Appendix B: Descriptions of the CCS retrofit models

1. Introduction

This file describes the CCS retrofit models. In the main article, the overall modelling steps are described as: (1) estimating the nominal energy balance from the database parameters, (2) calculating flue gases, (3) sizing an Aspen Plus model of amine capture, compression and liquefaction, (4) determining the new energy balance after integrating the capture process and (5) estimating capital and operational expenditures. Below, the article figure is re-printed to provide an overview of these steps.

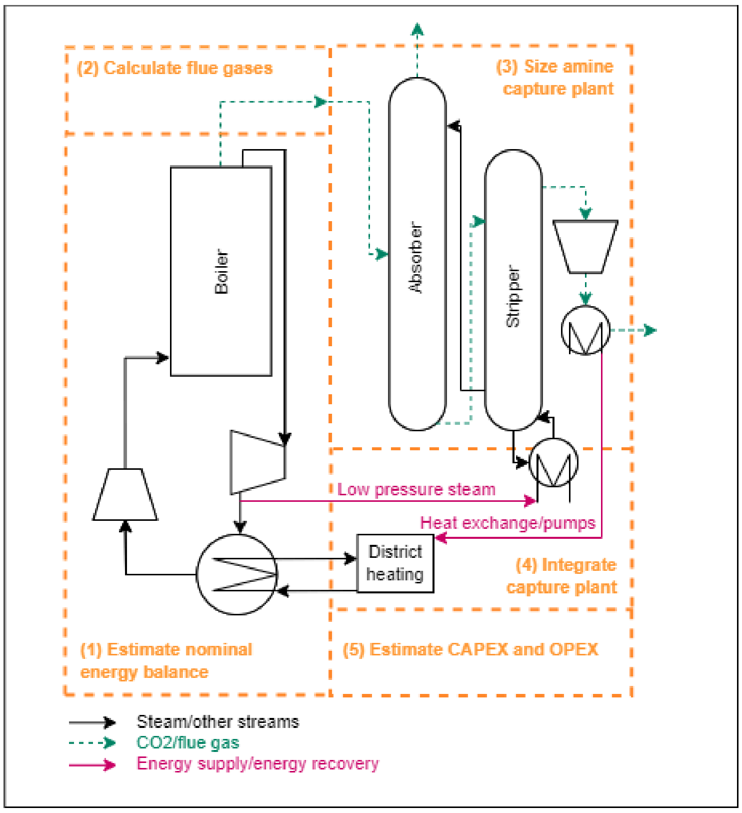
(2) Calculate flue gases
(3) Size amine capture plant
Boiler
Absorber
Stripper
Low pressure steam
Heat exchange/pumps
District heating
(4) Integrate capture plant
(1) Estimate nominal energy balance
(5) Estimate CAPEX and OPEX
Steam/other streams
CO2/flue gas
Energy supply/energy recovery

Figure 1: Model outline for retrofitting a CHP plant with CCS.

Notably, many theoretical and modelling details have been described by e.g. Kumar et al. (2023), Biermann et al. (2019) and Onarheim et al. (2017). However, this file mainly elaborates on model structure and key assumptions specific to this study. *Not all modelling steps and assumptions are described here, as the models contain many calculation steps. But the main features are presented.*

The models and data are collected in the GitHub repository BECCS-Sweden. While many other files also populate the repo, it's overall structure is illustrated below. The controller.py and model.py are the most important files. In each controller, you specify what plant data should be read and what uncertainties, levers and outcomes should be explored. The controller then calls the model in the corresponding model.py file repeatedly for a specified number of scenarios. Each model file consists of a set of functions corresponding to the modelling steps (1) to (5) previously outlined. These steps are detailed in the following sections.

In [ ]:
```
BECCS-Sweden repository structure
│ scenario_discovery.py
│ parsed_scenarios.xlsx
│
├───CHP experiments
│ │ all_experiments.csv
│ │ CHP data all.csv
│ │ chp_controller.py
│ │ chp_model.py
│
├───PULP experiments
│ │ all_experiments.csv
│ │ Pulp data.csv
│ │ pulp_controller.py
│  │  pulp_model.py
│
├──WASTE experiments
│  │  all_experiments.csv
│  │  WASTE data all.csv
│  │  waste_controller.py
│  │  waste_model.py
│
├──FIGURES
```

2. The woodchip- and waste-fired CHP models

Determining an initial energy balance and preparing Aspen data

The CCS retrofit models for woodchip- and waste-fired plants are very similar so only the woodchip-fired model will be used for demonstration purposes. Firstly, a **CHP** plant object is constructed based on the plant of interest. This plant could e.g. be "Värtaverket KVV8" from the CHP plant data csv file. The nominal energy balance of this plant is then estimated based on the fuel, power and steam data. The **estimate_nominal_cycle()** function essentially guesses various condensation pressures. The function stops when the power and heat generated matches the values of the CHP plant data. At this point, the steam states of the CHP have been estimated.

