Instructions PSS/E Nordics model

Setup

Running the model

- Setup in PyCharm
- Generating the model through Anaconda prompt

Structure of the code

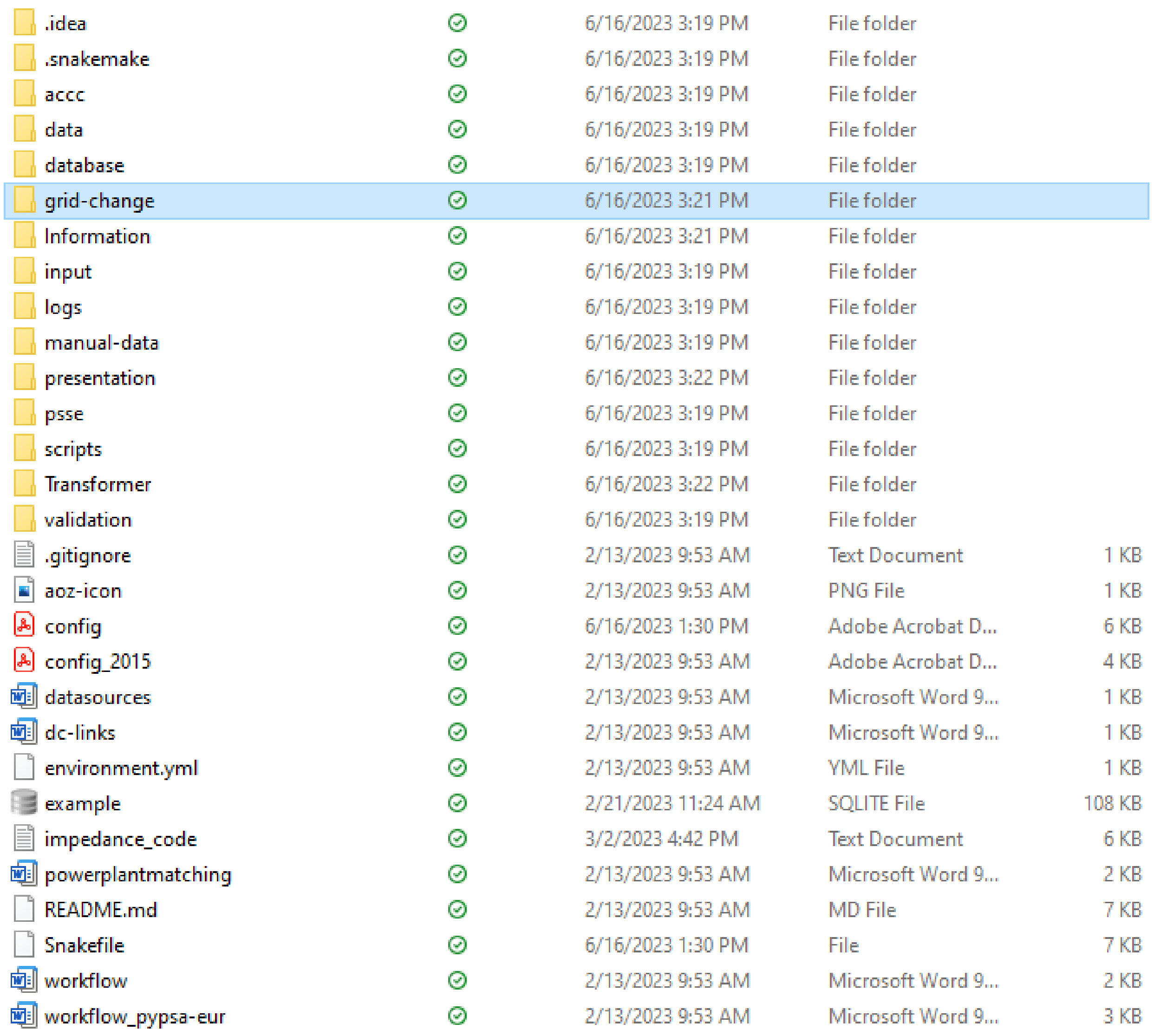
Changes to the model:

- Change line impedances
- Add new lines
- Add new lines for specific cases in the 2045 model
- Remove lines
- Add new buses and bus names
- Remove buses
- Add generators
- Change q_set on generators
- Remove generators
- Add and remove loads
- Add shunts

Input load and generation data:

- Generation data
- Load data
- Cross-border-flow/interconnectors
- FI-SE3
- Sydvästlänken

Below is an image of the folder structure, it is the same for 2020 and 2045



.idea	✓	6/16/2023 3:19 PM	File folder	
.snakemake	✓	6/16/2023 3:19 PM	File folder	
accc	✓	6/16/2023 3:19 PM	File folder	
data	✓	6/16/2023 3:19 PM	File folder	
database	✓	6/16/2023 3:19 PM	File folder	
grid-change	✓	6/16/2023 3:21 PM	File folder	
Information	✓	6/16/2023 3:21 PM	File folder	
input	✓	6/16/2023 3:19 PM	File folder	
logs	✓	6/16/2023 3:19 PM	File folder	
manual-data	✓	6/16/2023 3:19 PM	File folder	
presentation	✓	6/16/2023 3:22 PM	File folder	
psse	✓	6/16/2023 3:19 PM	File folder	
scripts	✓	6/16/2023 3:19 PM	File folder	
Transformer	✓	6/16/2023 3:22 PM	File folder	
validation	✓	6/16/2023 3:19 PM	File folder	
.gitignore	✓	2/13/2023 9:53 AM	Text Document	1 KB
aoz-icon	✓	2/13/2023 9:53 AM	PNG File	1 KB
config	✓	6/16/2023 1:30 PM	Adobe Acrobat D...	6 KB
config_2015	✓	2/13/2023 9:53 AM	Adobe Acrobat D...	4 KB
datasources	✓	2/13/2023 9:53 AM	Microsoft Word 9...	1 KB
dc-links	✓	2/13/2023 9:53 AM	Microsoft Word 9...	1 KB
environment.yml	✓	2/13/2023 9:53 AM	YML File	1 KB
example	✓	2/21/2023 11:24 AM	SQLITE File	108 KB
impedance_code	✓	3/2/2023 4:42 PM	Text Document	6 KB
powerplantmatching	✓	2/13/2023 9:53 AM	Microsoft Word 9...	2 KB
README.md	✓	2/13/2023 9:53 AM	MD File	7 KB
Snakefile	✓	6/16/2023 1:30 PM	File	7 KB
workflow	✓	2/13/2023 9:53 AM	Microsoft Word 9...	2 KB
workflow_pypsa-eur	✓	2/13/2023 9:53 AM	Microsoft Word 9...	3 KB

List of files

Manual data:

- For 2020: Case
  - Buses: Add buses manually
  - Generators: Add generators manually, *Will not be included in scaling*
  - Inactive-lines: Deactivate specific lines
  - Lines: Add lines
  - Loads: Add loads manually, *Will not be included in scaling*
  - Shunts: Add shunts manually
- For 2045: Case
  - Case lines: Manually added lines needed to avoid overloads for example,

    - Generators: Add generators manually, *Will not be included in scaling*
    - Generators_case: Manually added generation units, will be scaled.
    - Loads: Add loads manually, *Will not be included in scaling*

Database:

- Case
    - Nordics: Working database, includes all necessary information to build PSS/E model
    - Nordics_raw: Database with all available information, includes information that is not used to build PSS/E model

Data:

- Case
    - Actual generation: Data on generation from ENTSO-E / or manually input
    - Cross border flow: Data on cross border flow from ENTSO-E / or manually input
    - Installed capacity: Data on installed capacity from ENTSO-E / or manually input
    - Load: Data on load from ENTSO-E / or manually input

Psse

- Case
    - Nordics.loc: Contains geo-data of all buses
    - Nordics.sav
    - Nordics.sld

Input: For information on generating new elec.nc files, watch installation PyPSA-Eur video.

