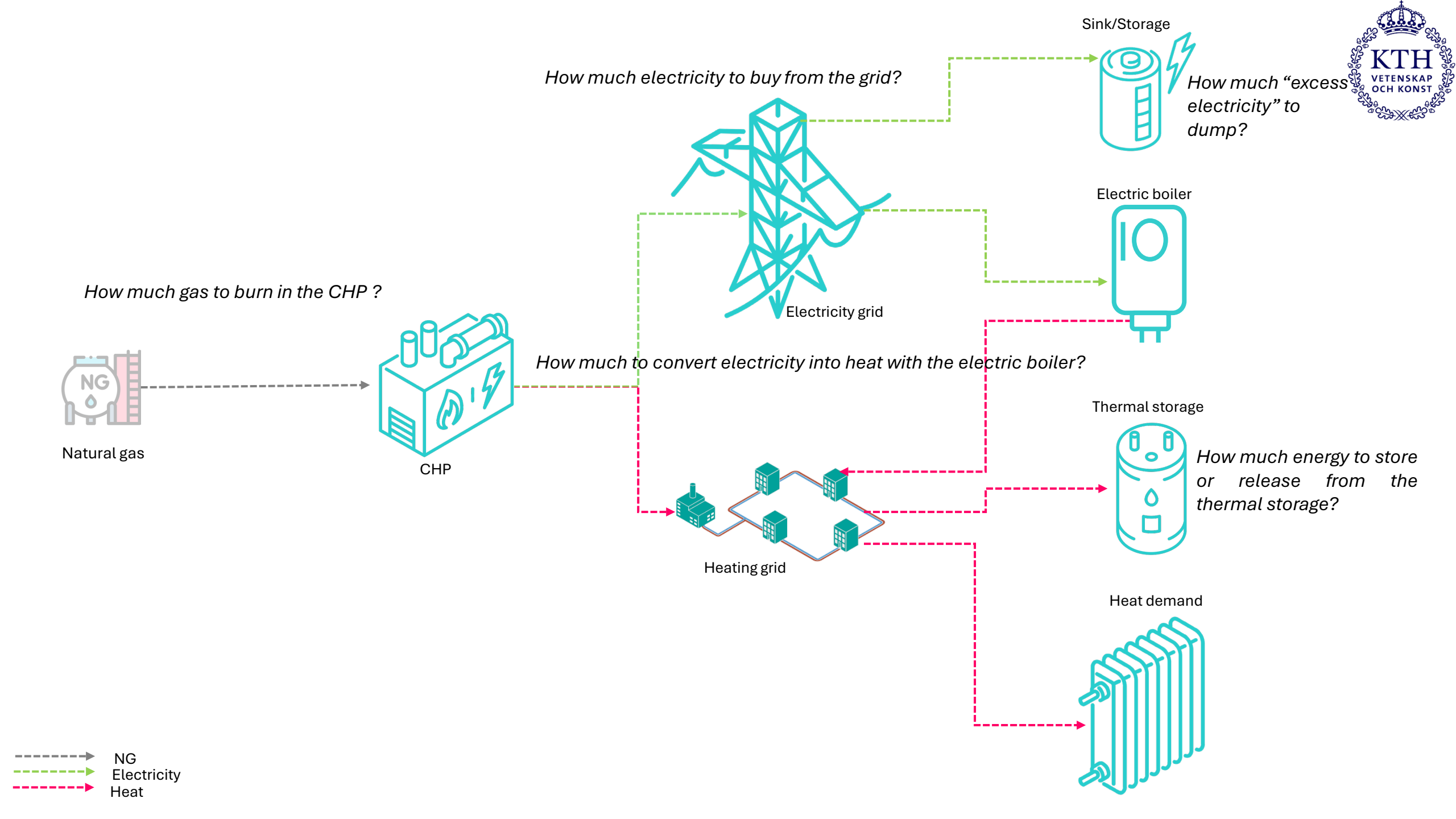
How much heat should go from the boiler to the storage tank or to the district heating network?

- What is the optimal system configuration?
- *Should we invest in a heat pump, more storage, or solar collectors?*

- What are the total costs and emissions of the system?

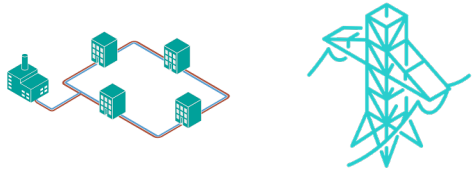
- How does the system behave under different scenarios?

What if fuel prices rise? or What if we increase renewable energy targets?