In [ ]:
```
CHP = CHP_plant(
    name=plant_data["Plant Name"],
    fuel=plant_data["Fuel (W=waste, B=biomass)"],
    Qdh=plant_data["Heat output (MWheat)"],
    P=plant_data["Electric output (MWe)"],
    Qfgc=plant_data["Existing FGC heat output (MWheat)"],
    ybirth=plant_data["Year of commissioning"],
    Tsteam=plant_data["Live steam temperature (degC)"],
    psteam=plant_data["Live steam pressure (bar)"],
)

CHP.estimate_nominal_cycle()
```

To estimate how an amine retrofit would impact the energy balance of the plant, we adapted the Aspen Plus V12.1 model detailed in Kumar et al. (2023) to various flue gas volumes and capture rates. The model includes compression and liquefaction as specified by Deng et al. (2019). A screenshot of the model is shown below, although not all components are necessarily shown or active.

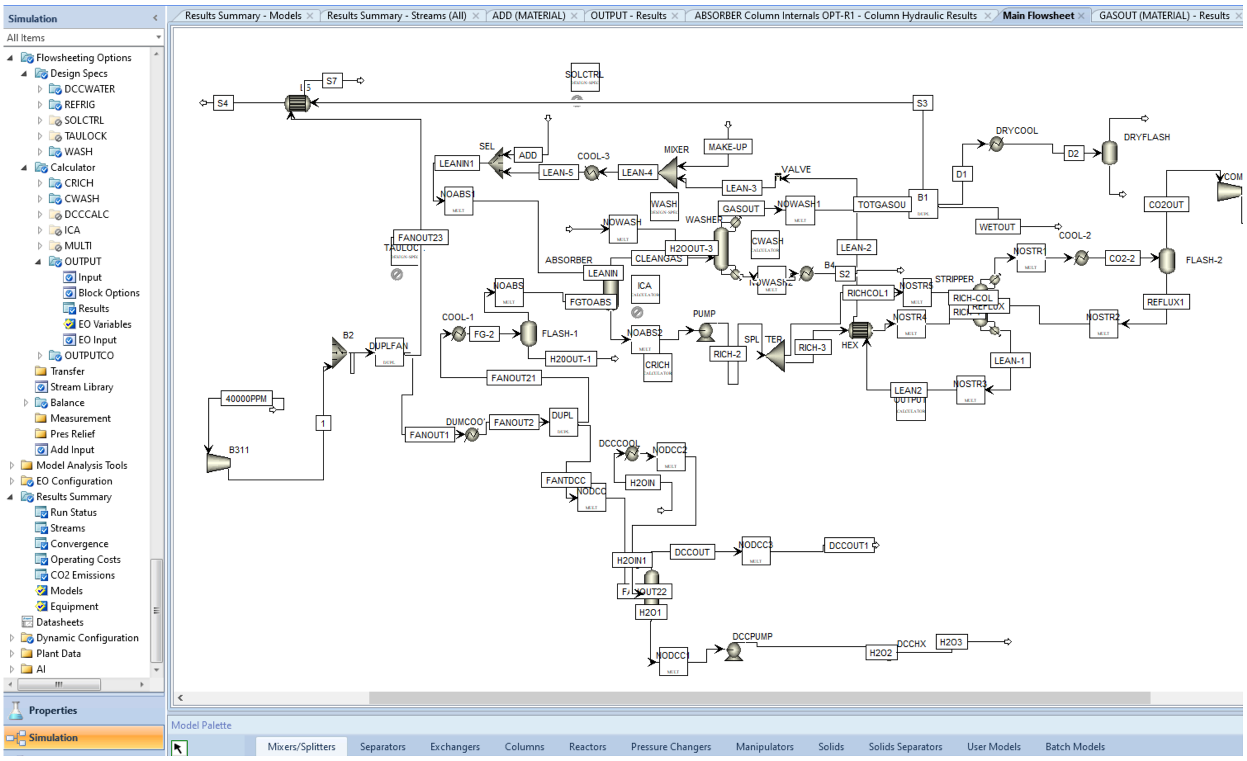

Figure 2: Screenshot of the amine Aspen model used.

