Notes on `Grimsel_H`

GeneRal Integrated Modeling environment for the Supply of Electricity and Low-temperature Heat

M. C. Soini

April 26, 2018

Contents

1 Introduction

- This document describes
  1. an framework implemented in Python using the Pyomo optimization module 2
  2. a simple national-scale linear dispatch model including Switzerland and its neighboring countries 3
- Power plants are aggregated by type in each country. The generation of variable power is based on fixed historical profiles from historic data.
- The inter-node power exchange between countries is optimized endogenously; however, electric grid constraints within the countries are not implemented (copper plate approximation).
- Stylized model of hydro power: reservoirs with natural inflow are aggregated in each country, both run-of-river plants and pumped hydro storage (PHS) are a separate power plant types with specific operational constraints.
- The operation during a single year with hourly resolution is optimized.
- All over-night investment costs are annualized using an appropriate capital recovery factor.
- The modelling framework Grimsel is designed in a flexible way such that any time resolution > 1 hour and any subset of hours (e.g. week_id=27) can be chosen as a simple keyword argument for fast model runs during testing and development.

2 Implementation summary

The model is based on three classes

- ModelBase
- ModelLoop
- IO

The basic role of each class is described below. This is followed by a summary of their typical usage when running the model.

In addition to these three classes, tailor-made analysis and plotting modules allow for an efficient processing of the model results. These parts of Grimsel are not discussed here.

2.1 BaseModel

The core model is a class inheriting from pyomo.environ.ConcreteModel. Main purposes:

- definition of the linear problem including all constraints and variables.
- selection/creation of subsets of input data (in case we don't want to use all nodes, or energy carriers, or use a temporal resolution > 1 hour or just a subset of the hours of the year)
- handling of CPLEX solver initialization and solver calls
- various model-related convenience methods for pre-solving and infeasibility avoidance (see below)

The description of the methods follows below:

2.1.1 __init__

__init__ calls the __init__ of the pyomo.environ.ConcreteModel and assigns basic keyword arguments, most importantly the time resolution nhours, the selection of the nodes slct_node as well as the selection of the energy carriers slct_encar (not relevant so far, since *EL* only). The output schema name sc_out is generated from a time stamp unless it is provided in the kwargs.

2.1.2 build_model

This calls all methods relevant for the final definition of the model:

- definition of the list of sets dictionary setlst which holds all model sets for convenient access; as an example, the following code snippet defines the sets of run-of-river plants ror, as well as the set of all power plants:

- mapping of input data to the selected time resolution by averaging of the hourly profiles
- calculation of the maximum demand from the demand input profile for each node (after averaging to the selected time resolution); this is used in some constraints
- definition of all sets, parameters, variables and constraints, which are primarily found in the corresponding mixin classes. This just follows standard pyomo procedures, facilitated by some convenience methods
- initialization of the solver

Note that this method can only be called after the definition of the input dataframes, which is typically taken care of by the IO class.

2.1.3 switch_soln_file

Warmstart and solution files names are alternated to allow to use the last solution as starting values while avoiding excessive disk space use (which would be the case if we kept all solution files). This method returns the variables isolnfile, which alternates between 0 and 1, as well as the temporary solution file name solnfile, which alternates between e.g. **TEMPDIR/manual_soln_file_0.cplex.sol** and **TEMPDIR/manual_soln_file_1.cplex.sol**

2.1.4 run

This method calls the solve method of the solver object. Upon completion the warmstart and solution file names are switched using the method switch_soln_file.

2.1.5 presolve_fixed_capacities

Runs the model using fixed power plant capacities (or other variables). This is to obtain starting conditions which are closer to the expected solution. Presolve variable values are taken from the schema defined by the string attribute self.sc_warmstart. Note that the usefulness of this approach depends strongly on the specific design of the model constraints and is currently not being used. (It seemed to be highly useful when constraints on the hour-to-hour ramping of the electricity production were in place.)

2.1.6 fill_peaker_plants

Calculate capacity of expensive peaker plants from power capacity and demand parameters and set the capacity parameter cap_pwr_leg accordingly. This serves to avoid infeasibilities due to insufficient installed power plant capacity in case the capacities are fixed. Infeasibilities would be the case if the the peak (residual) demand is greater than than the total capacity of the (dispatchable) power plants.

2.2 ModelLoop

This class is used to manage consecutive model calls while varying parameters, constraint activations/deactivations, etc. The most important attributes are described here below, followed by the most important methods.