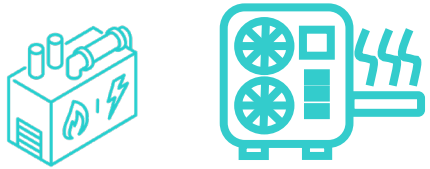

Sink/Storage
How much electricity to buy from the grid?
How much "excess electricity" to dump?
Electric boiler
How much gas to burn in the CHP ?
Electricity grid
NG
Natural gas
CHP
How much to convert electricity into heat with the electric boiler?
Thermal storage
How much energy to store or release from the thermal storage?
Heating grid
Heat demand
NG
Electricity
Heat

oemof.solph

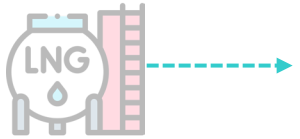
Buses : *represent commodities such as electricity, heat or natural gas. The main aim of buses to balance input and output flow at any given point in time.*



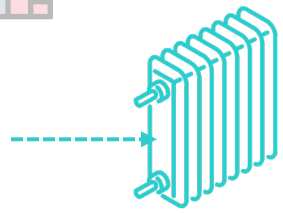
Converters : *are like heat pumps, combined heat and power plants or electrolyzers.*



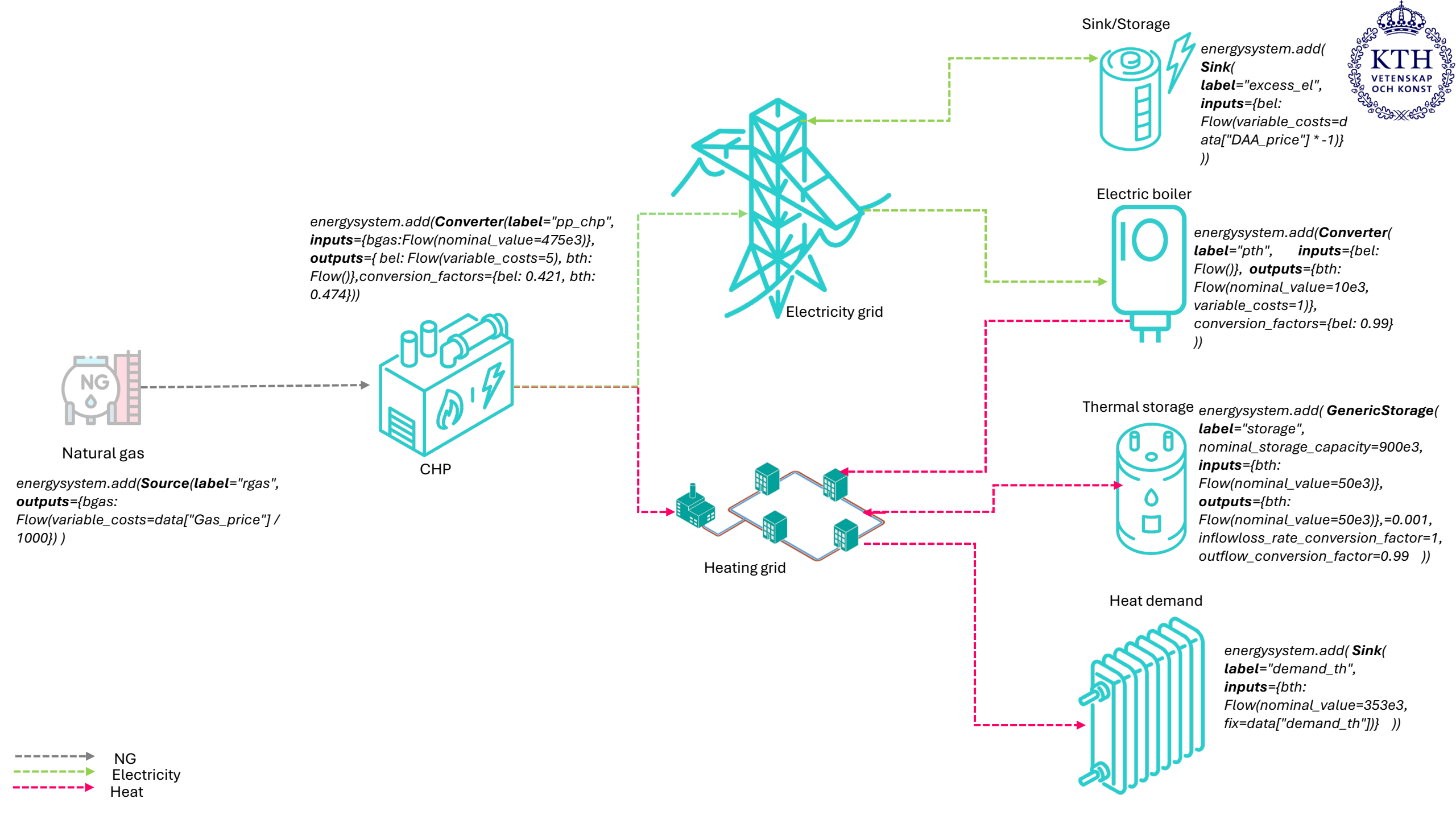
Sources : *Does not require any inputs, for example: PV panels or electricity imports from the grid.*