The (wet) flue gas composition for woodchip- and waste-fired CHP plants were taken from Kumar et al. (2023) and Hammar (2022), respecitvely. The model was adapted to various flue gas flows between 3 and 170 kg/s and capture rates between 78 and 94 %. The results were saved to e.g. the AspenWoodchip.csv file, which is illustrated below. The key idea is that this data can be used to quickly estimate various outputs from the Aspen model by interpolating based on flue gas flow and capture rate. This lets us estimate model outputs for thousands of scenarios, without having to adjust the Aspen model manually for each scenario.

Wet flue gas composition (woodchip)		
CO2	vol.%	16.0
O2	vol.%	3.2
H2O	vol.%	5.3
N2 and inert gases	vol.%	75.5

Wet flue gas composition (waste)		
CO2	vol.%	11.0
O2	vol.%	7
H2O	vol.%	5
N2 and inert gases	vol.%	77

```
In [30]: import pandas as pd
         aspen_df = pd.read_csv("AspenWoodchip.csv", sep=';', decimal='.')
         aspen_df = aspen_df.iloc[3:7] # Only showcases 4 rows of Aspen results, but 35 are available.
         aspen_df.head()
```

```
Out[30]:
   CO2  Flow  Rcapture      RCO2        Qreb        Treb     Wpumps        Wcfg         Wc1         Wc2  ...       COP
3   16   170        82  0.820246  110585.762  121.148653  129.565427  3631.66210  2295.00515  2138.99307  ...  2.566057
4   16   170        78  0.780732  104472.462  121.162140  125.601333  3631.66210  2184.41889  2035.92417  ...  2.566057
5   16   140        94  0.941247  112530.899  121.058270  122.285697  2990.78056  2168.71060  2021.28605  ...  2.566057
6   16   140        90  0.898620  102863.646  121.108749  114.808663  2990.78056  2070.60917  1929.85210  ...  2.566057

4 rows × 58 columns
```

Given the Aspen outputs, including e.g. reboiler duty, pump and compressor work, temperature levels and cooling demands etc., we construct a set of interpolators - one for each output. The Aspen outputs can thus later be estimated by just passing a flue gas volume and a capture rate to the interpolators.

```
In [ ]: aspen_interpolators = create_interpolators(aspen_df)
```

The CCS retrofit model

At this point, the nominal energy balance of the CHP is determined and interpolators of the Aspen outputs are prepared. The CHP object and Aspen interpolators can then be passed as an input to the main model **CCS_CHP()**.

Scenario-specific input parameters, i.e. uncertainties and levers, are specified in the beginning of the model. These correspond to Table 2 in the main article. While many scenarios are sampled, one scenario could look something like this:

```
In [ ]: def CCS_CHP(
            dTreb=10,
            Tsupp=86,
            Tlow=38,
            dTmin=7,
            COP = 3,
            U = 1500,

            alpha=6.12,
            beta=0.6336,
            CEPCI=880,
            fixed=0.06,
            ownercost=0.2,
            WACC=0.05,
            yexpenses=3,
            rescalation=0.03,
            i=0.075,
            t=25,
            celc=40,
            cheat=0.80,
            cbio=30,
            cMEA=2,
            cHP=0.86,
            cHEX=0.571,

            time=5000,

            duration_increase="1000",
            rate=0.90,
            heat_pump=True,

            chp_interpolators=aspen_interpolators,
            CHP=CHP
        ):
```

The model now calls various functions to determine the final energy penalties and costs of the CCS retrofit. Firstly, the amount of flue gases, feedstock use and CO2 generated are calculated using the **burn_fuel()** function. The necessary data is taken from Kumar et al. (2023) and Hammar (2022), e.g. to calculate the molar mass of the flue gases. The volume of the flue gases is assumed to be 22.4 Nm3/kmol flue gas, i.e. an ideal gas assumption. This is also where any additional biomass feedstock is quantified, as a result of potential increases in annual plant utilization.

```
In [ ]: CHP.burn_fuel()
```