2.2.1 nsteps

The basic layout of the model runs/parameter variations is defined by the list attribute nsteps, which contains the name of the dimension name, the number of steps, and the type of range. Examples:

- ('swvr', 10, numpy.linspace), energy share of wind solar, 10 steps; this is varied between 0 and a maximum value (e.g. 50%), therefore numpy.linspace is used to define the step values.
- ('swco', 3, numpy.arange), cost of CO2 emissions; here, the step value are used as keys to get the cost from a dictionary (which is convenient if the values of CO2 emission prices are not equally spaced); therefore, numpy.arange is used
- ('swcd', 2, numpy.arange), charging only wind/solar yes/no; here, the step value is used to switch a constraint on or off; therefore, numpy is arange is used (which returns [0, 1] for np.arange(2)).

2.2.2 df_def_loop

This pandas dataframe attribute holds the definitions of all model runs for each run_id.

Based on the nsteps, the dataframe df_def_loop is defined. Initially it contains all combinations of all step values. This can be adjusted outside the ModelLoop class, according to needs. df_def_loop is indexed with the run_id and for each run_id defines all

- **steps** with column name equal the dimension name
- **step ids** with column name equal the dimension name plus suffix _id
- **step value** with column name equal the dimension name plus suffix _vl; these are typically strings and defined within the run_id loop

Example:

run_id	'swvr'	'swcd'	'swvr_id'	'swcd_id'	swvr_vl'	swcd_vl'
0	0	0	0	0	'0.0%'	'Chg all'
1	0.5	0	1	0	'50.0%'	'Chg all'
2	1	0	2	0	'100.0%'	'Chg all'
3	0	1	0	1	'0.0%'	'Chg ws'
4	0.5	1	1	1	'50.0%'	'Chg ws'
5	1	1	2	1	'100.0%'	'Chg ws'

2.2.3 select_run

This method obtains all relevant indices and parameters for a certain slct_run_id. For this purpose the relevant row of the dataframe df_def_loop is selected and converted into a set of convenient dictionaries.

2.2.4 Other methods

Other methods deal with def_loop database table definition and writing, among some convenience methods not described here.

2.3 IO

This class handles the data exchange between the model and the database, i.e.:

- reading data from the database and define corresponding dataframe attributes of the BaseModel; these tables are written to the output database schema in order to preserve a consistent set of model data
- (modified) input data is written to the output database schema; this concerns hourly profiles whose time resolution is changed

- extraction of results and parameter data from the pyomo objects
- initialization and writing of the output database tables

The description of the main attributes and methods follows:

2.3.1 Lists var_*, par, dual

These define the data to be extracted from the model objects and written to the database. They are lists of tuples ('object_name', ('set1', 'set2', ...)):

```
var_yr = [
('erg_yr', ('pp_id', 'ca_id', 'bool_out')),
('erg_fl_yr', ('pp_id', 'nd_id', 'ca_id', 'sf_id')),
('pwr_ramp_yr', ('pp_id', 'ca_id')), ...]
```

The column 'bool_out' shown in the example is a binary index which encodes whether the corresponding energy value is an outward flow *from the node's perspective*. E.g.:

- False: power production from power plants, power discharge from storage
- True: exports, storage charging power, demand, imports

This is used in the analysis.

2.3.2 Dictionary dict_cross_table

Some model variables are appended to other output tables. This dictionary defines the cross-table writing as {'variable_or_parameter_name': 'target_database_table'}. Examples: * Yearly inter-node transmission is appended to yearly energy generation; * inter-node transmission per time slot is appended to per time slot power generation table; * the demand profile parameter is appended to the time slot power generation table.

2.3.3 write_runtime_tables

Some input tables depend on model parameters (time resolution). They are written to the output database schema on runtime.

2.4 Summary of relationships between the classes IO/ModelBase/ModelLoop

The ModelLoop class provides the most top-level objects of a typical series of model runs. The other two classes are instantiated by it. The ModelBase is lowest level class and is accessed by both IO and ModelLoop. The relations are defined in the table below.

	ModelLoop	IO	ModelBase
ModelLoop	-	instantiates/has attribute	instantiates/has attribute
IO	instantiated by/ is attribute of	-	has attribute
ModelBase	instantiated by/ is attribute of	is attribute of	-

2.4.1 Sets

Power plant subsets are defined in the `def_plant` table.

