plan4res user manual

version 2.0 Feb 14, 2025

Authors	Organisation	Date
Sandrine Charousset	EDF	2025.02.20
Thomas Ouillon	EDF	2023.06.30

Release	Date	Reason for change
1.0	2024.06.10	Initial version
2.0	2025.02.20	Restructuring of documentation New version of plan4res (2.0)

DISCLAIMER / ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 773897

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 835896.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101069694.

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101118123.

Table of Content

1 Quick Install and Run

1.1 Quick Install

- Clone the install repo in a chose directory, e.g. P4R_DIR: *git clone* https://github.com/plan4res/install
- Move the files *.sh from in P4R_DIR, make them executable (*chmod a+x *.sh*)
- Run the install script: *./plan4res_install.sh* with needed options (-S $SOLVER with $SOLVER=CPLEX, GUROBI, SCIP ; else it will use HiGHS ; -I $installer if $SOLVER is CPLEX or GUROBI; -L $LICENCE if $SOLVER is GUROBI; ….)

If you wish to run plan4res from a different directory, e.g P4R_DIR_LOCAL, move the script user_init_plan4res.sh in P4R_DIR_LOCAL and run it as follows: *./user_init_plan4res.sh -D P4R_DIR -S $SOLVER*

1.2 Quick Run

- Create dataset in P4R_DIR_LOCAL/data (e.g. YourDataset), with a subdirectory IAMC, a subdirectory settings and a subdirectory TimeSeries
- Move your IAMC file (as computed by GENeSYS-MOD for instance) in IAMC/ and rename it YourDataset.xlsx
- Copy the settings files from P4R_DIR_LOCAL/data/settings/toyDataset to settings/
- Create or get and Upload your timeseries to TimeSeries
- Edit configuration files in settings:
    - Edit settingsCreatePlan4res_simul.yml and settingsCreatePlan4res_invest.yml (in particular the IAMC scenario and year, the list of regions, the regions partitions, the list of technologies, the list of scenarios, the definition of coupling constraints, the initial filling rates, the (optional) parameters for capacity expansion, and if necessary the list of variables to use from your IAMC file)
    - Edit DictTimeSeries.yml (so that is has the good names for your timeseries)
    - If necessary (if you have changed some variable names in settingsCreatePlan4res_XXX.yml), edit the VariablesDictionnary.yml
- Run CREATE: p4r CREATE YourDataset (with option -M invest if running a capacity expansion case)
- Edit settings_format_optim.yml, settings_format_simul.yml and settings_format_invest.yml configuration files in settings (in particular the sections Calendar, and the Scenarios and ScenarisedData in section ParametersFormat)
- Run SSV: p4r SSV YourDataset
- Run SIM: p4r SIM YourDataset
- Run CEM: p4r CEM YourDataset
- Edit settingsPostTreatPlan4res_simul.yml and settingsPostTreatPlan4res_invest.yml in settings. In particular, the start and end dates of you study, the list of technologies (if you added new technologies), and the size of graphs (to allow to cope with all regions / all interconnections / all technos)
- Run POSTTREAT: p4r POSTTREAT YourDataset (with option -M invest if running a capacity expansion case)

2 Installing Plan4res

The process to install plan4res is described in the following subsections.

In summary:

- Clone the install repo in a chose directory, e.g. P4R_DIR: *git clone* *https://github.com/plan4res/install*
- Move the files *.sh from in P4R_DIR, make them executable (*chmod a+x *.sh*)
- Run the install script: *./plan4res_install.sh* with needed options (-S $SOLVER with $SOLVER=CPLEX, GUROBI, SCIP ; else it will use HiGHS ; -I $installer if $SOLVER is CPLEX or GUROBI; -L $LICENCE if $SOLVER is GUROBI; ….)
- If you wish to run plan4res from a different directory, e.g P4R_DIR_LOCAL, move the script user_init_plan4res.sh in P4R_DIR_LOCAL and run it as follows: *./user_init_plan4res.sh -D P4R_DIR -S $SOLVER*

After proceeding to the install of plan4res, the software is installed in P4R_DIR. The datasets should be stored in the subdirectory ***data*** of P4R_DIR_LOCAL. Depending of your choices in the install process, P4R_DIR and P4R_DIR_LOCAL may be identical or different. (in particular if you are installing on a server, the software will be in P4R_DIR, and each user will have his P4R_DIR_LOCAL directory, configured by the *user_init_plan4res.sh* script)

The following subsections details the installation process.

It is recommended to install plan4res with the full environment (p4r-env), which is embedded in a singularity container. This requires at least 3Gb. In case you do not have this available, you may install plan4res without the environment, but it requires installing all the dependencies first, which may sometimes be quite tricky.

Plan4res requires use of external solver, which can be CPLEX, GUROBI, SCIP or HiGHS. If you do have a CPLEX or GUROBI license, it is recommended to use one of these software. If not, it is recommended to use HiGHS.

If you wish to use CPLEX, you must have a CPLEX linux installer available, such as cplex_studio2211.linux_x86_64.bin, referenced as *cplex_studioXXXXXX.bin* below.

If you wish to use GUROBI, you must have a GUROBI linux installer, sur as gurobi11.0.1_linux64.tar.gz and a gurobi license file: gurobi.lic.

Before starting the install, create an empty directory (for example *P4R_DIR)* where you want to install plan4res. It should not contain special characters or whitespaces!

Once installed, the plan4res environment will appear as described in section

2.1 Installation of the full plan4res environment, for all operating systems

Requirements: at least 3G available.