Then, the (scenario specific) flue gas volume and capture rate are used as inputs to the **size_MEA()** function. This function utilizes the aforementioned Aspen interpolators to estimate how the amine capture plant, including compression and liquefaction stages, could be designed. Some key outputs are the reboiler duty and the total compressor and pump work.

```
In [ ]: CHP.size_MEA(rate, chp_interpolators)
```

Given the required reboiler duty and compressor and pump work, the amine capture plant can be powered. This is simulated using the **power_MEA()** function by calculating how much condensing steam would be required at a temperature (pressure) level slightly above the reboiler temperature. This temperature difference, dTreb, varies by scenario and the reboiler temperature is specified from the Aspen data. The required steam massflow is calculated and drawn at the calculated pressure level. This reduces the steam available for power and heat generation. The net power output is further reduced by subtracting the required compressor and pump work of the capture plant from the CHP turbine output.

```
In [ ]: CHP.power_MEA()
```

As all CHP plants were assumed to recover waste heat from the capture process through heat exchange with a district heating network, we developed functions which estimate the heat available for heat recovery. This process is largely guided by energy targeting and pinch analysis principles as detailed by Kemp (2007).

The **select_streams()** function is used to organize the Aspen output data. It ensures that various streams of the capture plant are named and their respective heating/cooling demand and in- and out- temperatures are specified.

The **merge_heat()** function merges the separate streams into an idealized composite curve. In this study, we regard the composite curve as representative of all the individual streams if perfect heat exchange would be possible between all streams.

Finally, the **recover_heat()** function estimates how much heat from the composite curve that is recoverable, i.e. that could be used to heat district heating water. This function shifts the composite curve based on the (scenario specific) dTmin temperature. It then identifies how much heat of this composite curve that is at a temperature higher than the district heating water temperature - and the district heating in- and out- temperature is also scenario dependent. We assumed that no heat above 47 C could be directly exchanged for district heating purposes. However, any heat below 47 C that is available from the capture plant could be heat pumped to an appropriate district heating temperature. This use of heat pumps and their COP efficiency is scenario dependent. After having recovered heat, the energy balance of the CHP is adapted, mainly by increasing the heat output (and possibly reducing power output to cover heat pump electricity demand).

```
In [ ]: stream_data = CHP.select_streams()
        composite_curve = CHP.merge_heat(stream_data)
        CHP.recover_heat(composite_curve)
```

Now, the CAPEX and OPEX of the CCS retrofit can be calculated. We are mainly guided by Hassan et al. (2019) and the NETL methodology described by Theis (2021), and we re-print their illustration below. The logic is to estimate a baseline CAPEX cost, and then apply various escalation factors to this cost. This is done within the **CAPEX_MEA()** function.

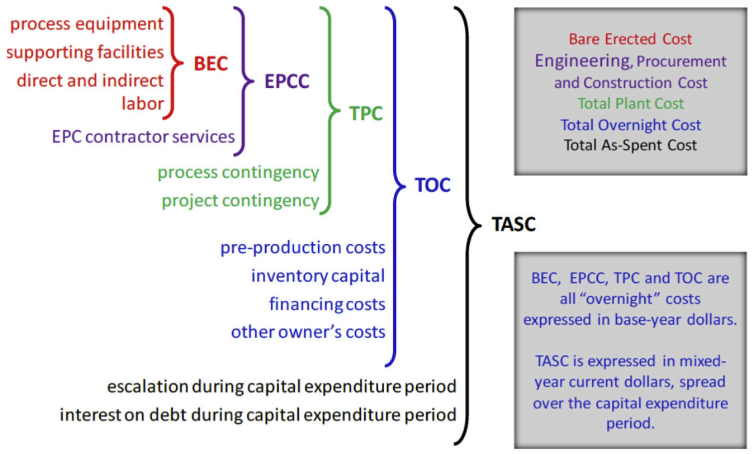
process equipment
supporting facilities
direct and indirect labor
BEC
EPC contractor services
EPCC
process contingency
project contingency
TPC
pre-production costs
inventory capital
financing costs
other owner's costs
TOC
escalation during capital expenditure period
interest on debt during capital expenditure period
TASC
Bare Erected Cost
Engineering, Procurement and Construction Cost
Total Plant Cost
Total Overnight Cost
Total As-Spent Cost
BEC, EPCC, TPC and TOC are all "overnight" costs expressed in base-year dollars.
TASC is expressed in mixed-year current dollars, spread over the capital expenditure period.

Figure 3: The NETL cost estimation methodology is based on escalation factors applied to a base cost estimate.