- 2015
    - Elec.nc: the input file gathered from PyPSA-Eur: Only use for 2015 cases
- 2016
    - Elec.nc: the input file gathered from PyPSA-Eur: Only use for 2016 cases
- 2017
    - Elec.nc: the input file gathered from PyPSA-Eur: Only use for 2017 cases
- 2020
    - Elec.nc: the input file gathered from PyPSA-Eur: Use for 2020 and future cases

Scripts

- Contains all scripts used to run the model

Transformer

- Transformer_list: List of transformers used if one wants to move load from one side of a transformer to the other.

Validation

- Case
    - Various validation files comparing model data to official data

Grid-change

- 2020
    - remove_lines: removes lines in all cases run
    - buses: Adds buses for all cases run
    - bus-names: Adds names to all buses
    - lines: Adds lines in all cases run
    - load_rescaling: Lists the buses and the bidding -zone factor which loads are assigned to: more information in *Information/load per region - calculation*
    - remove-bus: removes buses for all cases
    - scaled-generators: For adding generation units that are supposed to be scaled
    - shunts: Adds shunts
- 2045
    - buses: Adds buses for all cases run
    - bus-names: Adds names to all buses
    - lines: Adds lines in all cases run
    - remove_lines: removes lines in all cases run
    - remove_transformers: removes transformers in all cases run
    - remove_bus: removes buses for all cases
    - shunts: Adds shunts

Setup

- Download anaconda @ Installation — conda 23.5.1.dev43 documentation

- There are two versions of the model. The 2020 model and the 2045 model. They can be found on Github using the links below.

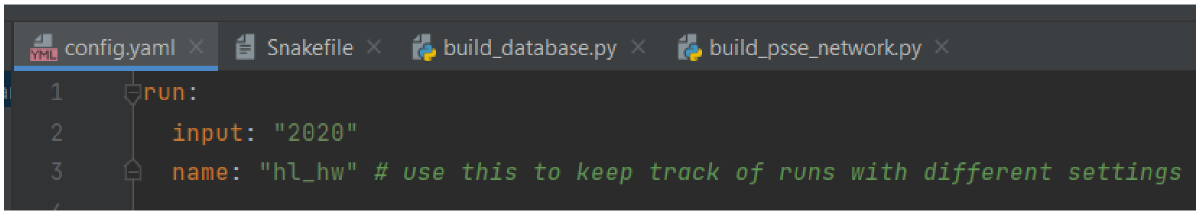
- Follow instructions given in readme @ matildaarvidsson/2045-nordic-grid-model (github.com)
- matildaarvidsson/2020-nordic-grid-model (github.com)

Running the model

In order to implement any changes to the model that you want to save, you need to add them in the code (or the files that are inputs to the code). How this is done is described in *Changes to the model* in this document. You also want to run the code if you want to try new generation or load scenarios. Every time the code is run it creates a new database, a new .sav file and a new .sld file – hence nothing implemented directly into PSS/E will remain if you run the code again.

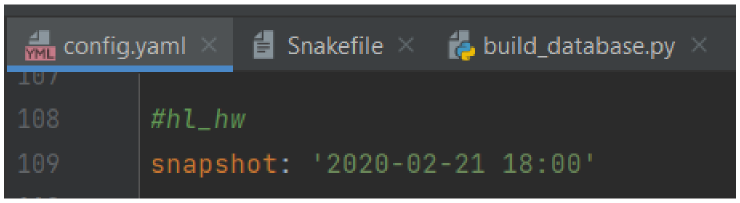
Setup in PyCharm:

- Choose a timestamp to simulate
- Define a name for the timestamp that will be simulated and write it in config.yaml→run:→name:



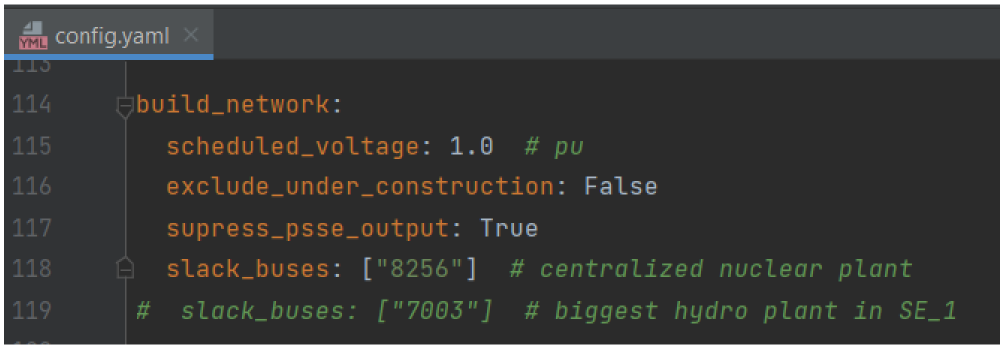
```
run:
  input: "2020"
  name: "hl_hw" # use this to keep track of runs with different settings
```

  This name is used by the script to navigate between folders and to save the resulting files. You need to create folders as described in *Structure of the code*, below.
- Make sure that the manual changes are saved in a folder that matches the name in the config file
- Set the snapshot in the config file to the chosen date and time



```
#hl_hw
snapshot: '2020-02-21 18:00'
```

- Choose slack bus and use it as input in config.yaml→build_network:→slack_buses

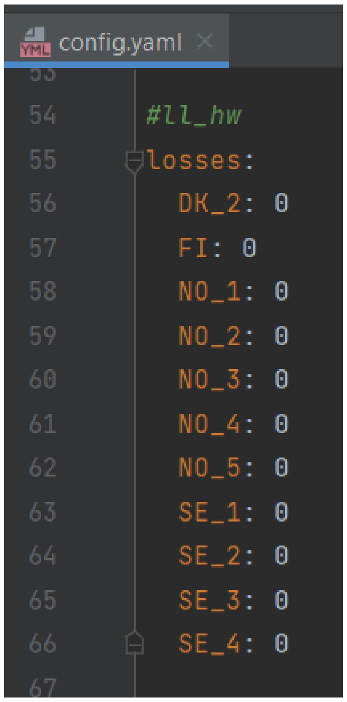


```
build_network:
  scheduled_voltage: 1.0  # pu
  exclude_under_construction: False
  supress_psse_output: True
  slack_buses: ["8256"]  # centralized nuclear plant
#  slack_buses: ["7003"]  # biggest hydro plant in SE_1
```

  Multiple slack buses can be chosen, if that is desired they are added as per following : slack_buses: ["8256", "9006"] – the bus number within citation marks, separated by a comma.

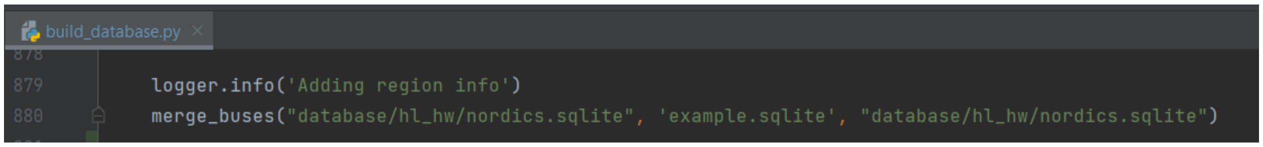
NOTE: At least one of the swing-buses needs to come from the py-psa-eur model, of the existing buses in the model, that is any bus number that do not start with a 9.