2.1.1 Installing on Linux:

Move to your install directory: cd P4R_DIR, and type the following commands:

```
cd P4R_DIR

git clone https://github.com/plan4res/install

mv ./install/* . && rm -rf install

chmod a+x *.sh

./plan4res_install.sh [-S <SOLVER>] [-I <installer>] [-L <license>] [-v
<version>] [-M <mpi>]
```

Where:

- SOLVER is the chosen solver among CPLEX, GUROBI, SCIP, HiGHS. If this option is not provided, HiGHS is chosen
- Installer is the installer file for CPLEX of Gurobi, the -I option must only be provided when CPLEX or GUROBI are chosen
- License is the license file, only for GUROBI
- Version is the solver version, only for SCIP
- Mpi is the chosen MPI version among MPICH and OpenMPI. If this option is not provided, OpenMPI is chosen.

Examples:

./plan4res_install.sh

- Installs plan4res with HiGHS

./plan4res_install.sh -S CPLEX -I cplex_studio2211.linux_x86_64.bin

- Installs plan4res with CPLEX, using ths installer cplex_studio2211.linux_x86_64.bin available in P4R_DIR

./plan4res_install.sh -S GUROBI -I gurobi11.0.1_linux64.tar.gz -L gurobi.lic

- Installs plan4res with GUROBI using the gurobi installer and licence available in P4R_DIR

./plan4res_install.sh -S SCIP -v 9.1.1

- Installs plan4res with SCIP, choses the version 9.1.1 from SCIP

It is possible to configure a specific location where the user wishes to run plan4res. This also allows to install the software once on e.g. a shared directory in an organization, and run it on different user's sessions.

For doing so, copy the script user_init_plan4res.sh which is present in P4R_DIR to the location where the user wants to run plan4res, e.g. P4R_DIR_LOCAL and run the command:

```
./user_init_plan4res -D P4R_DIR -S SOLVER
```

Where:

- P4R_DIR is the directory where plan4res has been previously installed
- SOLVER is the chosen solver among CPLEX, GUROBI, SCIP, HiGHS. If this option is not provided, HiGHS is chosen (this allows to configure the settings files in the example of dataset according to the choice of the solver)

2.1.2 Installing on Windows, using WLS

Follow the procedure 'Installing on Linux'

2.1.3 Installing on Windows, using Vagrant

The procedure is available at **https://gitlab.com/cerl/plan4res/p4r-env#windows**. It is reproduced below with some more information.

Installation requires Windows 7 Pro 64bit SP1 or higher and PowerShell 3.0 or higher. Furthermore, the CPU must support hardware virtualization. On many systems, the hardware virtualization features first need to be enabled in the BIOS.

2.1.3.1 Required packages Installation (execute once)

- Install Git for Windows (use default settings) https://git-for-windows.github.io/
- Install VirtualBox and Extension Pack https://www.virtualbox.org/wiki/Downloads
- Install Vagrant https://www.vagrantup.com/downloads.html
- (Optional) Install Vagrant Manager http://vagrantmanager.com/downloads/

The goal of Vagrant and VirtualBox is to emulate a UNIX system on the Windows computer. Vagrant Manager provides a GUI to facilitate the management of the VM.

2.1.3.2 Installation

Run Git Bash.

If working behind a proxy, define the following environment variables:

```
export http_proxy = <proxy address>:<port>
export https_proxy = ${http_proxy}
```

Move to your installation directory:

```
cd P4R_DIR
```

Then enter the following commands:

```
git clone https://github.com/plan4res/install
mv ./install/* .
chmod a+x *.sh
./plan4res_install.sh [-S <SOLVER>] [-I <installer>] [-L <license>] [-v
<version>] -M MPICH -V <memory>
```

Where:

- SOLVER is the chosen solver among CPLEX, GUROBI, SCIP, HiGHS. If this option is not provided, HiGHS is chosen
- Installer is the installer file for CPLEX of Gurobi, the -I option must only be provided when CPLEX or GUROBI are chosen
- License is the license file, only for GUROBI
- Version is the solver version, only for SCIP

Examples:

./plan4res_install.sh -M MPICH -V 8192

- Installs plan4res with HiGHS

./plan4res_install.sh -M MPICH -V 8192 -S CPLEX -I cplex_studio2211.linux_x86_64.bin

- Installs plan4res with CPLEX, using ths installer cplex_studio2211.linux_x86_64.bin available in P4R_DIR

./plan4res_install.sh -M MPICH -V 8192 -S GUROBI -I gurobi11.0.1_linux64.tar.gz -L gurobi.lic

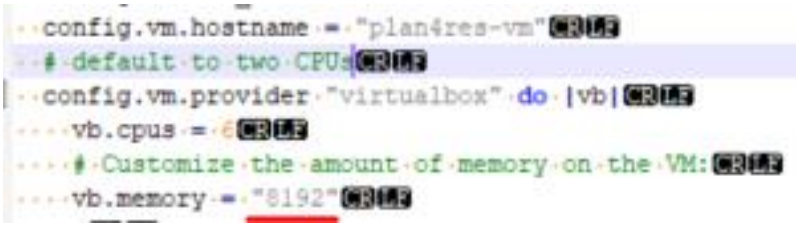
- Installs plan4res with GUROBI using the gurobi installer and licence available in P4R_DIR

./plan4res_install.sh -M MPICH -V 8192 -S SCIP -v 9.1.1

- Installs plan4res with SCIP, choses the version 9.1.1 from SCIP

2.1.3.3 Vagrant settings

To avoid issues with memory fragmentation while using Vagrant, we advise to allocate at least 8 Gb of RAM if the computer can afford it. This can be done via the -V option of the installation command (see above), but you can also edit the file *Vagrantfile* located in *P4R_DIR/p4r-env*, edit parameter `vb.memory` (in Mb):



You may also increase the number of CPUs allocated to the VM if possible, on your machine. In above example, it is set to 6.