We use the CAPEX estimate in Figure 6 by Eliasson et al. (2022), who also base their estimate on Hassan et al. (2019). This estimate represents the TPC, i.e. costs including process and project contingencies. The estimate uses the cost year of 2016. Therefore, a Chemical Engineering Plant Cost Index (CEPCI) was used to adjust the estimate to 2026, which was assumed to be the cost year for our study.

```
In [ ]: CAPEX = alpha * Vfluegases ^ beta # Where alpha and beta are scenario dependent,
        # but in Eliasson et al. (2022) they are 6.12 and 0.6336.
        CAPEX = CAPEX * CEPCI_adjustment # CEPCI values taken from University of
        # Manchester (2024)
```

Given this base CAPEX, the TOC is calculated by assuming a (scenario dependent) total overnight cost factor, roughly based on the Theis (2021) report values. The report then outlines how to calculate the final cost, TASC, using an escalation rate, assumed years of capital expenditures, weighted average cost of capital, economic lifetime and discount rates. These values are difficult to accurately specify, which is why they were made scenario dependent. Various combinations of these values were thus explored.

```
In [ ]: CAPEX = CAPEX * TOC_escalation
        CAPEX = CAPEX * (escalation + cfunding) # The escalation and cfunding factors
        # are calculated based on Theis (2021). This CAPEX is now the TASC.
```

Notably, the CAPEX was also increased depending on how much heat exchanger and heat pump capacity is used in the **recover_heat()** function. We use the below equations to estimate these costs.

```
In [ ]: CAPEX_hex = cost_hex * A_hex ^ 0.9145 # Where cost_hex is scenario dependent,
        # but in Eliasson et al. (2022) it is 0.571.
        CAPEX_hp = cost_hp * Qhp # Where cost_hp is scenario dependent. Based on
        # Bergander and Hellander (2024).
```

Finally, CAPEX was used to estimate the fixed OPEX. And CAPEX was then annualized over the economic lifetime.

```
In [ ]: fixed_OPEX = fixed_share * CAPEX # Where the fixed_share is scenario dependent,
        # but in Beiron et al. (2022) it is 0.06.
        annualization = (i * (1 + i) ^ t) / ((1 + i) ^ t - 1) # Where i is discount
        # rate and t is economic lifetime
        aCAPEX = annualization * CAPEX
```

The variable OPEX is calculated using the **OPEX_MEA()** function. This represents a key methodological choice - we assume that the energy cost should be determined as the difference between net energy revenues (from both heat and electricity sales) before and after the CCS integration. These net revenues are calculated and compared on an annual basis. Other OPEX costs were mostly neglected, except for the cost of MEA solvent makeup.

```
In [ ]: energy_OPEX = revenues_nominal - revenues # Calculated by multiplying power by
        # power prices and heat by heat prices.
        other_OPEX = solvent_makeup_losses * cost_makeup # Solvent losses are taken
        # from the Aspen data. Their cost is scenario based.
```

Now the energy balances and costs are calculated, so the Key Performance Indicators (KPIs) can be determined. The penalty_power and penalty_heat are calculated as the difference between power and heat generated before and after the CCS integration. The extra_biomass is calculated by multiplying the fuel intensity (calculated in the **estimate_nominal_cycle()** function) with the increase in plant utilization (which was specified when the **CCS_CHP()** model was instantiated). *The KPIs below are the main outputs of the model.*

```
In [ ]: capture_cost = (aCAPEX + fixed_OPEX + energy_OPEX + other_OPEX) / captured_CO2
        # [EUR/tCO2]
        penalty_services = (penalty_power + penalty_heat) / captured_CO2 # [kWh/tCO2]
        penalty_biomass  = extra_biomass / captured_CO2 # [kWh/tCO2]
```

3. The pulp mill models

Determining an initial energy balance and preparing Aspen data

The CCS retrofit models for pulp mills follow a similar logic to those of CHP plants. Firstly, we specify some energy balance assumptions for all mills - mainly that the recovery boiler generates 18 GJ steam per air dried ton of pulp produced, and the various pulp mill processes demand 11 GJ steam per air dried ton of pulp produced (EU Joint Research Centre, 2015). We also assume the mills are operated 8000 hours per year. We decided to not model this as an uncertainty, as it would cause misalignment with the constant pulp production data used. Furthermore, we decided that the lowest pressure level of any steam stream would be 0.1 bar.