- Since the load data includes losses, the (estimated) losses need to be subtracted from the load per bidding zone. To get a good approximation of the losses, the configuration needs to be set by making an initial guess for the losses per bidding zone, if ideas about initial guesses exist, 0 can be used. Then run the code building the PSS/E model and solve the load flow to get the actual losses (in PSS/E: power flow -> Reports -> area/zone totals). By updating the guessed losses with the resulting losses iteratively, it will converge to the right values in about 3 iterations.



```yaml
#ll_hw
losses:
  DK_2: 0
  FI: 0
  NO_1: 0
  NO_2: 0
  NO_3: 0
  NO_4: 0
  NO_5: 0
  SE_1: 0
  SE_2: 0
  SE_3: 0
  SE_4: 0
```

- One must also make sure that the line shown in the figure below is updated with the right case name as defined in the config.yaml file explained above, or the code will not run. This is done in *build_database.py,* the bottom row. Change hl_hw to whatever case name you have chosen.



```python
    logger.info('Adding region info')
    merge_buses("database/hl_hw/nordics.sqlite", 'example.sqlite', "database/hl_hw/nordics.sqlite")
```

Generating the model in Anaconda prompt

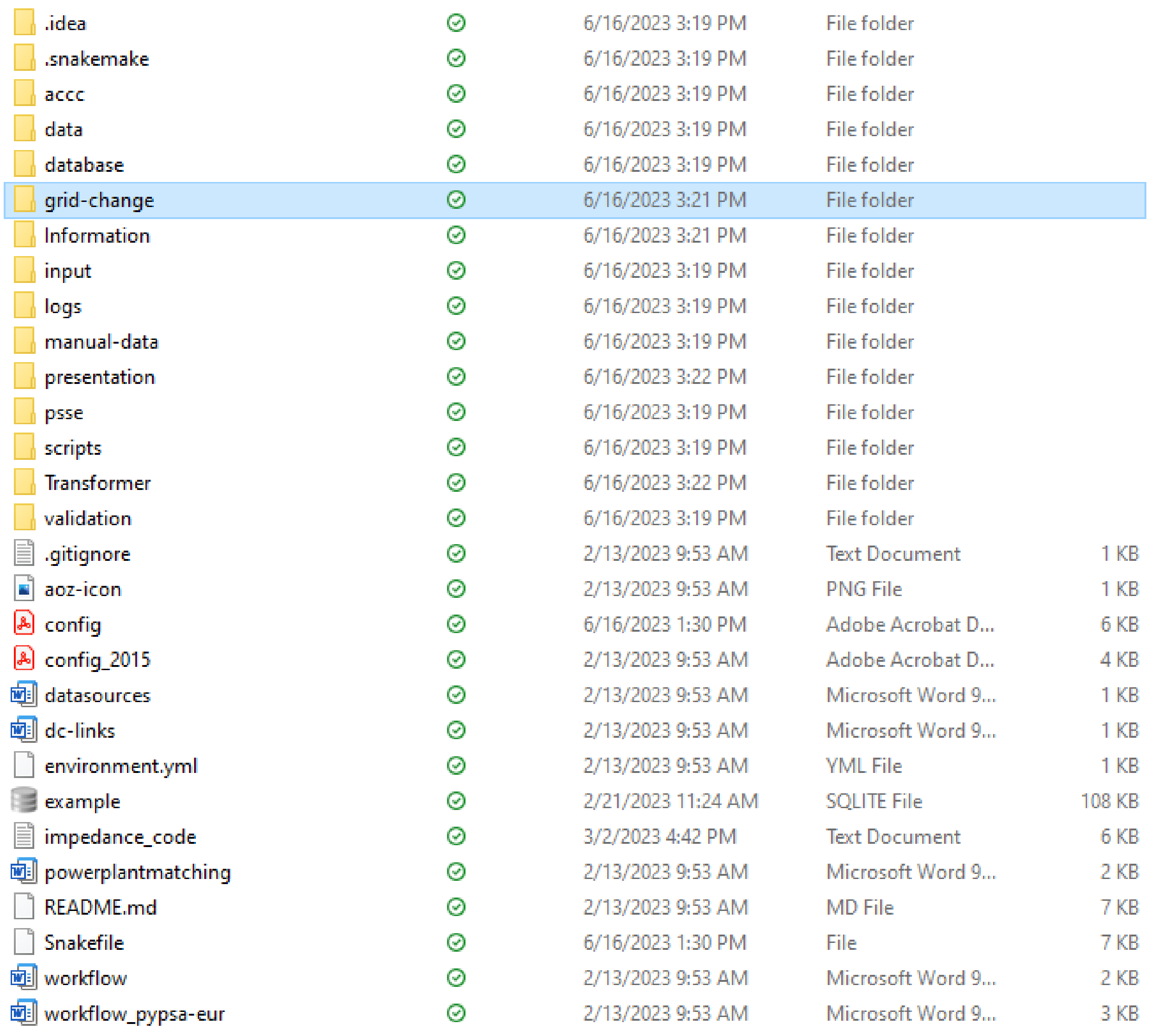
- Open anaconda prompt
- Write “cd <directory where the scripts are saved>”

(example: cd C:\Users\arvkla\Desktop\Model\2045\renewables)

- Write “conda activate dnv-nordics-model”
  This makes sure that all necessary packages are loaded
- Write “snakemake -c all”
  This runs all scripts and produces the model
- The PSS/E files are found in the *psse/case-name* folder

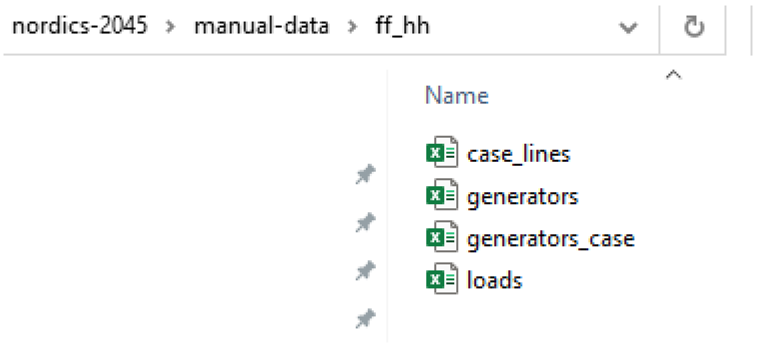
Structure of the code

The folder referred to are found here:



.idea	✓	6/16/2023 3:19 PM	File folder	
.snakemake	✓	6/16/2023 3:19 PM	File folder	
accc	✓	6/16/2023 3:19 PM	File folder	
data	✓	6/16/2023 3:19 PM	File folder	
database	✓	6/16/2023 3:19 PM	File folder	
grid-change	✓	6/16/2023 3:21 PM	File folder	
Information	✓	6/16/2023 3:21 PM	File folder	
input	✓	6/16/2023 3:19 PM	File folder	
logs	✓	6/16/2023 3:19 PM	File folder	
manual-data	✓	6/16/2023 3:19 PM	File folder	
presentation	✓	6/16/2023 3:22 PM	File folder	
psse	✓	6/16/2023 3:19 PM	File folder	
scripts	✓	6/16/2023 3:19 PM	File folder	
Transformer	✓	6/16/2023 3:22 PM	File folder	
validation	✓	6/16/2023 3:19 PM	File folder	
.gitignore	✓	2/13/2023 9:53 AM	Text Document	1 KB
aoz-icon	✓	2/13/2023 9:53 AM	PNG File	1 KB
config	✓	6/16/2023 1:30 PM	Adobe Acrobat D...	6 KB
config_2015	✓	2/13/2023 9:53 AM	Adobe Acrobat D...	4 KB
datasources	✓	2/13/2023 9:53 AM	Microsoft Word 9...	1 KB
dc-links	✓	2/13/2023 9:53 AM	Microsoft Word 9...	1 KB
environment.yml	✓	2/13/2023 9:53 AM	YML File	1 KB
example	✓	2/21/2023 11:24 AM	SQLITE File	108 KB
impedance_code	✓	3/2/2023 4:42 PM	Text Document	6 KB
powerplantmatching	✓	2/13/2023 9:53 AM	Microsoft Word 9...	2 KB
README.md	✓	2/13/2023 9:53 AM	MD File	7 KB
Snakefile	✓	6/16/2023 1:30 PM	File	7 KB
workflow	✓	2/13/2023 9:53 AM	Microsoft Word 9...	2 KB
workflow_pypsa-eur	✓	2/13/2023 9:53 AM	Microsoft Word 9...	3 KB