2.1.4 Updating

2.1.4.1 Update p4r-env

```
./plan4res_install.sh -U p4r-env
```

2.1.4.2 Update StOpt

```
./plan4res_install.sh -U stopt
```

2.1.4.3 Update SMS++

```
./plan4res_install.sh -U sms++
```

2.1.4.4 Change the solver

```
./plan4res_install.sh -S New_SOLVER [ -I installer -L licence -v version]
```

(installer to be provided in New_SOLVER is CPLEX or GUROBI, licence to be provided if New_SOLVER is GUROBI, version to be provided in New_SOLVER is SCIP)

2.1.4.5 Update the solver

`./plan4res_install.sh -S SOLVER -U SOLVER [ -I installer -L licence -v version]` (installer to be provided in New_SOLVER is CPLEX or GUROBI, licence to be provided if New_SOLVER is GUROBI, version to be provided in New_SOLVER is SCIP)

2.1.4.6 Update plan4res scripts and documentation

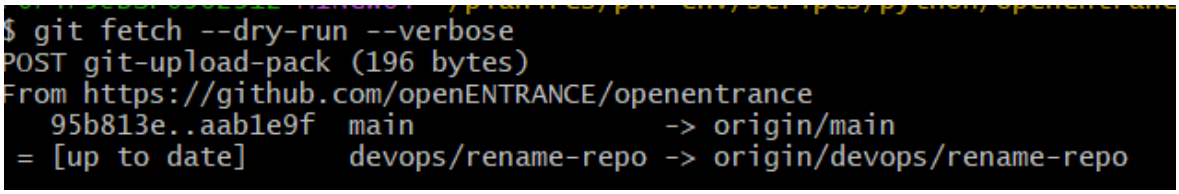
The following directories can be updated:

- `P4R_DIR/documentation`
- `P4R_DIR/p4r-env/scripts/python/openentrance`: to update the openENTRANCE nomenclature definition.
- `P4R_DIR/p4r-env/scripts/python/plan4res-scripts`: to update the data processing and visualization scripts
- `P4R_DIR/p4r-env/scripts/include`: to update the launching scripts

To check if new versions are available, run the following commands:

1. `cd P4R_DIR/p4r-env/scripts/python/openentrance`
2. `git fetch --dry-run --verbose`

If the following output is displayed, it means your installation is up to date:



Otherwise, you can perform the update using:

3. `git pull`

Do the same with the repositories `p4r-env/scripts/python/plan4res-scripts` and `p4r-env/scripts/include` if necessary.

2.1.4.7 Update example of dataset

When installing plan4res, an example of dataset is created in P4R_DIR/data/toyDataset

As plan4res may have been ran in this dataset, it is not recommended to update it but to download the last version of this toyDataset:

- From P4R_DIR/p4r-env/data, change the name of the dataset (mv toyDataset toyDataset_save)
- Download the new version of the dataset:
  `git clone https://github.com/plan4res/toyDataset`

2.2 Installation without p4r-env

It is possible to install plan4res without the p4r-env environment. This requires installing all the dependencies first. It is possible to install on Linux (Ubuntu or Debian), MacOS or Windows

2.2.1 Install the dependencies

2.2.1.1 Python:

Python3 must be the default version of python

The following python packages must be installed:

> *setuptools, openpyxl, scipy, pandas, pyyaml, matplotlib, geopandas, fiona, descartes, shapely, pillow, mapclassify, numpy, requests, requests-toolbelt, rtree, pygeos, wheel, netCDF4, pyam-iamc*

2.2.1.2 SMS++ dependencies

The requirements for installing SMS++ are described in the SMS++ gitlab repo[1]. Their installation may be included in the installation script of SMS++, but, in particular in the case of a Linux installation, the version which would be installed depend on which linux OS is installed (and which version), and thus may be too old. (This in particular happens on linux Debian::bullseye). In that case you should install manually a more recent version.

For each SOFTWARE install, on linux, PATH and LD_LIBRARY_PATH must be updated to include $SOFTWARE_PATH/mpi and $SOFTWARE_PATH/mpi/lib. For BOOST, the variable $BOOST_PATH must also be set.

- **Boost[2] (minimum version 1.87).**
- **OpenMPI[3] (minimum version 3.1.4) OR MPICH[4] (minimum version 3.3.2)**
- **NETCDF[5] (minimum version 4.9.2) and NETCDFCXX (minimum version: 4.3.1).**
- **Eigen[6] (minimum version 3.3.7)**
- **Cmake[7] (minimum version 3.28.1)**

2.2.2 Install plan4res

```
cd P4R_DIR

git clone https://github.com/plan4res/install

chmod a+x *.sh

./plan4res_install.sh -X -S <SOLVER> [-D <solverpath>] [-I
<installer>] [-L <license>] [-v <version>] [-M <mpi>]
```

Where:

- SOLVER is the chosen solver among CPLEX, GUROBI, SCIP, HiGHS. If this option is not provided, HiGHS is chosen
- `solverpath` is to be provided if the solver is already installed in your system. In that case you need to provide the path where it is installed

[1] https://gitlab.com/smspp/smspp-project/-/wikis/Installing-SMS++#requirements
[2] https://www.boost.org/more/getting_started/unix-variants.html
[3] Open MPI: Open Source High Performance Computing
[4] MPICH | High-Performance Portable MPI
[5] netCDF Downloads
[6] libeigen / eigen · GitLab
[7] Kitware/CMake: Mirror of CMake upstream repository

    - for CPLEX: solverpath is where you find e.g. cplex/bin and cplex/lib
    - for GUROBI: solverpath is where you find e.g. gurobi1101/linux64 and gurobi1101/gurobi.lic is version 11.01 is installed
    - for HiGHS: solverpath is where you find bin, include and lib
- Installer is the installer file for CPLEX of Gurobi, the -I option must only be provided when CPLEX or GUROBI are chosen
- License is the license file, only for GUROBI
- Version is the solver version, only for SCIP
- mpi is the chosen MPI version among MPICH and OpenMPI. If this option is not provided, OpenMPI is chosen.=

3 The plan4res environment

3.1 The plan4res environment

The plan4res environment is a container[8]. The plan4res container, p4r-env[9], can be installed in various environments (Linux, Windows, MacOS, CRAY). It provides a complete Linux environment with all dependencies already installed, allowing to run all plan4res modules. Once the environment is installed, the functional behavior will be the same on all kinds of systems. Computation time will depend on the computer's performance.
p4r-env is a singularity container, which means that it requires singularity to be installed.