2.5 Notes on energy carriers and fuels

The aim is to provide full support for conversions between arbitrary energy carriers while avoiding the definition of explicit hourly input variables. The approach chosen here is composed as follows:

- **Definition of dedicated fuel types which are associated with certain output energy carriers.** For example, `ca_electricity` might be the electricity generated endogenously while `electricity` might be bought electricity. These must be defined separately.
- **Mapping between energy carriers and the corresponding fuels.** In the input table `def_encar`.

- **The input-output energy carrier set** `pp_ndcaca` which contains all relevant combinations of power plants, nodes, output energy carrier, and input energy carrier.
- **An additional demand-term in the supply constraint** which contains the energy carrier "fuel" demand `self.pwr[sy, pp, ca_out] / self.pp_eff[pp, ca_out]` of relevant plants, summed over all power plants which make use of the corresponding energy carrier as an input. Note that the supply constraint is defined for all energy carriers separately.

2.5.1 Multiple fuels for a plant

2.5.2 Storage and energy carriers

Storage assets (defined by `set_def_st=1` in the input table `def_plant`) operate on energy carriers exclusively. Their input and output is equal to the carrier defined in input table `plant_encar`. The "fuel" assigned to the plant in `def_plant` is not relevant in the model and only used for table consistency and result analysis.

2.5.3 Energy carriers are node-based

The production and consumption of energy carriers is node-based, i.e. there is no explicit way of feeding exclusively the output electricity of a wind park into an electrolyzer in order to produce hydrogen if the same node contains other electricity generators. However, this arrangement could still be realized by defining an extra node containing only the wind park and the electrolyzer, and connecting it to the rest of the system through uni-directional inter-node transmission capacity.

2.6 Input data structure

All input data is read from a schema in the PostgreSQL database. The schema name is a keyword argument of the IO class (default: *lp_input*). The database references graph is shown in figure 1 below. This is followed by the description of the input tables.

2.6.1 Description of tables

A *consistent* set of these tables in the corrsponding database schema is strictly required to run the model.

- **def_encar: Table defining energy carriers.**

def_encar		
ca_id	SMALLINT	Energy carrier id.
ca	VARCHAR	Energy carrier name.

- **def_month: Month definition.**

def_month		
mt_id	SMALLINT	Month id
month_min_hoy	SMALLINT	The month's first hour the year. This is used for hydro power filling state boundary conditions.
mt	VARCHAR	Month name