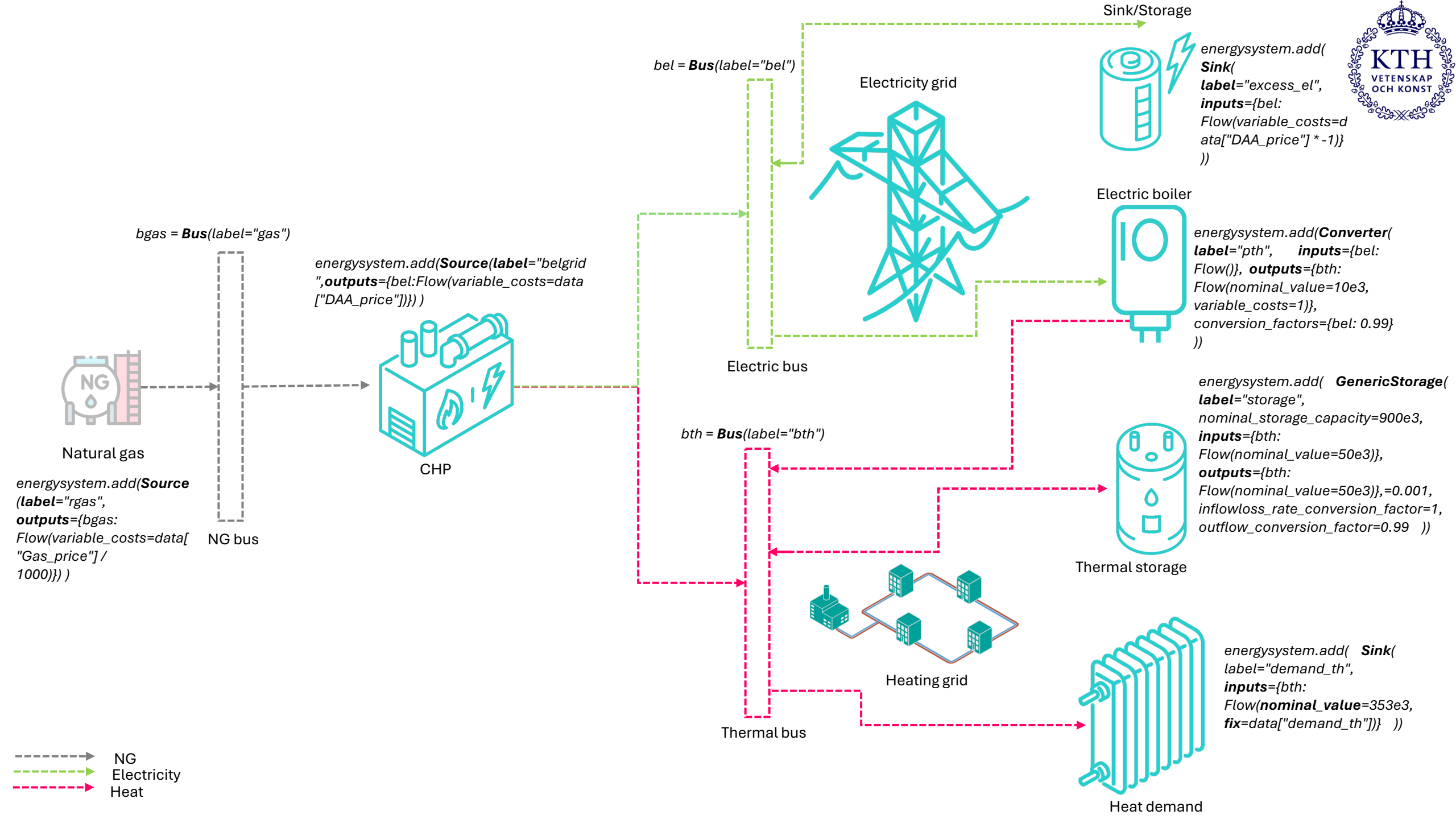


Sinks : *Does not generating any output, for example: Heat or electricity demand of consumers.*



Energy storage : *Such as batteries or heat storage tanks.*


Sink/Storage
energysystem.add(Sink(label="excess_el", inputs={bel: Flow(variable_costs=data["DAA_price"] * -1)}))
Electric boiler
energysystem.add(Converter(label="pth", inputs={bel: Flow()}, outputs={bth: Flow(nominal_value=10e3, variable_costs=1)}, conversion_factors={bel: 0.99}))
Thermal storage
energysystem.add(GenericStorage(label="storage", nominal_storage_capacity=900e3, inputs={bth: Flow(nominal_value=50e3)}, outputs={bth: Flow(nominal_value=50e3)},=0.001, inflowloss_rate_conversion_factor=1, outflow_conversion_factor=0.99))
Heat demand
energysystem.add(Sink(label="demand_th", inputs={bth: Flow(nominal_value=353e3, fix=data["demand_th"])}))
Electricity grid
energysystem.add(Converter(label="pp_chp", inputs={bgas:Flow(nominal_value=475e3)}, outputs={ bel: Flow(variable_costs=5), bth: Flow()},conversion_factors={bel: 0.421, bth: 0.474}))
NG
Natural gas
CHP
Heating grid
energysystem.add(Source(label="rgas", outputs={bgas: Flow(variable_costs=data["Gas_price"] / 1000}))
NG
Electricity
Heat