The code is structured in such a way that there are different scripts running different parts of the model, using different inputs and creating different outputs. When a new case is created, new empty folders with the case name needs to be added in the *data, database, logs, psse,* and *validation* folders. When the case name is defined in the config file, the code will find these folders automatically. A new case folder is also needed in the *manual-data* folder, but it needs to include three files; *generators*, *loads* and *inactive-lines* for the 2020 model, and *case_lines, generators*, *generators_case* and *loads* for the 2045 model.


nordics-2045 › manual-data › ff_hh

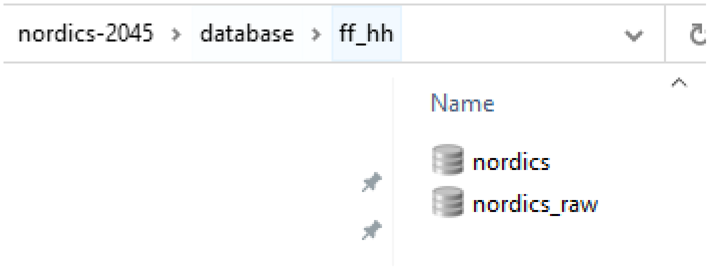
Name

case_lines
generators
generators_case
loads

They can be copied from the *empty files* folder in the *manual-data* folder.

If old data remains in output files with unchanged file names, these will be replaced when rerunning the code. To save them, change the name.

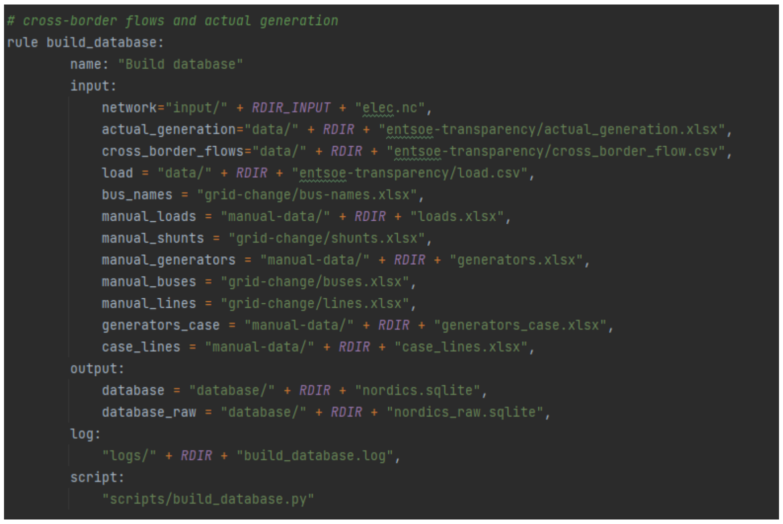
Also, the code notices where changes have been made, so when running the code, only the necessary steps will be rerun. However, this does not include changes in the *config.yaml* file, in order to rerun all the necessary steps, the fastest way is to delete the database in the corresponding case file. Mark both database files and press delete.


nordics-2045 › database › ff_hh

Name

nordics
nordics_raw

In the snakefile all the steps are defined, it can be open in PyCharm by pressing it. In the snakefile the input going into each script is read, and the output defined. An example is shown below:

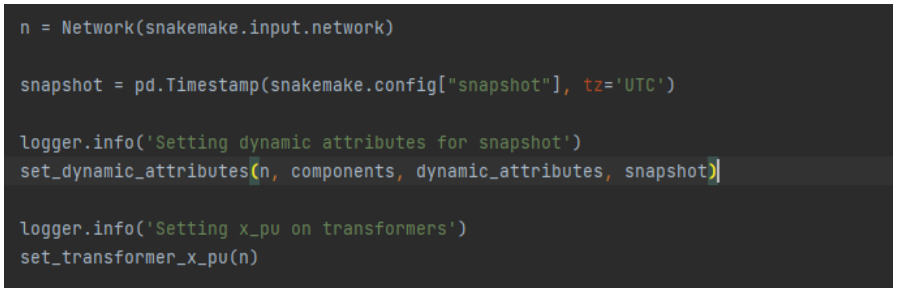


```
# cross-border flows and actual generation
rule build_database:
        name: "Build database"
        input:
            network="input/" + RDIR_INPUT + "elec.nc",
            actual_generation="data/" + RDIR + "entsoe-transparency/actual_generation.xlsx",
            cross_border_flows="data/" + RDIR + "entsoe-transparency/cross_border_flow.csv",
            load = "data/" + RDIR + "entsoe-transparency/load.csv",
            bus_names = "grid-change/bus-names.xlsx",
            manual_loads = "manual-data/" + RDIR + "loads.xlsx",
            manual_shunts = "grid-change/shunts.xlsx",
            manual_generators = "manual-data/" + RDIR + "generators.xlsx",
            manual_buses = "grid-change/buses.xlsx",
            manual_lines = "grid-change/lines.xlsx",
            generators_case = "manual-data/" + RDIR + "generators_case.xlsx",
            case_lines = "manual-data/" + RDIR + "case_lines.xlsx",
        output:
            database = "database/" + RDIR + "nordics.sqlite",
            database_raw = "database/" + RDIR + "nordics_raw.sqlite",
        log:
            "logs/" + RDIR + "build_database.log",
        script:
            "scripts/build_database.py"
```

Then there are different scripts doing different things. Most changes are done in either the *build_database.py* file, or *the build_psse_network.py* file. The *build_database.py* file builds the database. It combines the information from the PyPSA project using the *elec.nc* input file, with information that is added manually in the code or through excel files, and outputs a database with information on all the components going into the model. Then the PSS/E model is built based on the information in the database.

As for the scripts, they are run by having code lines call the definitions. The code does not go through the definitions top to bottom, but rather the lines that call the definitions in the same script. These are placed in the bottom of each script.

In the figure below, two lines that call definitions (set_dynamic_attribues, set_transformer_x_pu) are shown. The definitions are found in the same script, but above.



```
n = Network(snakemake.input.network)

snapshot = pd.Timestamp(snakemake.config["snapshot"], tz='UTC')

logger.info('Setting dynamic attributes for snapshot')
set_dynamic_attributes(n, components, dynamic_attributes, snapshot)

logger.info('Setting x_pu on transformers')
set_transformer_x_pu(n)
```

The input file elec.nc is the output from running the PyPSA-eur model created by Niek. A link to that GitHub is provided. We do, however, have very limited knowledge about this code, but that is where the input of the model when it comes to the topology and generators comes from.

Changes to the model

When implementing changes to the model, most changes that you would want to implement can be implemented through adding or removing components in different excel sheets. What excel file controls what component is described in this section for each component.

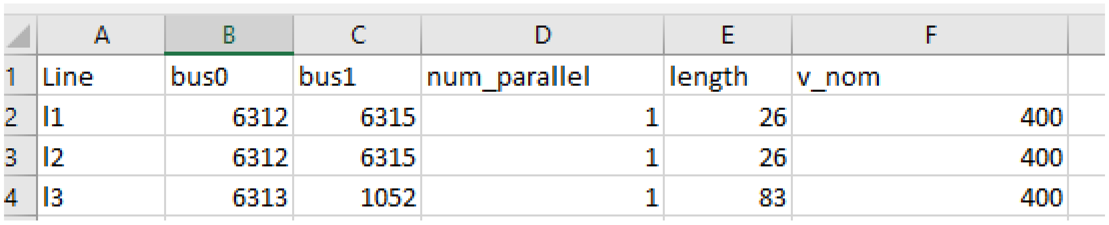
There are two different folders where changes to the grid can be implemented. In the excel files in the folder *grid-change* you implement changes to the model that is always included when running the code, and in the excel sheets in the *manual-data* folder it is possible to implement changes that are specific for the one case you are running.