Note that all components of plan4res can also run without this environment but it requires to install all dependencies manually, which can prove time-consuming and tricky.

The main softwares installed in p4r-env are:

- **StOpt** (see Stochastic Control / StOpt · GitLab): a stochastic optimization library used for solving the Seasonal Storage Valuation (SSV) problem.
- **SMS++** (see SMS++ / The SMS++ Project · GitLab): modelling and optimization library including the 3 main solvers of plan4res in the OM4A toolbox.
- **A preprocessing tool** `p4r-env/scripts/python/plan4res-scripts/CreateInputPlan4res.py`: this tool allows to create plan4res dataset out of long-term pluriannual energy system scenarios (e.g. from GENeSYS-MOD).
- **A formatting tool**, `p4r-env/scripts/python/plan4res-scripts/format.py`, (mandatory apart if you are creating the NetCDF input data files in the format required by SMS++ on your own); this tool creates input data file with the format required by SMS++ taking as inputs user friendly CSV and XLSX files.
- A **Postprocessing tool** `p4r-env/scripts/python/plan4res-scripts/PostTreatPlan4res.py`: this tool transforms the files created by the solver (SMS++) into more user-friendly files and creates some synthetic results.
- **IAMC format transformation tools**: this tool, included in PostTreatPlan4res.py transforms the results from the postprocessing tool into files within the standardized IAMC format.
- A **Visualization tool**: creates graphs.

It is not necessary to launch the p4r-env environment to use plan4res (as launching it is embedded in the plan4res running scripts), but it may be useful in some cases. To launch the plan4res

[8] How to explain containers in plain English | The Enterprisers Project
[9] see CERL / Plan4Res / p4r-env · GitLab

environment, go in the `p4r-env` repository and type `bin/p4r` (on Windows, use Git Bash to do so). This will start a shell in the plan4res containerized environment such as:

```
[P4R-ENV] /…/P4R_DIR/p4r-env >
```

You then have a Linux installation, based on Debian. Basic softwares and utilities are already installed and ready to use, in particular all the dependencies which are necessary for plan4res, in particular python3 and the packages used by the plan4res python scripts.

3.2 Plan4res directories

When installing plan4res (using the plan4res_install.sh script in plan4res/install) in a chosen directory (say, P4R_INSTALL_DIR), you will get the structure shown in Figure 1, and described in section 3.2.1.

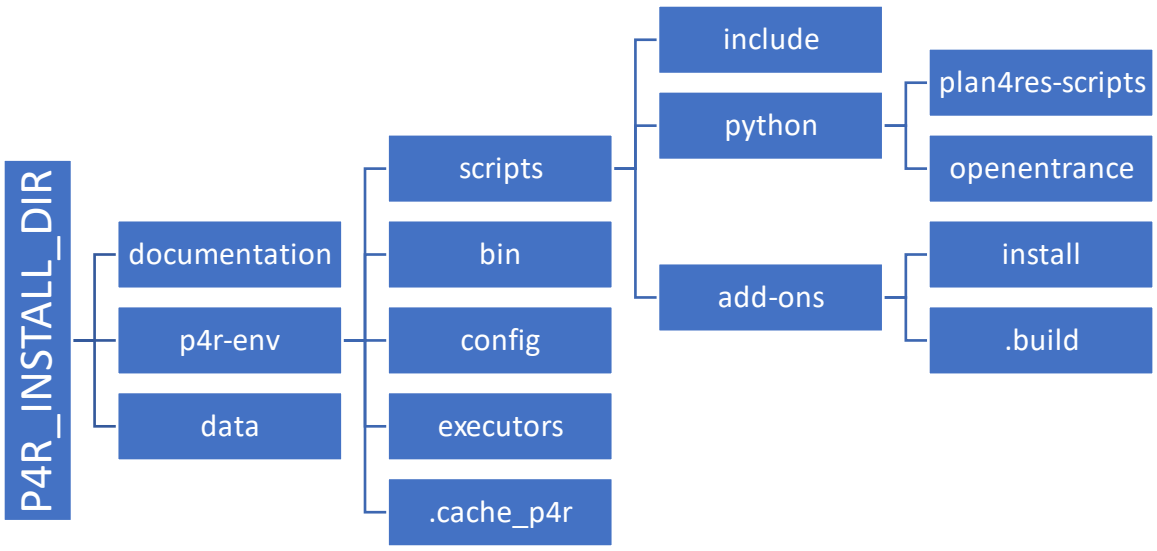


Figure 1: structure of P4R_INSTALL_DIR

You may also choose to run plan4res in a separated directory. This allows, in particular when installing on a server, to install the software only once, and have each user specify a personal directory (by using the user_init_plan4res.sh script in plan4res/install). Each user will then have in their P4R_LOCAL_DIR directory the structure shown in Figure 2:

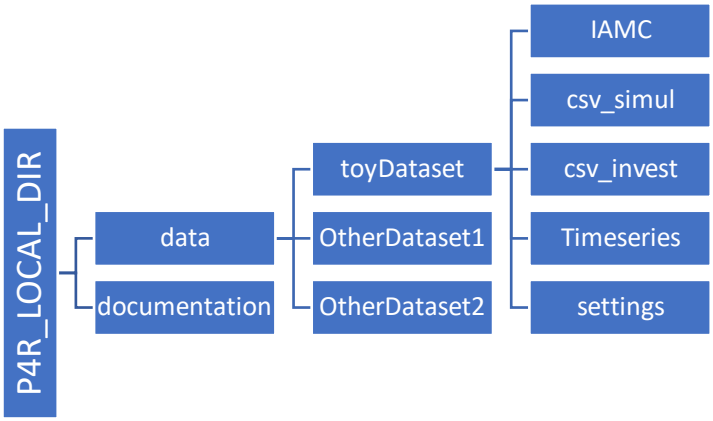


Figure 2: structure of P4R_LOCAL_DIR

3.2.1 P4R_INSTALL_DIR

This directory is structured as follows:

- Documentation: contains the plan4res documentation, downloaded from plan4res/documentation: plan4res documentation
- p4r-env : contains the plan4res environment (container) as well as all the modules of plan4res.
    - .cache_dir: plan4res container image (.SIF). In some cases it is necessary for the user to move the SIF files in another location as some systems specify mandatory locations for SIF images. In that case, it is necessary to update the p4r-env environment variable SINGULARITY_BIND with the location of the SIF file.
    - bin: contains the main p4r-env executables (bin to run the container and the executables used to update or rebuild the container)
    - config: contains the configuration file of p4r-env
    - executors: contains the definitions files of the container
    - scripts: contains all the plan4res modules
        - include: contains all plan4res bash scripts which are used to run the different modules
        - python:
            - plan4res-scripts: contains the python scripts for the CREATE, FORMAT and POSTTREAT modules
            - openentrance: contains the definition of the IAMC nomenclature such as developed in the open ENTRANCE[10] project (openENTRANCE/openentrance: Definitions of common terms (variables, regions, etc.) for the openENTRANCE project)
        - add-ons: contains the solving modules of plan4res: SSV, SIM and CEM, which are based on the SMS++[11] library.
            - Install: installation dir for sms++ and the solvers used within sms++
            - .build: directory where the source code is downloaded and compiled

[10] openENTRANCE – open ENergy TRanstion ANalyses for a low-Carbon Economy

[11] SMS++ / The SMS++ Project · GitLab

- data: optional directory. Created only when the user wishes to run plan4res in the same location where it is installed. The structure of data is described in section 3.2.2.

3.2.2 P4R_LOCAL_DIR

This directory is structured as follows:

- Documentation: contains the plan4res documentation, downloaded from plan4res/documentation: plan4res documentation
- data : directory where the user will run plan4res. The structure of data is described below.

3.2.3 Dataset directory

Figure 3 shows the structure of plan4res.

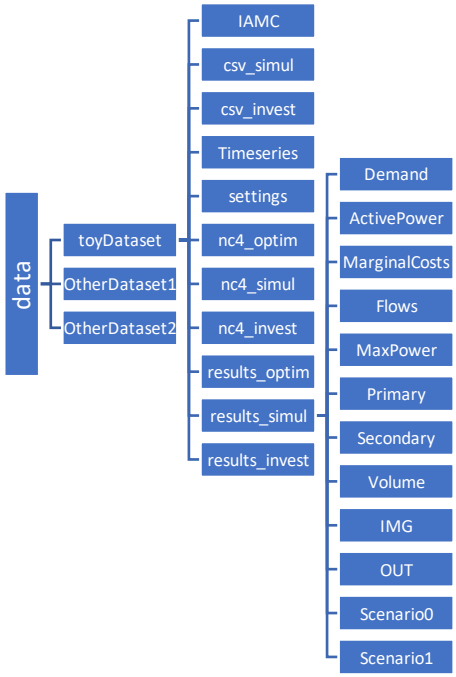


Figure 3: structure of data

Each dataset corresponds to a directory in data (here toyDataset, Otherdataset1 and OtherDataset2). Datasets always have the same structure:

- Timeseries: contains the timeseries which are used as inputs, as well as the timeseries created by the data treatments of plan4res – the CREATE module- (aggregated timeseries over regions, or timeseries with a different time granularity)
- IAMC: contains the scenario which can be used to create the plan4res input data (this can be a result of an energy modelling tool such as e.g. GENeSYS-MOD. The dataset used must be a csv file named DATASE.csv. It must comply to the IAMC format[12].
- csv_simul and csv_invest: those 2 sub repositories contain the native input files of plan4res. These can be created manually by the user or from a scenario in IAMC/ by using the CREATE module of plan4res. csv_simul contains files which cannot be used for performing an investment study, while csv_invest contains files which can be used for performing an investment study. The difference between those 2 sets of files lies in the fact that csv_invest

[12] Description of the IAMC Format: https://doi.org/10.5281/zenodo.5521098

may include some generation assets which are described only because it is allowed to invest in these units, while their capacity is 0. As the investment module (CEM) can only increase the existing capacities, it is necessary to start from a dataset with all possible technologies. The 'new' technologies are then present but with a very low capacity (whose level is defined in the plan4res settings files).

- nc4_optim, nc4_simul and nc4_invest: these 3 sub repositories contain the plan4res NetCDF inputs, which are created from the data in csv_XXX/ and in Timeseries/ by the FORMAT module of plan4res. These datasets are not easily readable so the user may not look at them.
- results_optim: This repo will contain the results of the SSV module
- results_simul: This repo will contain the results of the SIM module
- results_invest: This repo will contain the results of the CEM module
- settings: This repo contains all the configuration files used by all different modules of plan4res.

The data generation modules CREATE and FORMAT as well as the solving modules SSV, SIM and CEM also create the directories when they are not yet present. It is then possible to run CREATE when only IAMC and Timeseries are present, or FORMAT when only Timeseries and csv_simul are present.

3.3 Plan4res modules and workflow

plan4res is composed of the modules listed below. The workflow between the modules is illustrated in Figure 4.