- **def_node: Table defining nodes (for now: one node per country).**

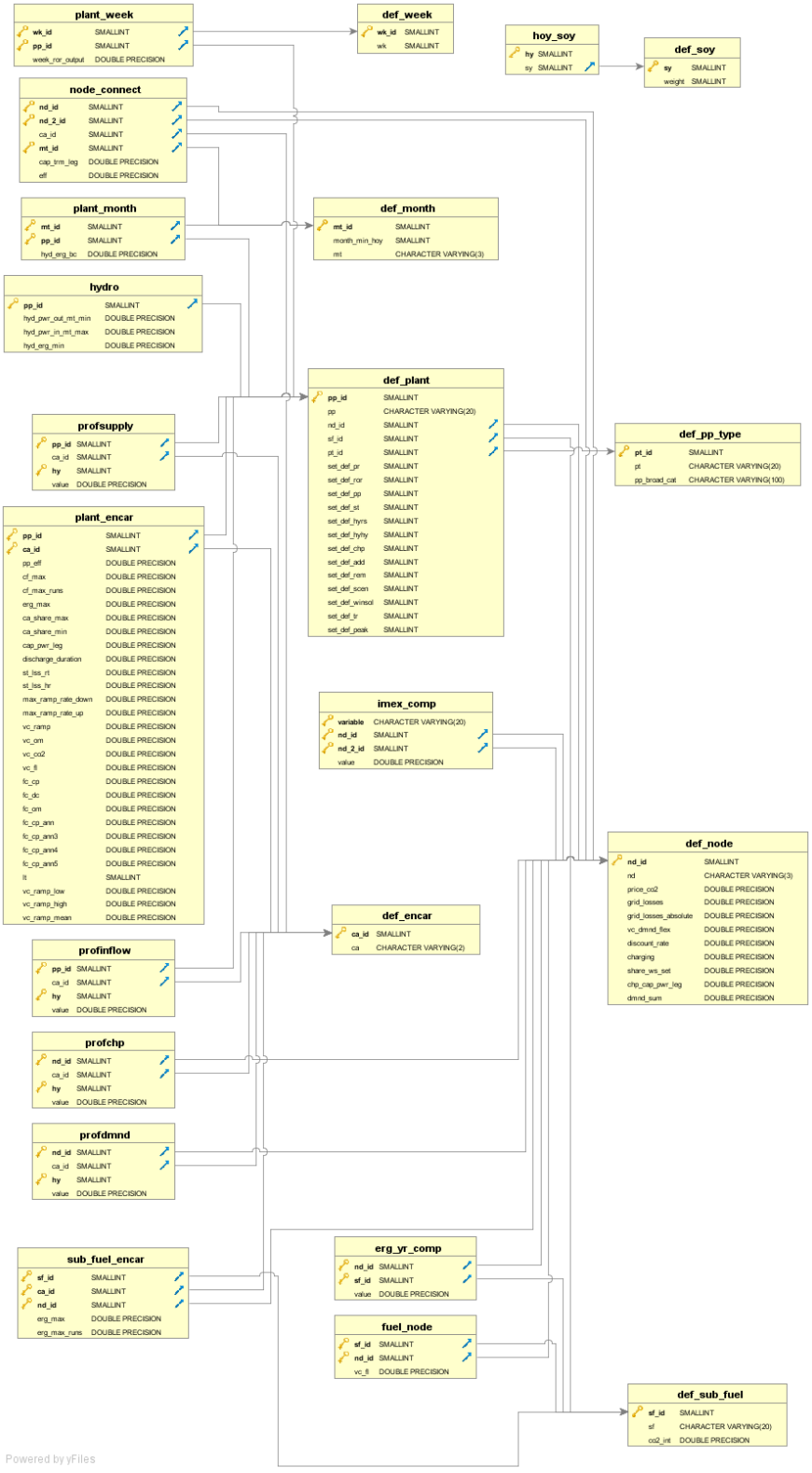

Figure 1: Figure: Input database structure.

def_node		
nd_id	SMALLINT	Node id {0, 1, 2, 3, 4}
nd	CHARACTER VARYING	Node name {DE0, AT0, IT0, FR0, CH0}
price_co2	DOUBLE PRECISION	Price of CO_2 emissions [EUR/t_{CO_2}]
grid_losses	DOUBLE PRECISION	Grid losses as percentage of generation + imports + discharge
grid_losses_absolute	DOUBLE PRECISION	Absolute grid losses in 2015 for comparison and calibration [MWh]
vc_dmnd_flex	DOUBLE PRECISION	Cost of flexible additional demand
discount_rate	DOUBLE PRECISION	Discount rate
charging	DOUBLE PRECISION	Total charging energy of existing pumped hydro plants, for comparison and calibration.
share_ws_set	DOUBLE PRECISION	Total share of wind and solar. This is currently only used to initialize the corresponding model parameter.
chp_cap_pwr_leg	DOUBLE PRECISION	Total electric cogeneration capacity. Used in the constraint set_chp_cap.
dmnd_sum	DOUBLE PRECISION	This is currently used to define the target share of wind and solar.

- **def_plant: Defines power plants**

def_plant		
pp_id	SMALLINT	Power plant id
pp	VARCHAR	Power plant name
nd_id	SMALLINT	Node id
sf_id	SMALLINT	Fuel id (one per power plant)
pt_id	SMALLINT	Power plant type id
set_def_pr	SMALLINT	Yes/No has exogenous profile
set_def_ror	SMALLINT	Yes/No is run-of-river-plant
set_def_pp	SMALLINT	Yes/No is dispatchable plant with fuel
set_def_st	SMALLINT	Yes/No is pure storage plant
set_def_hyrs	SMALLINT	Yes/No is hydro reservoir plant
set_def_hyhy	SMALLINT	Yes/No is hybrid reservoir/PHS plant (obsolete)
set_def_chp	SMALLINT	Yes/No is CHP plant
set_def_add	SMALLINT	Yes/No capacity can be expanded
set_def_rem	SMALLINT	Yes/No capacity can be removed
set_def_scen	SMALLINT	Yes/No is set exogenously by scenarios/parameter variation
set_def_winsol	SMALLINT	Yes/No is wind or solar plant
set_def_tr	SMALLINT	Yes/No is transmission (imports or exports)
set_def_peak	SMALLINT	Yes/No is safety peaker plant (added exogenously to avoid low capacity infeasibilities)

- **def_pp_type: Defines power plant types, which are roughly the same as plant without node-dependence.**

def_pp_type		
pt_id	SMALLINT	Power plant type id
pt	VARCHAR	Power plant type name
pp_broad_cat	VARCHAR	Broad power plant category for analysis.