Overall, when information in an excel file is used in the model, the excel file is first read in the Snakefile, where it is also defined as input for the different scripts.

The first step is to build the database, it is an sql database and is available in the database folder. The second step is to build the PSS/E network, and the third step is to build the slider.

- **Change the impedance of lines:** The impedance of the lines is decided based on voltage level and length. To change these values, you need to go into the *build_database.py* file and find the definition: *def set_line_impedance_from_linetypes*. Here r, x, b and g per length is input and it recalculates it to p.u. values. Also, s_nom, and s_nom2 is decided here. With the current configuration, no individual changes can be made to specific lines, this would have to be implemented by new code if it is desirable.

- **Adding new lines**: In the folder *grid-change* there is a file called *lines*. Here lines are added by filling in a line name, which buses it goes between, number of parallel lines, the length and the voltage of the line. This adds lines to the database.



	A	B	C	D	E	F
1	Line	bus0	bus1	num_parallel	length	v_nom
2	l1	6312	6315	1	26	400
3	l2	6312	6315	1	26	400
4	l3	6313	1052	1	83	400

  All the lines added here get the features (impedance, resistance, rated power) decided in *def set_line_impedance_from_linetypes* in the *build_database.py* script.

  All lines are added to PSSE/E in the *build_psse_network.py* file in the definition: *def add_branch*.

- **Adding specific lines for different cases in the 2045 model:** If lines need to be added for specific cases in the 2045 model, the excel file *case-lines* in *manual-data* can be used. If a line is added in that file, it is only included for that specific case.

- **Removing lines:** There are two different was of removing lines, the first one is to completely remove a line, and the other is to deactivate a line.

  To remove a line, the excel file *remove_lines* in the *grid-change* folder is used. Here the bus numbers (in the correct order as seen in PSS/E) is input, as well as the branch id. This completely removes the line.

  To deactivate a line, if it's out of commission for instance, the excel file *inactive-lines* in the *manual-data* folder can be used in the 2020 model. This only deactivates the line for the specific case that is run, and the input is bus numbers (in the correct order as seen in PSS/E), as well as the branch id.

  The removing and deactivating of lines is implemented automatically in the definition: *def remove_branch_and_buses* in the *build_psse_network.py* file.

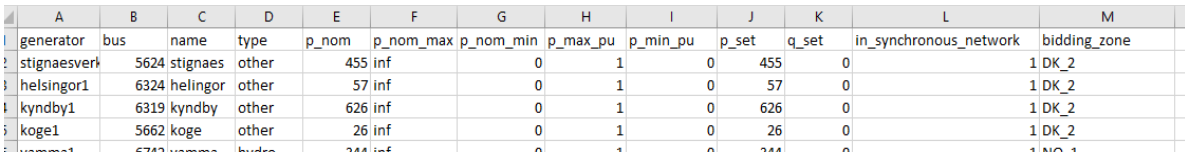
- **Add buses and bus names:** To add new buses they should be added in the *buses* file in the *grid-change* folder. There the name of the bus is added, the bus number, the coordinates, the country and the bidding zone. To find the coordinates its recommended to use the site GPS coordinates, latitude and longitude with interactive Maps (gps-coordinates.net).

  To change the names of buses already in the model, check the excel file *bus-names* in the *grid-change* folder.

- **Remove buses:** To remove a bus the bus number is fed into the *remove-bus* excel file in the *grid-change* folder. The change is implemented in the definition: *def remove_branch_and_buses* in the *build_psse_network.py* file.

- **Add generators:** Generators can be added in two different ways, either a generator is added to the "total" collection of generators and uses the model's way of assigning P_out, or it is added after P_out has been set for the other generators and P_out is assigned manually.

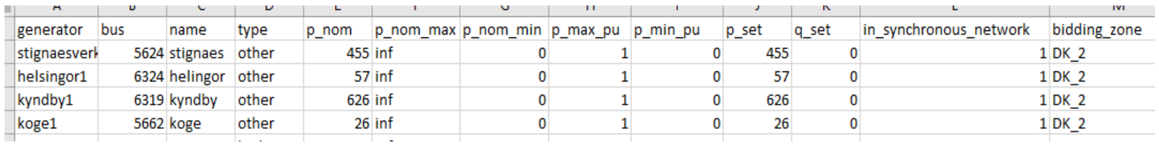
  In the model, P_out is set by generation type and installed capacity in the relevant bidding zone. By calculating the total installed capacity of a certain generation type in a bidding zone and comparing that to the specified generation in a bidding zone, a percentage of the installed capacity is set for all generators of the specific type for each region. To include the generator in this calculation, it must be added in the *scaled-generators* file in the *grid-change* folder in the 2020 model.



generator	bus	name	type	p_nom	p_nom_max	p_nom_min	p_max_pu	p_min_pu	p_set	q_set	in_synchronous_network	bidding_zone
stignaesverk	5624	stignaes	other	455	inf	0	1	0	455	0	1	DK_2
helsingor1	6324	helingor	other	57	inf	0	1	0	57	0	1	DK_2
kyndby1	6319	kyndby	other	626	inf	0	1	0	626	0	1	DK_2
koge1	5662	koge	other	26	inf	0	1	0	26	0	1	DK_2

In the 2045 model, to include generators that should be included when calculating p_set they need to be added in the *generators-case* excel file in the *manual-data* folder. This file includes the generators for the different scenarios, and hence are different between the different cases. Adding a generator in this file will make sure it is included in the scaling. Here too, it is important to make sure that p_set = p_nom when adding it.

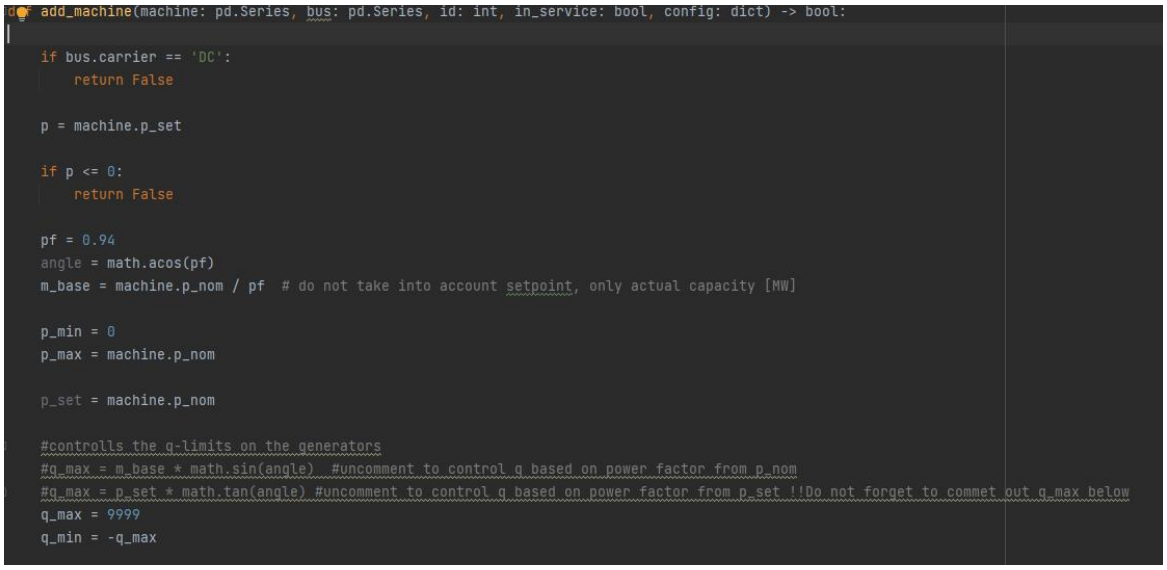
When adding generators to the *scaled-generators* file, a name of the generator is needed, this name must be unique for each generator added. The bus number it's assigned to, the type of the generator (hydro, solar, other, nuclear, wind-onshore, wind-offshore, no other types can be used). p_nom and p_set should be given the same values, which is the maximum capacity of the generator added. p_nom is the installed capacity shown in the PSS/E model, and p_set is the value that will be scaled; hence they should be the same. Which bidding zone it belongs to is also assigned here. q_set is later recalculated independent of what is written in this column. The columns p_nom_max, p_nom_min, p_max_pu, and in_synchrounous_network always have the same values for all generators input in the model, but these can be changed. However, generators will not be counted if the in_synchrounous_network column is not 1. The settings are shown in the figure below.