Sink/Storage
energysystem.add(
Sink(
label="excess_el",
inputs={bel:
Flow(variable_costs=d
ata["DAA_price"] * -1)}
))
bel = Bus(label="bel")
Electricity grid
Electric boiler
energysystem.add(Converter(
label="pth", inputs={bel:
Flow()}, outputs={bth:
Flow(nominal_value=10e3,
variable_costs=1)},
conversion_factors={bel: 0.99}
))
Electric bus
bgas = Bus(label="gas")
energysystem.add(Source(label="belgrid
",outputs={bel:Flow(variable_costs=data
["DAA_price"])}))
NG
Natural gas
energysystem.add(Source
(label="rgas",
outputs={bgas:
Flow(variable_costs=data[
"Gas_price"] /
1000)}))
NG bus
CHP
energysystem.add(GenericStorage(
label="storage",
nominal_storage_capacity=900e3,
inputs={bth:
Flow(nominal_value=50e3)},
outputs={bth:
Flow(nominal_value=50e3)},=0.001,
inflowloss_rate_conversion_factor=1,
outflow_conversion_factor=0.99))
bth = Bus(label="bth")
Thermal storage
Heating grid
Thermal bus
energysystem.add(Sink(
label="demand_th",
inputs={bth:
Flow(nominal_value=353e3,
fix=data["demand_th"])}))
Heat demand
NG
Electricity
Heat

The code

```
import os
import warnings
import logging
import pandas as pd
from oemof.solph import (Bus, EnergySystem, Flow, Model, create_time_index, processing)
from oemof.solph.components import (Sink, Source, Converter, GenericStorage)
# Read input data
filename = os.path.join(os.getcwd(), r"E:\Jagruti\input_data.csv")
try:
    data = pd.read_csv(filename)
except FileNotFoundError:
    warnings.warn(f"Data file not found: {filename}. Using fallback data!", UserWarning)
    data = pd.DataFrame({
        "DAA_price": [500, 400],
        "demand_th": [0.1, 0.3],
        "CO2_price": [300, 200],
        "Gas_price": [70, 80]
    })

solver = "glpk"
solver_verbose = False

# Create energy system
datetimeindex = create_[illegible]ata)) [illegible]default, the function creates an hourly index for one year.
energysystem = EnergySy[illegible]etimeindex, infer[illegible]_interval=False)

# Buses
bel = Bus(label="bel") # Electricit[illegible]
bth = Bus(label="bth") # Heat bus
bgas = Bus(label="gas") # Natural g[illegible]
energysystem.add(bgas, bel, bth)
```

```
# Excess electricity sink
energysystem.add(
    Sink(
        label="excess_el",
        inputs={bel: Flow(variable_[illegible]=data["DAA_price"][illegible])}
    )
)

# Gas source with cost from gas + CO2
energysystem.add(
    Source(label="rgas",outputs={bgas: Flow(variable_costs=data["Gas_price"] / 1000 + data["CO2_price"] * 0.000202)})
            )

# Grid electricity source
energysystem.add(
    Source(label="belgrid",outputs={bel: Flow(variable_costs=data["DAA_price"])})
            )

# Thermal demand sink
energysystem.add(
    Sink(
        label="demand_th",
        inputs={bth: Flow(
            nominal_value=353e3,
            fix=data["demand_th"]
        )},
    )
)
```

```
# Power-to-heat (electric boiler)
energysystem.add(
    Converter(
        label="pth",
        inputs={bel: Flow()},
        outputs={bth: Flow(nominal_value=10e3, variable_costs=1)},
        conversion_factors={bel: 0.99}
    )
)

# Combined heat and power plant (CHP)
energysystem.add(
    Converter(
        label="pp_chp",
        inputs={bgas: Flow(nominal_value=475e3)},
        outputs={
            bel: Flow(variable_costs=5),
            bth: Flow()
        },
        conversion_factors={bel: 0.421, bth: 0.474}
    )
```

```
# Thermal storage
energysystem.add(
    Gene[illegible]torage(
        [illegible]l="stora[illegible]
        [illegible]inal_storag[illegible]
        inputs={bth: Flow([illegible]ue=50e3)},
        outputs={bth: Flow(nominal_value=50e3)},
        loss_rate=0.001,
        inflow_conversion_factor=1,
        outflow_conversion_factor=0.99
    )
)

# Build and solve model
model = Model(energysystem)
logging.info("Solving the optimization problem.")
model.solve(solver=solver, solve_kwargs={"tee": solver_verbose})

# Process results
energysystem.results["main"] = processing.results(model)
energysystem.results["meta"] = processing.meta_results(model)

# Save results to file
output_file = os.path.join(os.getcwd(), "results.oemof")
energysystem.dump(os.getcwd(), "results.oemof")

logging.info("Results have been dumped.")
```