- **def_soy: Definition of time slots.**

def_soy		
sy	SMALLINT	Time slot of the year
weight	SMALLINT	Number of hours per time slot

- **def_sub_fuel: Defines fuels; called sub-fuel for historic reasons.**

def_sub_fuel		
sf_id	SMALLINT	Fuel id
sf	VARCHAR(20)	Fuel name
co2_int	DECIMAL	CO_2 emission intensity [t_{CO_2}/MWh_{fuel}]

- **def_week: Week definition. Used for run-of-river operation constraints.**

def_week		
wk_id	SMALLINT	Month id
wk	SMALLINT	Week name (equals wk_id)

- **erg_yr_comp: Power production stats for comparison.**

erg_yr_comp		
nd_id	SMALLINT	Node id
sf_id	SMALLINT	Sub-fuel id
value	DOUBLE PRECISION	Yearly electricity generation.

- **fuel_node: Defines node-dependent fuel properties.**

fuel_node		
sub_fuel_id	SMALLINT	Fuel id
node_id	SMALLINT	Node id
vc_fl	DECIMAL	Fuel variable cost [t_{CO_2}/MWh_{fuel}] (not used in model due to calculation of vc_fl per MWh_el in table plant_encar)

- **hoy_soy: Map from hour of the year to time slot of the year; written during the model initialization.**

hoy_soy		
hy	SMALLINT	Hour of the year
sy	SMALLINT	Time slot of the year

- **hydro: Parameter only relevant for hydro power. This could also be included in def_plant or plant_encar, but is its own table for convenience.**

hydro		
pp_id	SMALLINT	Power plant id
hyd_pwr_out_mt_min	DECIMAL	Minimum energy output from hydro reservoirs during any given month. Expressed as share of maximum monthly inflow throughout the year.
hyd_pwr_in_mt_max	DECIMAL	Maximum monthly inflow throughout the year.
hyd_erg_min	DECIMAL	Defines an absolute lower bound for the monthly production.

- **imex_comp: Import/export statistics for calibration.**

imex_comp		
variable	VARCHAR	'import'/'export' from nd_id's perspective
nd_id	SMALLINT	Node id
nd_2_id	SMALLINT	Other node id
value	DOUBLE PRECISION	Yearly electricity [MWh]

- **node_connect: Defines connections between nodes.**

def_node_connect		
nd_id	SMALLINT	Node id
nd_2_id	SMALLINT	The other node id
ca_id	SMALLINT	Energy carrier transmitted.
mt_id	SMALLINT	Month id.
cap_trm_leg	DECIMAL	Power capacity of connection.
eff	DECIMAL	Efficiency of connection. Set to 99% for solution stability.

- **plant_encar: Defines power plant properties depending on electricity carrier output (would currently not need to be its own table due to lack of explicit heat production in the model)**