generator	bus	name	type	p_nom	p_nom_max	p_nom_min	p_max_pu	p_min_pu	p_set	q_set	in_synchronous_network	bidding_zone
stignaesverk	5624	stignaes	other	455	inf	0	1	0	455	0	1	DK_2
helsingor1	6324	helingor	other	57	inf	0	1	0	57	0	1	DK_2
kyndby1	6319	kyndby	other	626	inf	0	1	0	626	0	1	DK_2
koge1	5662	koge	other	26	inf	0	1	0	26	0	1	DK_2

To add a generator that should not be included in the scaling, and is added additionally with regards to the input of generation in a bidding zone, one should use the *generators* file in the *manual-data* folder. The columns are the same as above, but the value assigned to p_set is the value that the generator will produce in the model. Q will be set in the same way as the other generators.

- **Change q_set on generators:** How much q that the generators are allowed to produce is determined in the definition: *def add_machine* in the *build_psse_network.py* script. If one want to use the grid code requirement as the q limit, use a power factor of 0.94.

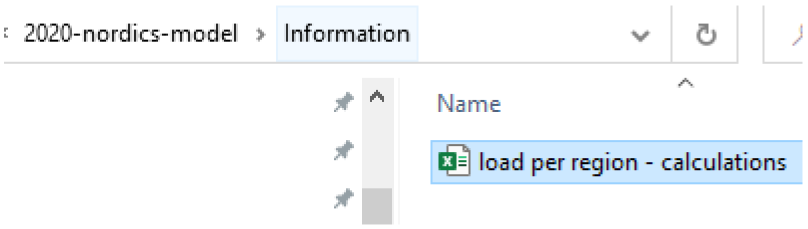


```python
def add_machine(machine: pd.Series, bus: pd.Series, id: int, in_service: bool, config: dict) -> bool:

    if bus.carrier == 'DC':
        return False

    p = machine.p_set

    if p <= 0:
        return False

    pf = 0.94
    angle = math.acos(pf)
    m_base = machine.p_nom / pf  # do not take into account setpoint, only actual capacity [MW]

    p_min = 0
    p_max = machine.p_nom

    p_set = machine.p_nom

    #controlls the q-limits on the generators
    #q_max = m_base * math.sin(angle)  #uncomment to control q based on power factor from p_nom
    #q_max = p_set * math.tan(angle) #uncomment to control q based on power factor from p_set !!Do not forget to commet out q_max below
    q_max = 9999
    q_min = -q_max
```

- **Remove generators:** To remove generators that are already in the model, separate code lines for each generator needs to be used. In the definition: *def scale_generation_per_bidding_zone* in the *build_database.py* file the following line needs to be used: n.generators = n.generators.drop('6937 onwind', axis=0) where in this case the '6937 onwind' is the text in the 'Generator' column in the sql database.

- **Loads 2020:** Loads are either added on a gdp/population basis, or on a regional level. To add it on a regional level the definition: *def scale_load_per_region* is used.

  How this is done, and how changes are made to this configuration is described in the *information* sheet in the *load per region – calculations* excel file located in the *Information* folder.



  It is also easiest to perform the calculations in the sheet *input load per region* in the same excel file and then updating the *buses* sheet in the same file. Adding a bus here will add a load in the model, and removing a bus will remove a load. Note that it is the file *load_rescaling* in the *grid-change* folder that is used as input to the model, so it is important to update that after performing the calculations.

  Just like with the generators, there is also a possibility to add loads that are not scaled, but added **additionally**. If one wants to do this, the load should be added in the *loads* file in the *manual-data* folder. Here, if you enter p_set that is that value that the load will take in the model as well, and it is added additionally to the loading of the bidding zone. The loads are then added in the definition: *def add_manual_loads* in *build_database.py*.

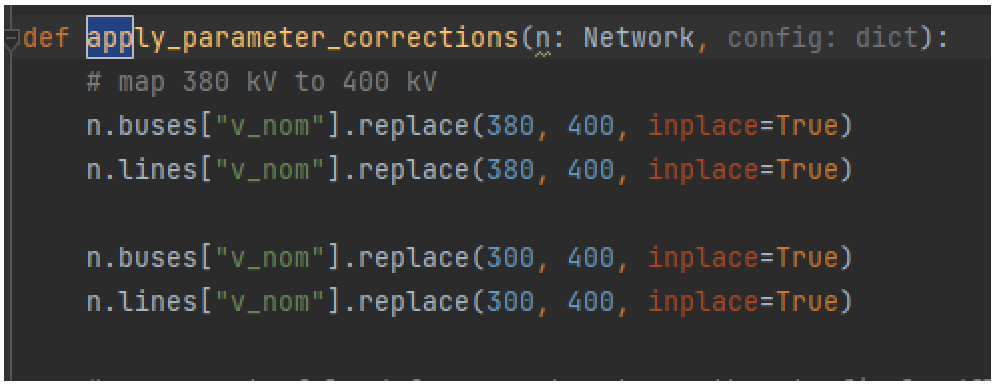
The loads are added to PSS/E in the definition: *def add_load_psse* in *build_psse_network.py*

- **Loads 2045**: The 2045 model uses the PyPSA way of dividing the loads which is dependent to the Voronoi cells and 60 % gdp and 40 % within these. This is done in the definition: *def scale_loads_per_bidding_zone*. There is no preprogrammed way to add loads here to include them in the distribution of the loads.

  Loads added needs to be added manually, **additional** to the load input per bidding zone. This is done by adding them in the *loads* file in the *manual-data* folder.

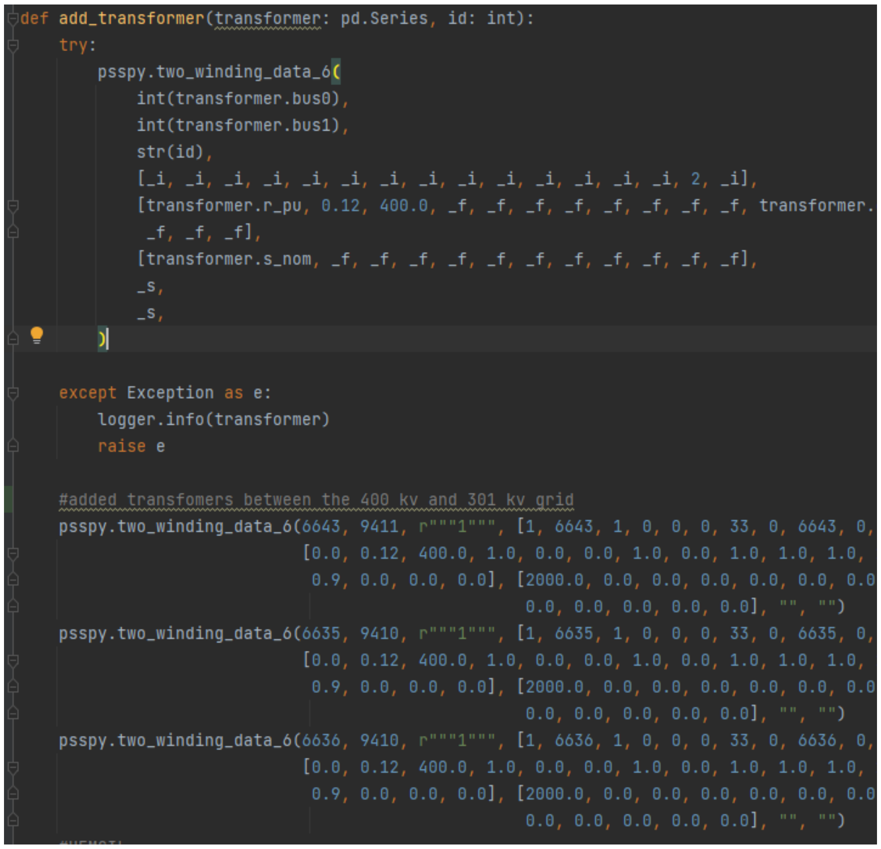
- **Shunts:** No shunts are added to the model to start with, but shunts can be added by adding the relevant values in the *shunts* file in the *grid-change* folder. The shunts are added to the database in the definition *def add_manual_shunts* in *build_database.py.* The shunts are added to PSS/E in the definition: *def add_shunt_impedance* in *build_psse_network.py.*