The code

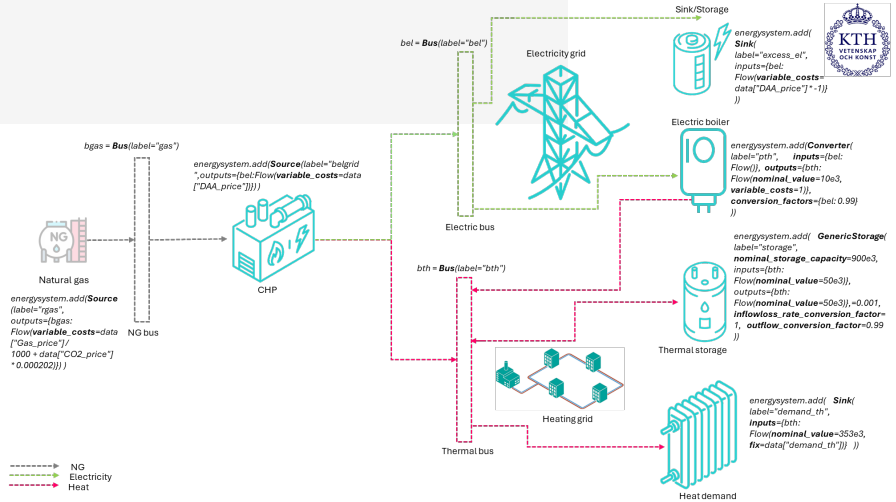
Reading the file

```
filename = r"E:\Jagruti\input_data.csv"
data = pd.read_csv(filename)

solver = "cbc"
solver_verbose = False

# Create energy system
datetimeindex = create_time_index(2022, number=len(data)) # By default, the function creates an hourly index for one year.
energysystem = EnergySystem(timeindex=datetimeindex, infer_last_interval=False)

# Buses
electrical_bus = Bus(label="electrical_bus") # Electricity bus
thermal_bus = Bus(label="thermal_bus") # Heat bus
gas_bus = Bus(label="gas_bus") # Natural gas bus
energysystem.add(electrical_bus, thermal_bus, gas_bus)

```

Assigning the solver

Instantiate the time index and create energy system model

Create buses



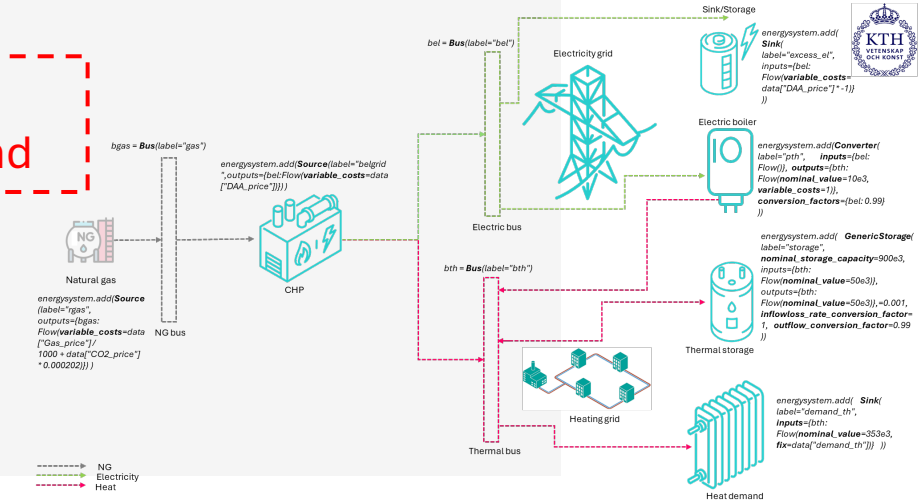
```python
# Excess electricity sink
energysystem.add(
    Sink(
        label="excess_electricity",
        inputs={electrical_bus: Flow(variable_costs=data["electricity_price"] * -1)}
    )
)

# Gas source with cost from gas + CO2
energysystem.add(
    Source(label="natural_gas",outputs={gas_bus: Flow(variable_costs=data["gas_price"] / 1000)})
                )

# Grid electricity source
energysystem.add(
    Source(label="electricity_grid",outputs={electrical_bus: Flow(variable_costs=data["electricity_price"])})
                )

# Thermal demand sink
energysystem.add(
    Sink(
        label="thermal_demand",
        inputs={thermal_bus: Flow(
            nominal_value=353e3,
            fix=data["thermal_demand"]
        )},
    )
)
```

Electric sink

NG

Electricity grid

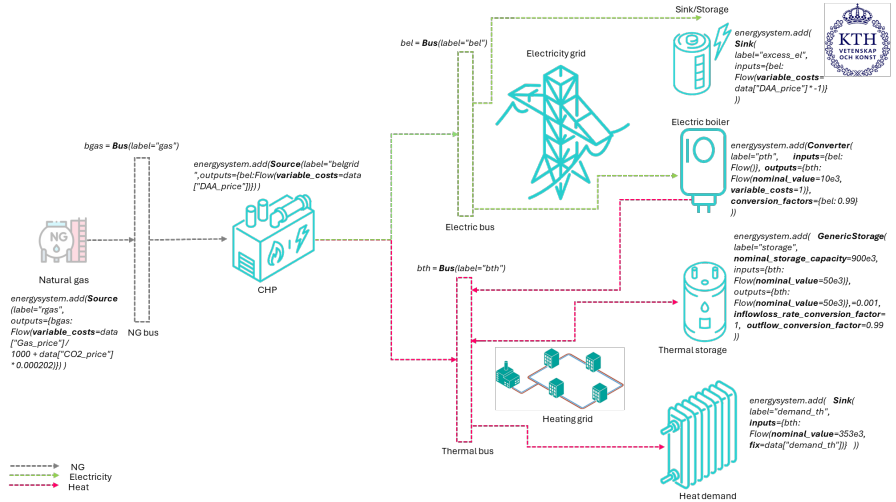
Heat demand



```python
# Power-to-heat (electric boiler)
energysystem.add(
    Converter(
        label="electric_boiler",
        inputs={thermal_bus: Flow()},
        outputs={thermal_bus: Flow(nominal_value=10e3, variable_costs=1)},
        conversion_factors={electrical_bus: 0.99}
    )
)

# Combined heat and power plant (CHP)
energysystem.add(
    Converter(
        label="chp",
        inputs={gas_bus: Flow(nominal_value=475e3)},
        outputs={
            electrical_bus: Flow(variable_costs=5),
            thermal_bus: Flow()
        },
        conversion_factors={electrical_bus: 0.421, thermal_bus: 0.474}
    )
)
```