plant_encar		
pp_id	SMALLINT	Power plant id
ca_id	SMALLINT	Energy carrier id
pp_eff	DECIMAL	Power plant efficiency
cf_max	DECIMAL	Maximum capacity factor (availability); during calibration reference year
cf_max_runs	DECIMAL	Maximum capacity factor (availability); for all other model runs, typical year
erg_max	DOUBLE PRECISION	Maximum yearly power production from this plant (currently not relevant; defined by sub_fuel in table sub_fuel_encar)
ca_share_max	DECIMAL	maximum share of output energy carrier (currently not relevant)
ca_share_min	DECIMAL	minimum share of output energy carrier (currently not relevant)
cap_pwr_leg	DECIMAL	Legacy power capacity
discharge_duration	DECIMAL	Discharge duration for storage and hydro. Serves to calculate the energy capacity from the power capacity.
st_lss_rt	DECIMAL	Roundtrip losses storage
st_lss_hr	DECIMAL	Hourly losses storage
max_ramp_rate_down	DECIMAL	Maximum ramp rate [MW/hour] (decrease of power output); currently not used
max_ramp_rate_up	DECIMAL	Maximum ramp rate [MW/hour] (increase of power output); currently not used
vc_ramp	DECIMAL	Cost of ramping [EUR/MW]
vc_om	DECIMAL	Variable O&M cost [EUR/MWh_{out}]
vc_co2	DECIMAL	Variable CO_2 cost [EUR/MWh_{out}]
vc_fl	DECIMAL	Variable fuel cost [EUR/MWh_{out}]
fc_cp	DECIMAL	Fixed over-night capital cost [EUR/MW_{cap}]
fc_dc	DECIMAL	Fixed decommissioning cost (nuclear) [EUR/MW_{cap}]; currently not used
fc_om	DECIMAL	Annual fixed O&M cost [EUR/MW_{cap}/yr]
fc_cp_ann	DECIMAL	Annualized fixed over-night capital cost [EUR/MW_{cap}/yr]
fc_cp_ann3	DECIMAL	Annualized fixed over-night capital cost [EUR/MW_{cap}/yr] for alternative discount rate
fc_cp_ann4	DECIMAL	Annualized fixed over-night capital cost [EUR/MW_{cap}/yr] for alternative discount rate
fc_cp_ann5	DECIMAL	Annualized fixed over-night capital cost [EUR/MW_{cap}/yr] for alternative discount rate
lt	SMALLINT	Life time of power plant [years].
vc_ramp_low	DECIMAL	Cost of ramping [EUR/MW]; alternative values
vc_ramp_high	DECIMAL	Cost of ramping [EUR/MW]; alternative values
vc_ramp_mean	DECIMAL	Cost of ramping [EUR/MW]; alternative values

- **plant_month: Table defining month-dependent power plant properies.**

plant_month		
mt_id	SMALLINT	Month id
pp_id	SMALLINT	Plant id
hyd_erg_bc	DOUBLE PRECISION	Boundary condition hydro reservoir level [MWh].

- **plant_week: Table defining week-dependent power plant properies.**

plant_week		
wk_id	SMALLINT	Month id
pp_id	SMALLINT	Plant id
week_ror_output	DOUBLE PRECISION	Weekly output of run-of-river plants [MWh].

- **profchp: Definition of exogenous CHP profile**

profchp		
nd_id	SMALLINT	Node id
ca_id	SMALLINT	Output energy carrier id
hy	SMALLINT	Hour of the year
value	DOUBLE PRECISION	Hourly capacity factor.

- **profdmnd: Definition of exogenous demand profile**

profdmnd		
nd_id	SMALLINT	Node id
ca_id	SMALLINT	Output energy carrier id
hy	SMALLINT	Hour of the year
value	DOUBLE PRECISION	Hourly demand [MW]. We choose to define the units as MW since for time resolution > 1 hour the profile is averaged over all hours of the time slot. Multiplication with time slot weight happens in the constraint.

- **profinflow: Definition of exogenous hydro reservoir inflow profile**

profinflow		
pp_id	SMALLINT	Power plant id
ca_id	SMALLINT	Output energy carrier id
hy	SMALLINT	Hour of the year
value	DOUBLE PRECISION	Hourly inflow [MW]. We choose to define the units as MW since for time resolution > 1 hour the profile is averaged over all hours of the time slot. Multiplication with time slot weight happens in the constraint.

- **profsupply: Definition of exogenous wind and solar profiles**

profinflow		
pp_id	SMALLINT	Power plant id
ca_id	SMALLINT	Output energy carrier id
hy	SMALLINT	Hour of the year
value	DOUBLE PRECISION	Hourly capacity factor.

- **sub_fuel_encar: Parameters depending on input fuel and output energy carrier.** ***Notes*** * It would be more natural to constrain the fuel availability (erg_max) through constraints on the input energy. However, the current constraint on the output energy is more straight forward with respect to energy balances.

sub_fuel_encar		
sf_id	SMALLINT	Fuel id
ca_id	SMALLINT	Power plant output energy carrier id
nd_id	SMALLINT	Node id
erg_max	DECIMAL	Maximum energy to be produced (calibration).
erg_max_runs	DECIMAL	Maximum energy to be produced (other model runs).

2.7 Output data structure

The output schema contains all variables and parameters of the model as well as a copy of the input data.