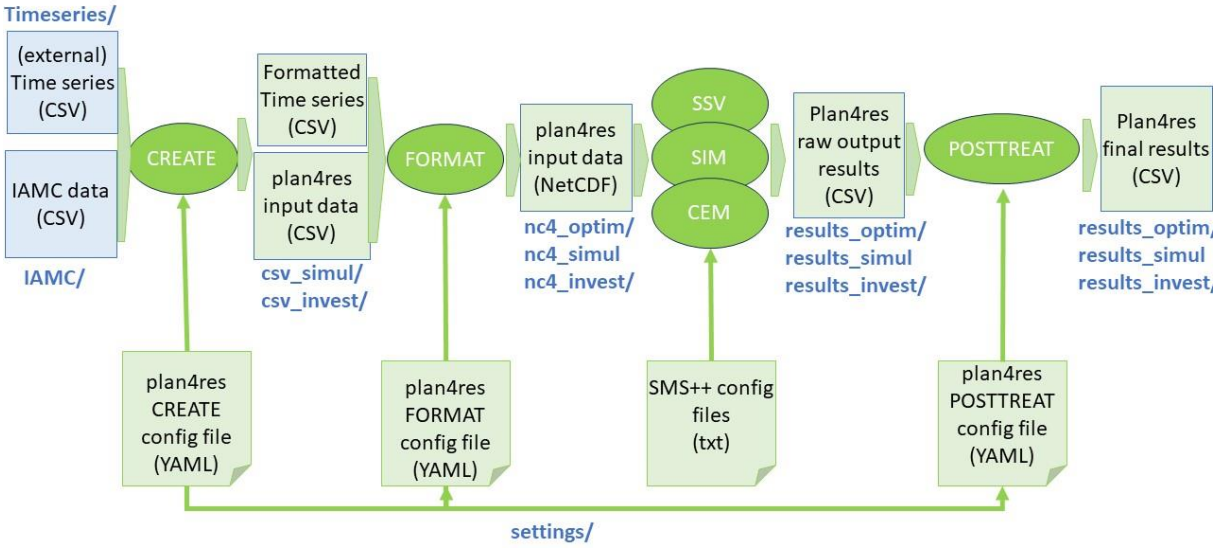


Figure 4: plan4res workflow

- CREATE: creates a plan4res CSV dataset out of a scenario in IAMC format (in IAMC/) and hourly timeseries
- FORMAT: creates a plan4res NetCDF dataset out of the plan4res CSV input data and the hourly timeseries
- SSV: solves the Seasonal Storage Valuation problem. Uses NetCDF files in nc4_optim and computes Bellman Values, stores the results in results_optim
- SSV: solves the Seasonal Storage Valuation problem. Uses NetCDF files in nc4_optim and computes Bellman Values, stores the results in results_optim

- SIM: solves the simulation problem. Uses NetCDF files in nc4_simul and computes schedules of all assets and marginal costs, stores the results in results_simul
- CEM: solves the capacity expansion problem. Uses NetCDF files in nc4_invest and computes new investments, as well as the schedules of all assets and marginal costs for the new power system, including invested assets, stores the results in results_invest

4 Input data for plan4res

This section describes what are the input data of plan4res, for running the different modules (SSV: computation of strategies for the management of seasonal storages, SIM: simulation on a selected year of the operation of the power system, CEM: capacity expansion)

The correspondence of each input data with the IAMC nomenclature is provided in purple.

Data which can be calculated out of the inputs/outputs of GENeSYS-MOD are also identified in **green**. Data which are required but must be added compared to the inputs/ouputs of GENeSYS-MOD are identified in **red.** Other optional data are in blue.

Data formats are described in section

4.1 Geography

Each dataset is linked to a geographical area that may be partitioned. Different partitions may be used for dealing with different levels of constraints or computations. The constraints related to the power demand (which must be equal to the power generation) are applied at the lowest level of the partition, which is called Nodes. It can correspond to countries, or to sub-country regions. Some constraints (namely system services constraints) may apply to different partitions (e.g. to group of countries).

The Figure 5 below illustrates this:

- In blue: the lowest level of partitions (level1), ie the Nodes; In this case there are 7 nodes in level1.
- In green: a second level of partition (level2), with 4 clusters of nodes (named here BE, FR1, FR2 and ES). Note that each level1 node belongs to a unique level2 cluster.
- In red and orange, there are 2 different partitions for the third level , grouping together clusters of level2. The orange partition is composed of 2 clusters (North and South), themselves composed of green clusters defined at level 2. The red partition is composed of 3 clusters (BE, FR, ES), themselves composed of green clusters defined at level 2.
- In black, the biggest partition has a unique cluster.

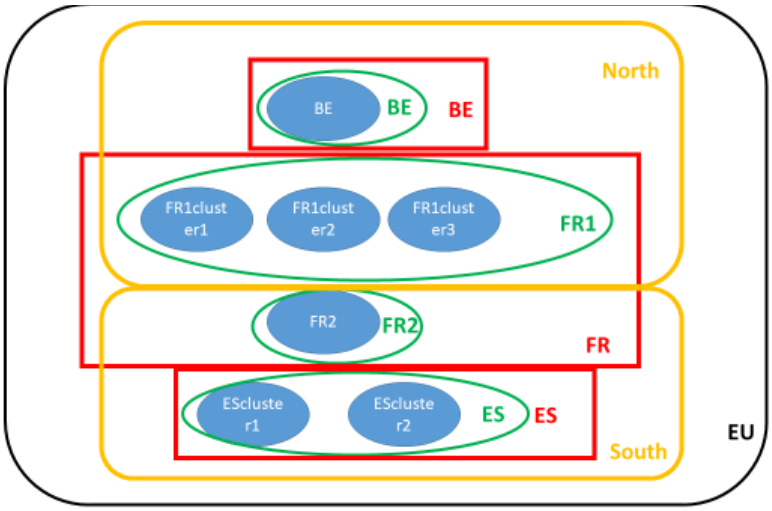


Figure 5: geographical partitions

4.2 Time horizon and granularity

4.2.1 Time horizon

The time horizon needs to be specified. It usually is one year, but for computing the strategies for management of seasonal storages it is recommended to run the SSV module on a longer horizon, namely 18 months. There is no restriction to using longer horizons, but it may lead to very long computation times.