After making these energy balnce assumptions, a **pulp_plant** object is constructed based on any of the pulp mills in the data csv file. The nominal energy balance of this mill is then estimated using the **estimate_nominal_cycle()** function. Here, the main goal is to determine how much excess steam could be available from the recovery and bark boilers, after steam has been used to cover mill process demands. This is done by calculating: the steam generated from the recovery boiler, i.e. 18 GJ times the production capacity; the steam generated from the bark boiler, i.e. a percent share of the steam generated from the recovery boiler (e.g. ~15 % at the Mönsterås pulp mill); and the steam demand for the kraft process, i.e. 11 GJ times the production capacity. These are then summarized according to Equation (1) in the main article, which is re-iterated below.

$$Q_{\text{available steam}}\,[\text{MWh p.a.}] = Q_{\text{recovery boiler}} + Q_{\text{bark boiler}} - Q_{\text{process demands}}$$

The function then estimates the massflow of available steam that could be expanded from live steam states down to low pressure (~4 bar) or condensing pressure (0.1 bar). The nominal power production can therefore also be calculated, as we assume this steam is expanded through turbines before being used to cover process demands (at ~4 bar). At this point, the nominal steam states and power generation of the pulp mill have been estimated.

```
In [ ]: energybalance_assumptions = {
            "recovery_intensity": 18,       #[GJ/t pulp]
            "heat_intensity": 11,           #[GJ/t pulp]
            "condensing_pressure": 0.1,     #[bar]
            "time": 8000                    #[h p.a.]
        }

        pulp_plant = PulpPlant(
            name=plant_data['Name'],
            pulp_capacity=plant_data['Pulp capacity'],
            bark_share=plant_data['Bark capacity'],
            rp=plant_data['RP'],
            rt=plant_data['RT'],
            bp=plant_data['BP'],
            bt=plant_data['BT'],
            lp=plant_data['LP'],
            energybalance_assumptions=energybalance_assumptions
        )

        pulp_plant.estimate_nominal_cycle()
```

The process for estimating how an amine retrofit would impact the energy balance of the pulp mill is analogous to that of CHP plants. We adapted the Aspen Plus model to various flue gas volumes and capture rates. Notably, only the flue gas from the recovery boiler was considered for capture. It's (wet) flue gas composition was taken from Onarheim et al. (2017). Again, running the Aspen model repeatedly resulted in a data set of Aspen outputs, e.g. reboiler duty, pump and compressor work etc. We used this data to construct a set of interpolators that, given a new flue gas flow and capture rate, could be used to quickly estimate various parameters across thousands of scenarios.

Wet flue gas composition (pulp recovery boiler)		
CO2	vol.%	13.0
O2	vol.%	2.3
H2O	vol.%	17
N2 and inert gases	vol.%	68

The CCS retrofit model

The nominal energy balance of the pulp mill is now determined and interpolators of the Aspen outputs are prepared. The pulp mill object and Aspen interpolators can then be passed to the main model **CCS_Pulp()**.

Similar to CHP plants, the uncertainties and levers are specified in the beginning of the model. These correspond to Table 3 in the main article. An example scenario could look something like this:

In [ ]:
```
def CCS_Pulp(
    factor_recovery = 0.4106,
    factor_bark = 0.322285714,
    fluegas_intensity = 10188.75,
    COP = 3,
    k = -29.998,
    m = 1.248,

    alpha=6.12,
    beta=0.6336,
    CEPCI=600/550,
    fixed=0.06,
    ownercost=0.2,
    WACC=0.05,
    yexpenses=3,
    rescalation=0.03,
    i=0.075,
    t=25,
    celc=40,
    cbio=30,
    cMEA=2,
    cHP=0.86,

    SupplyStrategy = "SteamLP",
    rate = 0.90,
    BarkIncrease = "30",

    pulp_interpolators=aspen_interpolators,
    PulpPlant=pulp_plant
):
```

Again, the model begins by calculating the amount of flue gases, feedstock use and CO2 generated using the **burn_fuel()** function. This is where any extra bark boiler use is accounted for. Flue gas factors are estimated based on the modelled data used by Onarheim et al. (2017), notably the production capacity of 800 000 air dried tons per year and the flue gas data of their Table 2. These factors are then scaled to the production capacities of the mills considered in this study. For example, a mill producing 400 000 tons of pulp per year would produce roughly half as much flue gases as the Onarheim model mill. Again, the flue gases are considered ideal, i.e. 22.4 Nm3/kmol gas. Just like for CHP, the flue gas flow and capture rate is used in the **size_MEA()** function to interpolate between the Aspen data points, and thus to estimate various parameters such as reboiler duty, compressor demands, solvent makeup etc.