3 Model documentation

3.1 Supply rule

The supply constraint makes sure that within each node n and for each time slot t supply and demand are balanced. Thereby, supply includes all power generation and storage discharge power $p_{pp,t}$ from all plants pp, as well as imports $\tau^{\text{imp}}_{\hat{n},t}$ from all neighboring countries $\hat{n}$. Demand includes the demand profile $d_{n,t}$, flexible additional demand $d^{\text{flx}}_{n,t}$ for curtailments, and charging power $c_{pp,t}$. The supply is reduced by the transmission and distribution losses ϵ.

$$\begin{aligned}&\sum_{pp} p_{pp,t} + \sum_{\hat{n}} \tau^{\text{imp}}_{\hat{n},t} \\ &= (1-\epsilon)\left(\sum_{pp} c_{pp,t} + d_{n,t} + d^{\text{flx}}_{n,t} + \tau^{\text{exp}}\right) \quad \forall t \in \{0,1,...,\bar{t}\} \quad \forall n \in \{0,1,...,N\}\end{aligned}$$

3.2 Modeling of co-generation

- **Must-run condition based on heat production profile**: The output $p_{pp,t}$ from co-generation plants $pp \in C$ (with C the set of CHP plants) can be subject to must-run conditions and therefore impose inflexibilities on the operation of the power system. Fixed CHP profiles $\gamma_{n,t}$ are used to describe the minimum output as a fraction of the capacity P^{tot}_{pp}. See the corresponding section for a description of the CHP profile generation.

$$p_{pp,t} \geqslant \gamma_{n,t} P^{\text{tot}}_{pp} \quad \forall pp \in C \quad \forall t \in \{0,1,...,\bar{t}\}$$

- **Investment and retirement:** The total amount of CHP capacity (and hence of minimum electricity production from co-generation) must not be reduced even if installed capacities can be retired.

$$\sum_{pp \in C} P^{\text{tot}}_{\text{pp}} \geqslant P^{C}_{0} \quad \forall n \in N$$

3.3 Modelling of ramp rate costs

The change of the power output from (thermal) plants leads to a cost which is modeled as a variable cost on the aggregated absolute changes of output over the year.

- **Calculation of the ramp rate**: The hourly difference $\Delta p_{pp,t}$ in electricity production for each relevant plant pp is calculated for each time slot cyclically, i.e. the last time slot $t = \bar{t}$ loops back to the first $t = 0$.

$$\Delta p_{pp,t} = p_{pp,t} - p_{pp,t-1} \quad \forall t \in T, \quad \forall pp \in P$$

- **Calculation of the absolute ramp rate**: The absolute ramp rate $|\Delta p_{pp,t}|$ is calculated using the standard linear programming approach which makes use of two distinct constraints:

$$\begin{aligned}\Delta p_{pp,t} &\leqslant |\Delta p_{pp,t}| \\ \text{and } -\Delta p_{pp,t} &\leqslant |\Delta p_{pp,t}|\end{aligned}$$

- **Total cost of ramping**: Finally, the total cost of ramping enters the objective function as the sum of all absolute changes in power output $\sum_{pp,t} |\Delta p_{pp,t}|$ times the corresponding variable cost. Note that here the summation over the time slots does not include a weight factor for the slot length.

3.4 Modelling of hydro power and storage assets

3.4.1 Parameters

- **Inflow** $i_{pp,t}$: Hourly inflow [MW] corresponding to constant values for all time slots t of each month.
- **Energy capacities** $E^{\text{tot}} = E^{\text{leg}}$ **[MWh]**: for hydro plants, the total existing reservoir capacity equals the existing (legacy) capacity. Capacity additions and retirements are not considered. The same holds for the power capacities: $P^{\text{tot}} = P^{\text{leg}}$ [MW].
- **Boundary conditions of filling levels** $l^{\text{bc}}_{pp,t}$ for time slots $t \in T^{\text{bc}}_{pp}$ from the corresponding sub-set of time slots T^{bc}_{pp}. (In practice this applies to the first time slot of January).