The user will then provide the following data (which may be different for the different modules, providing that consistency is ensured):

- Date/time of the first hour and last hour of the dataset
- Date/time of the first hour and last hour with timeseries available

Note that it is possible to provide the input timeseries on a single year. The CREATE and FORMAT modules will extend / duplicate the timeseries to adapt to the requested period if the time series are not available on the full period. As an example, we may have timeseries available for the year 2018 only, and wish to run a study from Jan 1 2022 to June 30 2023. In that case the 2018 time series will be used 'as if' they were representing 2023, and the first months will be duplicated at the end of the year.

4.2.2 Time granularity

Figure 6 shows how time is discretised in plan4res. There are 2 levels of discretization:

- the SSV timesteps, which are usually weeks (could be months or days). They represent the horizon for the management of short term storages. During the simulations (in SIM but also conducted while solving SSV and CEM), a unit commitment problem will be solved for each scenario and each SSV timestep.
- The timesteps, which are usually 1 hour and are the lowest granularity in plan4res. All timeseries should be provided at this granularity, but if data are not available, plan4res CREATE and FORMAT modules will be able to adapt to higher or lower granularities.

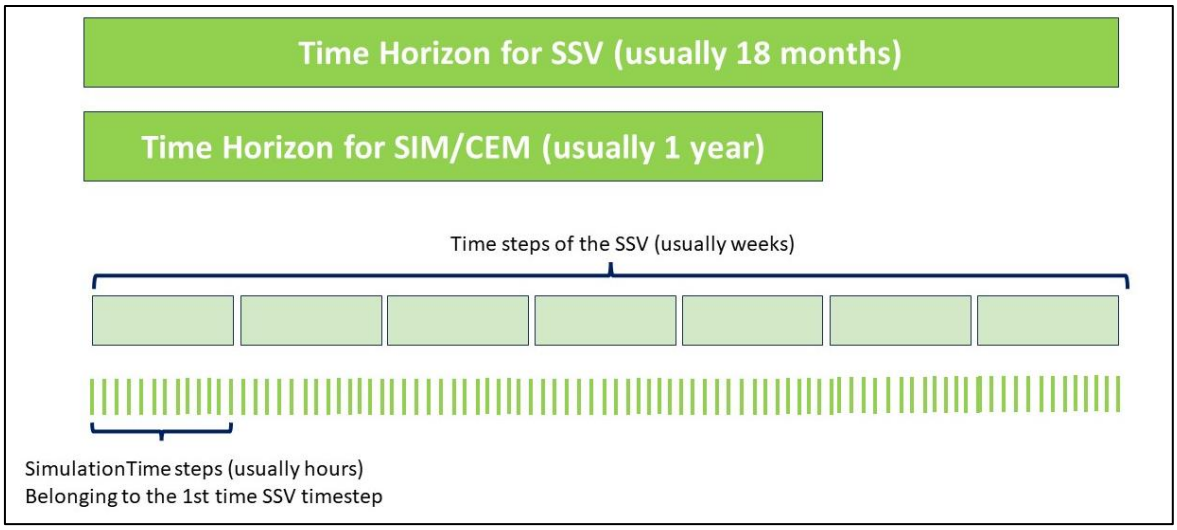


Figure 6: Time representation

4.3 Plan4res input data

The input data are composed of the following groups of data:

- Description of the power system:
  - Interconnections between nodes
  - Generation and storage assets
- Demands:
  - Power demand
  - (optional) System services demands (Primary, Secondary, Inertia)
- (optional) Pollutant limits

Each group may include 'static' data (ie data which are not timeseries) and timeseries.

4.3.1 List of regions

The list of regions related to each constraint level has to be provided:

- List of nodes (e.g. countries); the power demand constraint will be given per node.
- List of regions for each other coupling constraint (if included), and for each region, list of nodes composing it.
  - Reserves can be separated in Secondary and Primary in plan4res.
  - Inertia
  - Pollutant limits.

4.3.2 Data for the power system

4.3.2.1 Interconnections

The list of interconnections between nodes, with for each interconnection:

- **The node where the interconnection starts**
- **The node where the interconnection ends**
- **The maximum power flow in MW** (Network|Electricity|Maximum Flow)
- **The minimum power flow in MW**

- (optional) the impedance of the line
- (optional) the cost of using the line (in €/MWh)
- (optional) **the investment cost (in €/MW)** (Network|Electricity|Expansion Cost)

4.3.2.2 Generation

4.3.2.2.1 Thermal generation

Thermal generation includes all kinds of generation units which can be modelled as a Thermal Unit in plan4res. In particular it includes all traditional generation units running on fossil fuels (coal, gaz, oil), but also biomass units, nuclear units as well as geothermal units.