Electric boiler

CHP



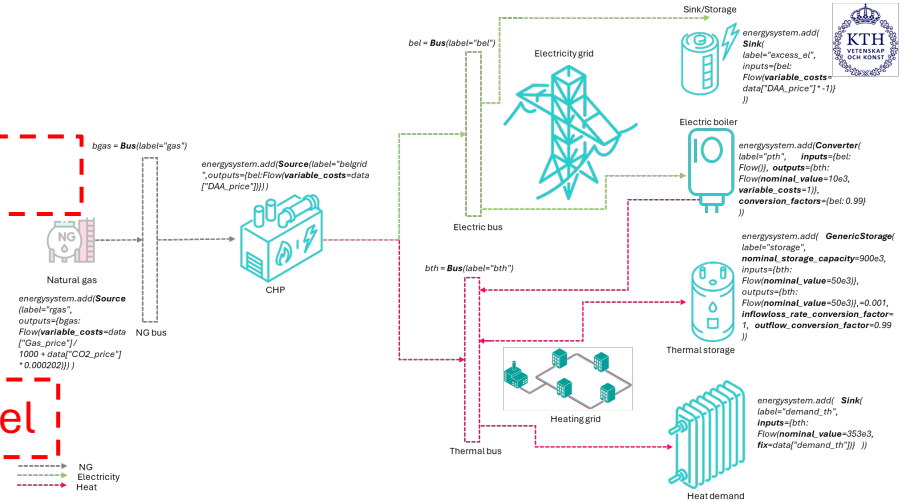
```python
# Thermal storage
energysystem.add(
    GenericStorage(
        label="storage",
        nominal_storage_capacity=900e3,
        inputs={thermal_bus: Flow(nominal_value=50e3)},
        outputs={thermal_bus: Flow(nominal_value=50e3)},
        loss_rate=0.001,
        inflow_conversion_factor=1,
        outflow_conversion_factor=0.99
    )
)

# Build and solve model
model = Model(energysystem)
logging.info("Solving the optimization problem.")
model.solve(
    solver='cbc',
    solve_kwargs={"tee": True},
    cmdline_options={"ratioGap": "0.02"}
)

# Process results
energysystem.results["main"] = processing.results(model)
energysystem.results["meta"] = processing.meta_results(model)

# Save results to file
output_file = os.path.join(os.getcwd(), "results.oemof")
energysystem.dump(os.getcwd(), "results.oemof")

logging.info("Results have been dumped.")
```