3.4.2 Constraints

- **Reservoir level (reservoirs)** $e_{pp,t}$ in time slot t, plant pp: Filling level during last time slot plus inflow $i_{pp,t}$ minus electricity production $p_{pp,t}$, times the weight of time slots w_t [hours]. The last time step $t = \bar{t}$ loops back to the first $t = 0$, i.e. time is circular for each year.

$$e_{pp,t} = e_{pp,t-1} + (i_{pp,t} - p_{pp,t}) \cdot w_t \quad \forall t \in \{0, 1, ..., \bar{t}\}$$

- **Reservoir level (Pumped hydro storage) and state of charge (SOC, other storage assets)** Pumped hydro storage is modeled like all other storage assets. No natural inflow is assumed to be available. The total round-trip losses ϵ are partly attributed to charging and discharging.

$$e_{pp,t} = e_{pp,t-1} + \left(c_{pp,t}\sqrt{1-\epsilon} - p_{pp,t}/\sqrt{1-\epsilon}\right) \cdot w_t \quad \forall t \in \{0, 1, ..., \bar{t}\}$$

- **Boundary conditions** on reservoir levels from reported filling levels l^{bc}_{pp} during a subset of hours T^{bc}_{pp}. Typically 50% at the beginning of the year.

$$e_{pp,t} = l^{\text{bc}}_{pp} \quad \forall t \in T^{\text{bc}}_{pp}$$

- **Capacity constraints** of power output $p_{\text{pp,t}}$, charging $c_{\text{pp,t}}$ and reservoir filling levels/SOC due to limited capacities E^{tot} and P^{tot}:

$$\begin{aligned} e_{\text{pp,t}} &\leqslant E^{\text{tot}}_{\text{pp}} \quad \forall t \in \{0, 1, ..., \bar{t}\} \\ p_{\text{pp,t}} &\leqslant P^{\text{tot}}_{\text{pp}} \quad \forall t \in \{0, 1, ..., \bar{t}\} \\ c_{\text{pp,t}} &\leqslant P^{\text{tot}}_{\text{pp}} \quad \forall t \in \{0, 1, ..., \bar{t}\} \end{aligned}$$

- **Minimum monthly production of reservoirs**: All monthly productions $p_{pp,m} = \sum_{t \in T_{pp,m}} p_{pp,t}$ must be greater or equal a certain share σ_{pp} of the maximum monthly inflow $i_{pp,m,\max} = \max_m\{i_{pp,m}\}$, with the monthly sums $i_{pp,m} = \sum_{t \in T_m} i_{pp,t}$ of the inflow. T_m denotes the subset of time slots in month m. This approximates operating constraints due to must-flow conditions.

$$p_{pp,m,\min} \leqslant \sigma_{pp} i_{pp,m,\max}$$

- **Minimum filling levels of reservoirs**: From the reported data on the filling levels in reservoirs it is evident that they never fall below a certain minimum (most notably: Italy). A certain minimum is defined for each reservoir hydro power plant which decreases the effective storage energy capacity.

$$e_{pp,t} \geqslant e_{pp,\min}$$

3.5 Modeling of run-of-river plants

- **Weekly power output**: Run-of-river plants $pp \in P_{\text{ror}}$ are a separate class of power plants with distinct operation constraints. The weekly production $p_{pp,w}$ must add up to the exogenous weekly inflow $i_{pp,w}$:

$$p_{pp,w} \leqslant i_{pp,w} \quad \forall w \in W,\, pp \in P_{\text{ror}}$$

- **Must-flow conditions**: Within each week the power plants are dispatchable, except for an 80% base load component which approximates the rather restrictive must-flow conditions of this power plant class. With the number of hours (weight) w_w of week w:

$$p_{pp,w} \cdot 0.8/w_w \leqslant p_{t,pp} \quad \forall t \in \{0, ..., \bar{t}\}$$

3.6 Short description of objective function

The objective function to be minimized is the sum over all costs included in the model:

- **Fuel costs** of each power plant pp, proportional to the total power plant output $\sum_t p_{pp,t}$
- **CO_2 emission costs** of each power plant pp, proportional to the total power plant output $\sum_t p_{pp,t}$; note that this does not consider CO_2 emissions due to fuels being burned during the off-line phase of the start-up and shutdown process.
- **Variable operation and maintenance costs** of each power plant pp, proportional to the total power plant output $\sum_t p_{pp,t}$
- **Cost of flexible demand** (curtailment) proportional to the total sum $\sum_t d_t^{\text{flx}}$
- **Ramping costs** for each plant pp proportional to the total absolute power output change $\sum_t |\Delta p_{pp,t}|$
- **Fixed operation and maintenance costs** for each plant proportional to the total installed power capacity $\bar{p}_{pp}^{\text{tot}}$
- **Investment costs for new power plant capacity** for each plant, proportional to the newly installed power capacity $\bar{p}_{pp}^{\text{add}}$