In [ ]:
```
pulp_plant.burn_fuel()
pulp_plant.size_MEA(rate, pulp_interpolators)
```

A key feature of the pulp CCS model is that *only one out of three energy supply strategies is employed in each scenario*. Their model implementation will now be described briefly.

Firstly, the mill owner could utilize high pressure, live steam to supply the capture plant reboiler with heat. This strategy, "SteamHP", has a dedicated function called **feed_then_condense()**. The function subtracts the reboiler duty from the available steam of the mill. The remaining steam is then expanded as normal to lower pressure levels. The various steam mass flows are then re-calculated and the relative loss in power generation is estimated.

Alternatively, the mill owner could utilize low pressure steam (~4 bar) to supply the capture plant reboiler with heat. This strategy, "SteamLP", has a function called **expand_then_feed()**. The function first expands (through turbines, presumably) the live steam of the recovery and bark boilers to their respective low pressure levels. It then forms a merit order for supplying the reboiler heat demand. It first tries to supply the whole demand using low pressure recovery boiler steam. It then supplies any remaining heat demand with bark boiler steam. Should this be insufficient (which is unlikely), any remaining reboiler demand is assumed to be met by direct electric heating. Again, the various steam mass flows are then re-calculated and the relative loss in power generation is estimated.

Finally, the mill owner could utilize the *theoretical option* of heat pumps to cover the reboiler heat demand. This strategy, "HeatPumps", has it's dedicated function called **recover_and_supplement()** and is inspired by the setup introduced by Jenssen et al. (2024). The idea is to lift the temperature of available excess heat of the mill to an appropriate reboiler temperature, i.e. above 120 C. Heat above 60 C is considered useful for this purpose, while recognizing that the temperature lift is large. Available excess heat above 60 C has been estimated by Cruz et al. (2024) as:

$$Q_{\text{excess}\geq 60^{\circ}C}\,[\text{GWh p.a.}] = k + m \times \text{market pulp}\,[\text{air dried tons p.a.}]$$

Where k, m and the heat pump COP efficiency are scenario dependent. This excess heat is lifted to meet the reboiler heat demand at the expense of lost electricity (determined by the COP). In many scenarios this is insufficient to cover the reboiler demand. Again, a merit order is constructed where low pressure recovery boiler steam is prioritized to meet the remaining demand. If insufficient, then low pressure bark boiler steam is also utilized to meet the remaining demand. After the reboiler demand is met, the various steam flows are re-calculated and the relative loss of power generation (including for heat pump work) is estimated.

```
In [ ]: if SupplyStrategy == "SteamHP":
            PulpPlant.feed_then_condense()

        elif SupplyStrategy == "SteamLP":
            PulpPlant.expand_then_feed()

        elif SupplyStrategy == "HeatPumps":
            PulpPlant.recover_and_supplement()
```

Now, the new energy balance of the mill has been determined and CAPEX and OPEX can be calculated. The **CAPEX_MEA()** calculations, including for fixed OPEX, are the same as the calculations used for CHP plants so we will not describe them again. The only difference is that the heat pump capacity for CHP plants is for upgrading capture plant excess heat, while heat pump capacity for pulp mills is for upgrading mill process excess heat. The **OPEX_MEA()** function simply calculates the cost of foregone revenues from reduced power generation, the cost of any additional biomass feedstock and the cost of solvent makeup.

```
In [ ]: energy_OPEX = lost_power * price_electricity + extra_biomass * cost_biomass
        # These factors are all scenario dependent.
        other_OPEX = solvent_makeup_losses * cost_makeup # Solvent losses are taken
        # from the Aspen data. Their cost is scenario dependent.
```

Finally, the energy balances and costs are calculated, so the Key Performance Indicators (KPIs) can be determined. These KPIs are the main outputs of the model.

```
In [ ]: capture_cost = (aCAPEX + fixed_OPEX + energy_OPEX + other_OPEX) / captured_CO2
        # [EUR/tCO2]
        penalty_services = lost_power / captured_CO2 # [kWh/tCO2]
        penalty_biomass  = extra_biomass / captured_CO2 # [kWh/tCO2]
```