- **Note:** If one wants to add buses to the 300 kV grid in the 2045 model, the voltage level here is set to 301 kV. This is because in definition*: def apply_parameter_corrections* everything with voltage level 300 kV is changed to 400 kV.



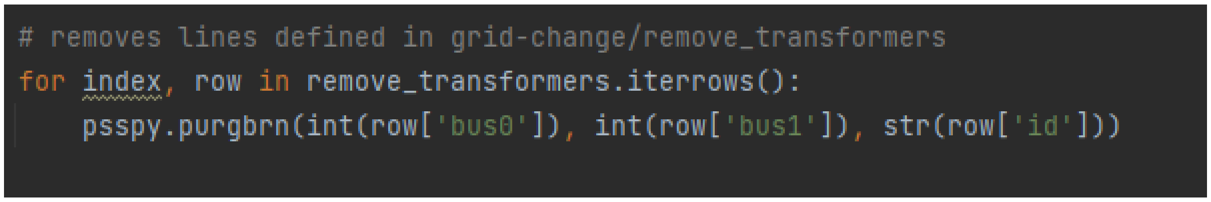
```
def apply_parameter_corrections(n: Network, config: dict):
    # map 380 kV to 400 kV
    n.buses["v_nom"].replace(380, 400, inplace=True)
    n.lines["v_nom"].replace(380, 400, inplace=True)

    n.buses["v_nom"].replace(300, 400, inplace=True)
    n.lines["v_nom"].replace(300, 400, inplace=True)
```

Comment this out if this change is undesired, but keep in mind that the manually added 301 kv lines, buses and transformers will need to be removed in this case. I would use ctrl+f in the excel files *grid-change/lines* and *grid-change/buses* to find and remove these. To remove the transformers on these buses the manually added transformers in the definition*: def add_transformers* in the *build_psse_network-py* script needs to removed or commented out.



```
def add_transformer(transformer: pd.Series, id: int):
    try:
        psspy.two_winding_data_6(
            int(transformer.bus0),
            int(transformer.bus1),
            str(id),
            [_i, _i, _i, _i, _i, _i, _i, _i, _i, _i, _i, _i, _i, _i, 2, _i],
            [transformer.r_pu, 0.12, 400.0, _f, _f, _f, _f, _f, _f, _f, _f, transformer.
             _f, _f, _f],
            [transformer.s_nom, _f, _f, _f, _f, _f, _f, _f, _f, _f, _f, _f],
            _s,
            _s,
        )

    except Exception as e:
        logger.info(transformer)
        raise e

    #added transfomers between the 400 kv and 301 kv grid
    psspy.two_winding_data_6(6643, 9411, r"""1""", [1, 6643, 1, 0, 0, 0, 33, 0, 6643, 0,
                             [0.0, 0.12, 400.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0,
                              0.9, 0.0, 0.0, 0.0], [2000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
                                                    0.0, 0.0, 0.0, 0.0, 0.0], "", "")
    psspy.two_winding_data_6(6635, 9410, r"""1""", [1, 6635, 1, 0, 0, 0, 33, 0, 6635, 0,
                             [0.0, 0.12, 400.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0,
                              0.9, 0.0, 0.0, 0.0], [2000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
                                                    0.0, 0.0, 0.0, 0.0, 0.0], "", "")
    psspy.two_winding_data_6(6636, 9410, r"""1""", [1, 6636, 1, 0, 0, 0, 33, 0, 6636, 0,
                             [0.0, 0.12, 400.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0,
                              0.9, 0.0, 0.0, 0.0], [2000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0
                                                    0.0, 0.0, 0.0, 0.0, 0.0], "", "")
```

Also, in the 2045 model, the 400/300 kV tranformers are removed, since the buses have an upgraded voltage, this is done in the definition *def remove_branch* in the *build_psse_network.py* script. The excel sheet defining these is *remove-transformers* in the *grid-change* folder.
If the change in voltage from 300 kV to 400 kV should be removed, the code removing the transformers also needs to be commented out. This is done by commenting out these lines in the definition *def remove_branch* in the *build_psse_network.py* script.



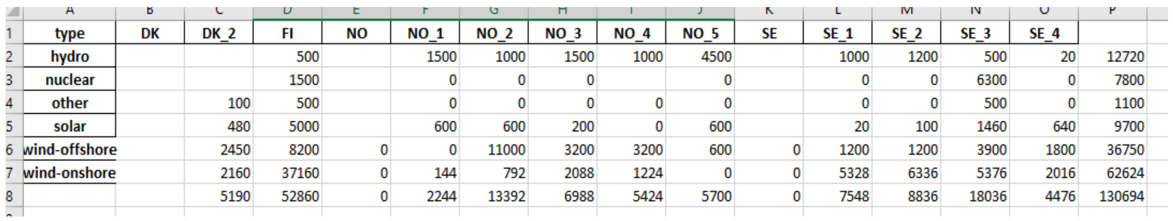
```
# removes lines defined in grid-change/remove_transformers
for index, row in remove_transformers.iterrows():
    psspy.purgbrn(int(row['bus0']), int(row['bus1']), str(row['id']))
```

Input load, generation and cross-border-flow data

The data can either be gathered automatically from ENTSO-E by running a new case and having an empty data folder (for the specific case) (just create an empty *data/"case-name"* folder). It is also possible to manually fill in the load and generation data for each bidding zone, if this is desired it is recommended to copy the files form any other *data/"case-name"* folder to your *data/"case-name"* folder and manually change the data.

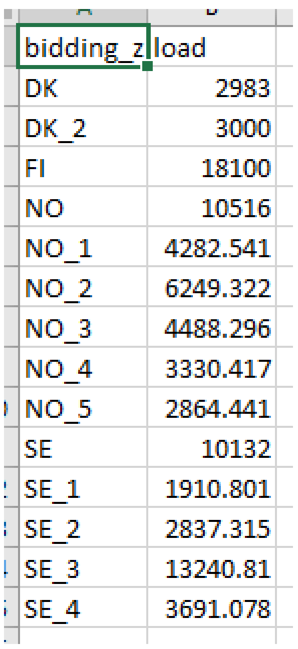
- **Generation data:** The generation data is input based on generation type and bidding zone. On ENTSO-E transparency platform all data is available on bidding zone level for Denmark, Finland and Norway, but not for Sweden. If one wants to run on ENTSO-E data, the data for Sweden needs to be gathered elsewhere (mimer.se is a good option, or Svk's yearly excel file available on their website). The data going into the model comes from the *actual_generation* file in *data/"case-file"/entsoe-transparancy* folder. If ENTSO-E data is used, the date needs to be selected before 6th of October 2020. If manual data is used as input, then there is no limitation on date. However, a timestamp during 2020, before 6th of October 2020 needs to be set in the config.yaml file, or else the code will not run. It will not impact the simulation once the data is input.