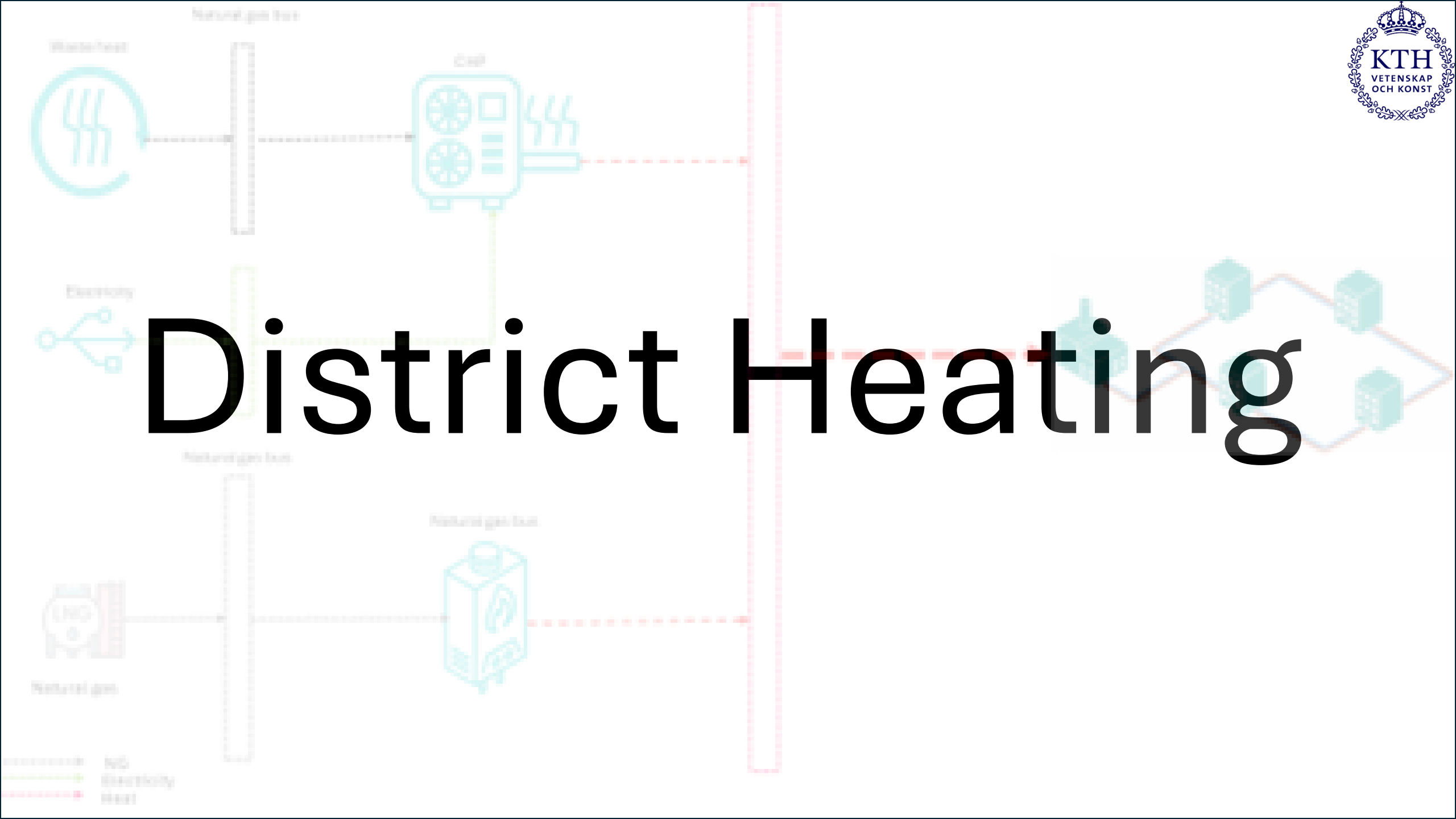
Thermal storage

Build and solve

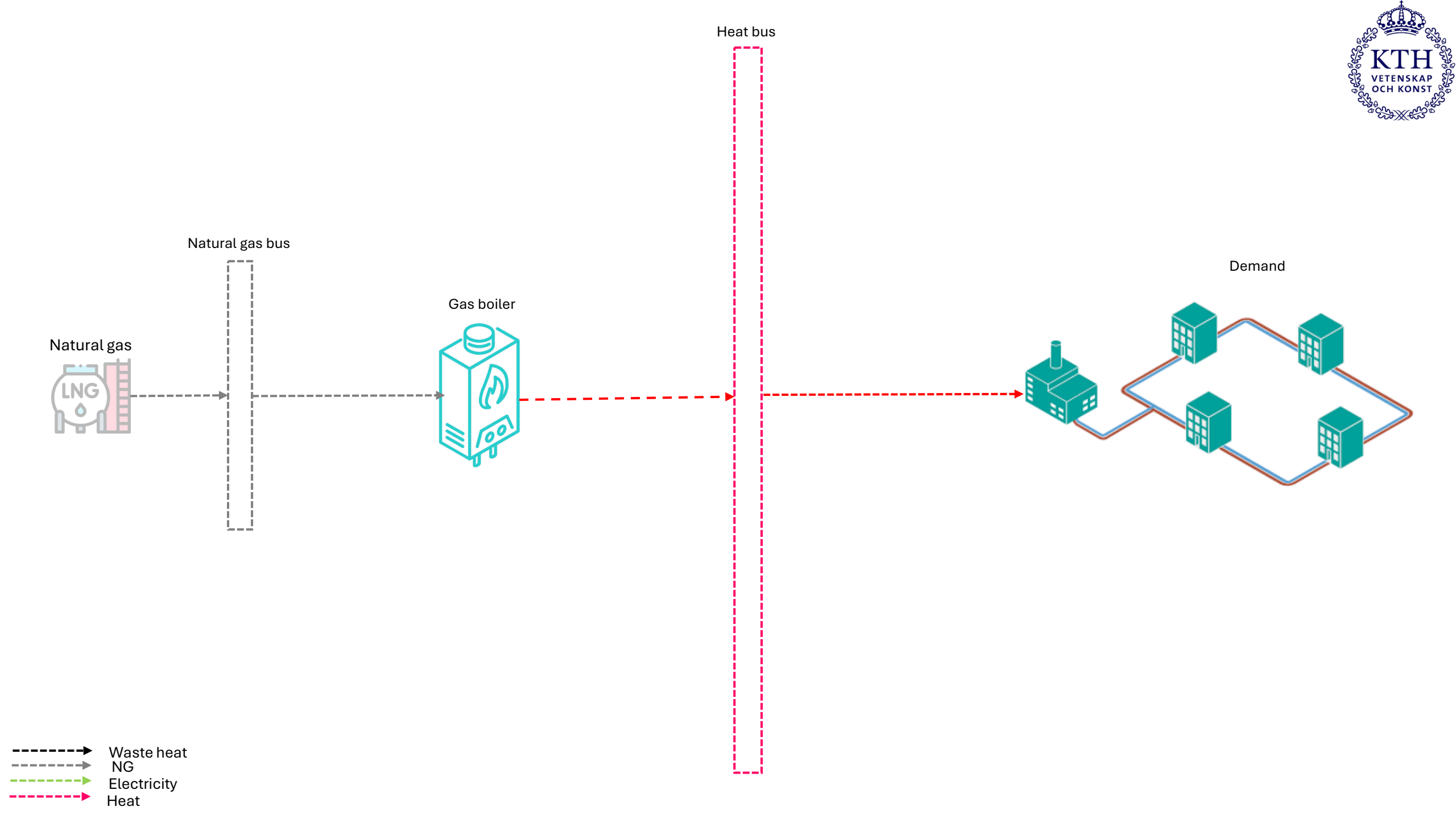
Fetch results

Save the model





District Heating


Heat bus
Natural gas bus
Gas boiler
Natural gas
Demand
LNG
Waste heat
NG
Electricity
Heat

Exercise

In this task, you will design and simulate a district heating system to satisfy the thermal demand of a community over time. The system should include multiple energy technologies, a storage unit, and account for operational costs and efficiencies.
You are provided with a time series dataset containing heat demand, electricity prices, gas prices, and optionally waste heat availability. Using this data, your task is to build an oemof energy system model that includes the following components:

Gas boiler: Converts natural gas to heat. Define its efficiency and variable operational cost.
Heat pump: Converts electricity and waste heat into thermal energy. Specify its coefficient of performance (COP) and variable cost.
Thermal storage: Stores and releases heat to match supply and demand over time. Specify storage capacity, charging/discharging limits, losses, and variable cost.
Optional: Include a solar thermal collector using a custom irradiance profile to produce additional heat.

Your tasks include:

- Define the system components: Create buses for heat, gas, electricity, and waste heat, and connect sources, sinks, converters, and storage.
- Assign input parameters: Set capacities, efficiencies, COP, and costs for all technologies.
- Run the model: Optimize the operation of the system over the entire simulation period to satisfy heat demand.
- Analyze results: Extract time series of heat production for each technology, storage charging/discharging, and total heat delivered. Calculate operational costs and evaluate whether demand is fully met.
- Visualize system behavior: Plot hourly heat production by technology and the storage content over time.

Discussion points:

- How do different technologies complement each other to meet demand?
- How does storage operation improve system flexibility and reduce peak loads?
- How do electricity and gas price variations affect the optimal dispatch of heat generation?
- Are there periods of unmet demand or surplus heat? What does this indicate about system design?
- How could capacities or operational strategies be adjusted to improve efficiency or reduce costs?

Cont.

Gas boiler:

- Operational cost: 1.10 €/MWh (or per unit energy)
- Efficiency: 95% (0.95 conversion factor from gas to heat)

Heat pump

- Coefficient of performance (COP): 3.5
- Variable operational cost: 1.2 €/MWh for heat output
- Conversion factors: Electricity input: 1 / COP (1 unit of electricity produces COP units of heat)Waste heat input: (COP - 1) / COP (proportion of heat from waste heat)

Thermal Storage

- Nominal storage capacity: 98.6 MWh
- Charging and discharging variable costs: 0.1 €/MWh
- Charging/discharging limits: Can charge or discharge full capacity in 24 hours (invest_relation_input/output_capacity = 1/24)
- Loss rate: 0.001 per hour
- Balanced operation: total charge equals total discharge over the simulation period

Exercise

In the previous task, you built and simulated a district heating system with multiple energy technologies and a thermal storage unit, focusing on operational dispatch to satisfy heat demand. In this sequel, you will introduce investment decisions for key components, allowing the model to determine optimal capacities of generation units and storage while minimizing total system costs, including both capital and operational costs.

You will use the same time series dataset (heat demand, gas prices, electricity prices, optional waste heat availability) and the same energy system structure (buses, sources, sinks, converters, storage).
Investment-enabled System Components

Gas Boiler

- Capital cost: 60,000 €/MW
- Variable cost: 1.10 €/MWh
- Efficiency: 95%
- Maximum investment capacity: 50 MW

Heat Pump

- Capital cost: 500,000 €/MW
- Variable cost: 1.2 €/MWh
- Coefficient of performance (COP): 3.5
- Minimum operation fraction: 50% of installed capacity
- Maximum investment capacity: 999 MW

Thermal Storage

- Capital cost: 1,060 €/MWh
- Variable costs: 0.1 €/MWh for both charging and discharging
- Maximum charging/discharging rate: full capacity in 24 hours
- Loss rate: 0.001 per hour

Installation

Solver

We recommend using conda for the python installation, with which you can also install a solver. If you want to install the solver externally (not via conda), you can follow these steps:

1. Download CBC or GLPK (64/32 bit)

2. Unpack CBC/GLPK to any folder (e.g. C:/Users/Somebody/my_programs)

3. Add the path of the executable files of both solvers to the PATH variable (can be done per user without administrator privileges).

4. Restart Windows

Thank you