  The data is retrieved in the *retrieve_actual_generation.py* script.



type	DK	DK_2	FI	NO	NO_1	NO_2	NO_3	NO_4	NO_5	SE	SE_1	SE_2	SE_3	SE_4	
hydro			500		1500	1000	1500	1000	4500		1000	1200	500	20	12720
nuclear			1500		0	0	0		0		0	0	6300	0	7800
other		100	500		0	0	0	0	0		0	0	500	0	1100
solar		480	5000		600	600	200	0	600		20	100	1460	640	9700
wind-offshore		2450	8200	0	0	11000	3200	3200	600	0	1200	1200	3900	1800	36750
wind-onshore		2160	37160	0	144	792	2088	1224	0	0	5328	6336	5376	2016	62624
		5190	52860	0	2244	13392	6988	5424	5700	0	7548	8836	18036	4476	130694

- **Load data:** The load data is gathered in the same way as the generation data, but the file used as input is called *load*. Here there are data on bidding zone level for Sweden on the ENTSO-E transparency platform. Note that if you add manual loads, this is added additionally to these loads.

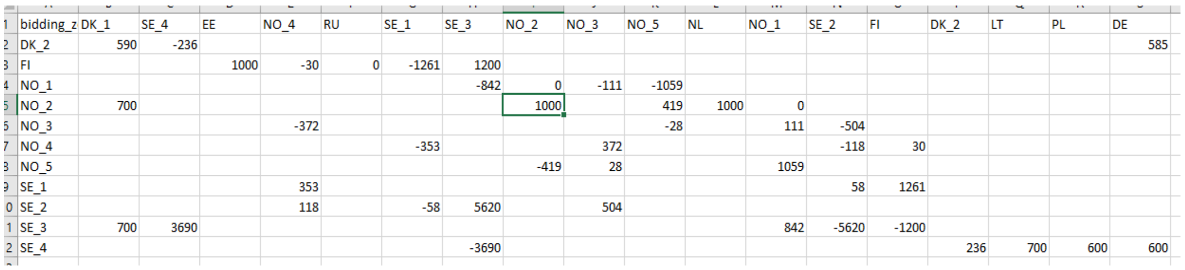


bidding_z	load
DK	2983
DK_2	3000
FI	18100
NO	10516
NO_1	4282.541
NO_2	6249.322
NO_3	4488.296
NO_4	3330.417
NO_5	2864.441
SE	10132
SE_1	1910.801
SE_2	2837.315
SE_3	13240.81
SE_4	3691.078

In the folder *entsoe-transparancy* there is also a file *installed_capacity*. This is not used as any input in the model, only for comparison.

- **Cross-border-flow:** The data on the cross-border-flows are gathered from ENTSO-E in the *retrieve_cross_border_flow.py* file and put in the *cross_border_flow* file in the *data/"case-file"/entsoe-transparancy* folder. These values are used as input if they are from an interconnector, and as comparison if it is between synchronized bidding zones.

  Just like with the load and generation data, this can also be manually added. If one wants to control the NO2-GB interconnector there is no separate column for this, but the data can be added in the NO2xNO2 cell to control the load connected to this link, marked in the figure.

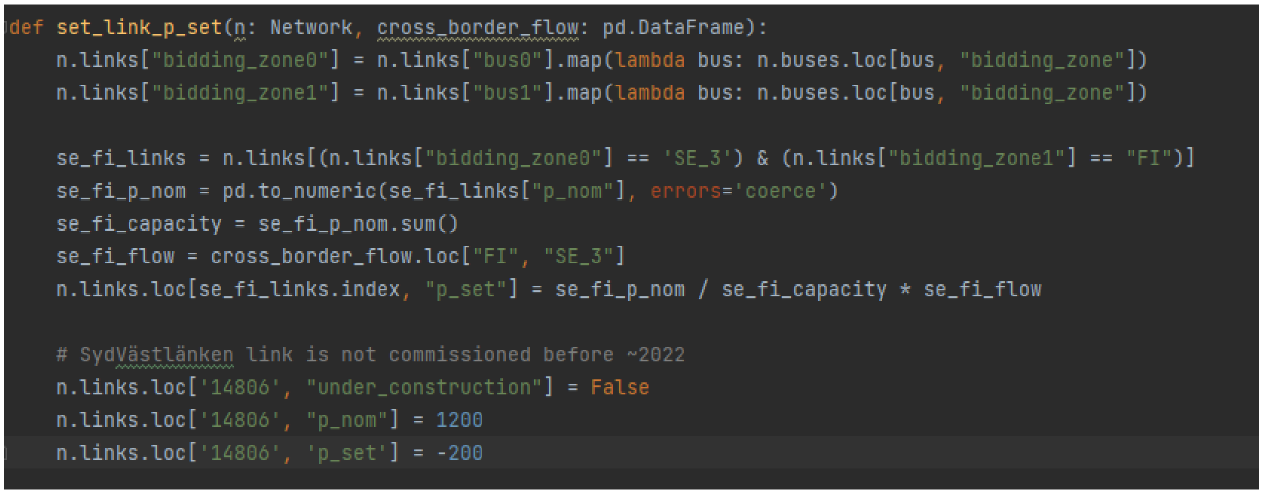


bidding_z	DK_1	SE_4	EE	NO_4	RU	SE_1	SE_3	NO_2	NO_3	NO_5	NL	NO_1	SE_2	FI	DK_2	LT	PL	DE
DK_2	590	-236																585
FI			1000	-30	0	-1261	1200											
NO_1							-842	0	-111	-1059								
NO_2	700							1000		419	1000	0						
NO_3				-372						-28		111	-504					
NO_4						-353			372				-118	30				
NO_5								-419	28			1059						
SE_1				353									58	1261				
SE_2				118		-58	5620		504									
SE_3	700	3690										842	-5620	-1200				
SE_4							-3690								236	700	600	600

The links are added in the definition: *def insert_external_links* in the *build_database.py* file. In cases where there are multiple links between one bidding zone and one country, the flow is divided equally between the links.

- **FI-SE3:** The flow on the FI-SE2 HVDC connection is gathered from the cross-border-flow file and input in the definition: *def set_link_p_set* in the *build_database.py* file. The flow in the two connections is divided by the capacity of the connections.

- **Sydvästlänken:** Whether you want to include Sydvästlänken in the model or not can be controlled in the definition: *def set_link_p_set* in the *build_database.py* file. It is also here the power flow in Sydvästlänken is controlled. There is no way of knowing how much power is flowing through Sydvästlänken since there are other connections there, hence this value needs to be manually updated, maybe by running in iterations to see how the power is flowing. This can easily be tried directly into PSS/E, and then the value chosen can be assigned in the code, in the in the definition: *def set_link_p_set* in the *build_database.py* file.



```
def set_link_p_set(n: Network, cross_border_flow: pd.DataFrame):
    n.links["bidding_zone0"] = n.links["bus0"].map(lambda bus: n.buses.loc[bus, "bidding_zone"])
    n.links["bidding_zone1"] = n.links["bus1"].map(lambda bus: n.buses.loc[bus, "bidding_zone"])

    se_fi_links = n.links[(n.links["bidding_zone0"] == 'SE_3') & (n.links["bidding_zone1"] == "FI")]
    se_fi_p_nom = pd.to_numeric(se_fi_links["p_nom"], errors='coerce')
    se_fi_capacity = se_fi_p_nom.sum()
    se_fi_flow = cross_border_flow.loc["FI", "SE_3"]
    n.links.loc[se_fi_links.index, "p_set"] = se_fi_p_nom / se_fi_capacity * se_fi_flow

    # SydVästlänken link is not commissioned before ~2022
    n.links.loc['14806', "under_construction"] = False
    n.links.loc['14806', "p_nom"] = 1200
    n.links.loc['14806', 'p_set'] = -200
```