The generation mix can be described either unit per unit, or technology per technology. The following data are necessary for each unit / technology

- **Capacity (MW) of the technology (if provided per technology, this needs to be given for each node). Can be provided as a deterministic timeseries if depending on time.** (Capacity|Electricity|)
- (optional) Maximum Power of the unit, or of an individual unit if provided per technology (MW) ; Can be provided as a deterministic timeseries if depending on time.
- **Variable Cost (including fuel cost) (€/MWh); Can be provided as a deterministic timeseries if depending on time.** (Variable Cost (incl. Fuel Cost)|Electricity|)
- (optional) **Fixed Cost (€) ; Can be provided as a deterministic timeseries if depending on time.** (Fixed Cost|Electricity|)
- (optional) Quadratic cost (€/Mwh^2); Can be provided as a deterministic timeseries if depending on time.
- **Investment Cost (€/MW); Can be provided as a deterministic timeseries if depending on time.** (Capital Cost|Electricity|)
- (optional) Minimum Power of the unit (MW) (or of an individual unit if provided per technology, or for the whole aggregated technology); Can be provided as a deterministic timeseries if depending on time.
- (optional) Auxiliary power, ie consumption when started but producing 0 (MW)
- (optional) Start up cost (€) for starting the unit after a shut-down.
- (optional) Minimum time down (hours)
- (optional) Minimum time up (hours)
- (optional) Ramping up and down, ie maximum gradients for power increase/decrease (MW per hour)
- (optional) Inertia that the unit can provide in MWs/MWA
- (optional) Share of the maximum power that can be devoted to the FCR
- (optional) Share of the maximum power that can be devoted to the AFRR
- (optional) Forced outage rate
- (optional) Planned outage rate (or schedules for planned outages)
- (optional) Mean outage duration (hours)
- (optional) **CO2 emissions (tons per MWh)** (Emission Rate|CO2|Electricity|)
- (optional) other pollutant emissions

It is possible to account for scenarios of available power for the thermal units. These must then be provided as stochastic timeseries per unit (or per technology) and node.

4.3.2.2.2 Hydro power

3 categories of Hydro Power can be represented in plan4res:

- Seasonal storage, corresponding to the hydro power with large reservoirs, which are operated over the year
- Pumped Hydro, corresponding to hydro power with pumping capacity, and medium size storages, operated on short term periods
- Run of river, corresponding to hydro power without reservoirs.

4.3.2.2.2.1 Seasonal storage

Seasonal storage is usually described as an aggregated unit per node. It can be disaggregated but this may increase a lot the computation time. The following data are necessary for each unit:

- **Capacity (MW) of the technology.** Can be provided as a deterministic timeseries if depending on time. (Capacity|Electricity|Reservoir)
- (optional) if disaggregated, Maximum Power in MW or Maximum Flow (in MW) of each unit
- **Maximum volume of the storage in MWh.** Can be provided as a deterministic timeseries if depending on time. (Maximum Storage|Electricity|Reservoir)
- (optional) As the seasonal storages are managed in MWh, it is assumed that the Minimum Volume is 0, but this data can also be provided as a number or a timeseries.
- (optional) Inertia that the unit can provide in MWs/MWA
- (optional) Share of the maximum power that can be devoted to the FCR
- (optional) Share of the maximum power that can be devoted to the AFRR
- **Inflows to the reservoir in MWh. This must be provided as timeseries, if possible stochastic timeseries (ie a number of scenarios). The time granularity may be hourly or bigger. The timeseries can be profiles which will be used to distribute the yearly inflows in MWh over the period. These profiles must have the following characteristics: the expectancy on all scenarios of the sum of the values over one year is equal to 1. If we know the expected volume of inflows in MWh on the given year, we can then use those profiles to compute representative inflows scenarios by multipliying the profiles by the total inflows. The energy produced by reservoir hydro** Secondary Energy|Electricity|Reservoir **in the GENeSYS-MOD scenario is a good approximation of the total Inflows.**
- (optional) Efficiency: this coefficient will be applied to the flow from the reservoir to compute the power generation. If not present it will be assumed to be 1.
- If the seasonal storage has some pumping capacity, there also need to be provided the Minimum Flow or minimum power (which is negative), as well as the pumping efficiency (which should be lower than 1)
- (optional) Ramping up and down, ie maximum gradients for power increase/decrease (MW per hour)
- (optional) Inertia that the unit can provide in MWs/MWA
- (optional) Share of the maximum power that can be devoted to the FCR
- (optional) Share of the maximum power that can be devoted to the AFRR
- **Volume in the reservoir at the beginning of the study in MWh**

4.3.2.2.2.2 Pumped Hydro

Pumped hydro is usually described as an aggregated unit per node. It can also be disaggregated:

- **Capacity in MW.** Can be provided as a deterministic timeseries if depending on time. (Capacity|Electricity|Pumped Storage)
- (optional) if disaggregated, Maximum Power in MW or Maximum Flow (in MW) of each unit
- **(optional) Minimum power or Minimum Flow in MW (=maximum pumping).** Can be provided as a deterministic timeseries if depending on time. If not provided, assumed to be equal to Capacity * pumping efficiency

- **Maximum volume of the storage in MWh.** Can be provided as a deterministic timeseries if depending on time. (Maximum Storage|Electricity|Pumped Storage)
- (optional) As the pumped storages are managed in MWh, it is assumed that the Minimum Volume is 0, but this data can also be provided as a number or a timeseries.
- **Efficiency when turbining.** Can be provided as a deterministic timeseries if depending on time. (Pumping Efficiency|Electricity|Pumped Storage)
- **Pumping efficiency.** Can be provided as a deterministic timeseries if depending on time. (Charging Efficiency|Electricity|Pumped Storage)
- **Investment Cost (€/MW); Can be provided as a deterministic timeseries if depending on time.** (Capital Cost|Electricity|Pumped Storage)
- (optional) Inertia that the unit can provide in MWs/MWA
- (optional) Share of the maximum power that can be devoted to the FCR
- (optional) Share of the maximum power that can be devoted to the AFRR
- (optional) Inflows to the reservoir in MWh. This can be provided as number in MWh (which will be linearly distributed to the timesteps) or as a deterministic timeseries.
- (optional) Ramping up and down, ie maximum gradients for power increase/decrease (MW per hour)
- (optional) Cost in €/MWh
- (optional) Volume level target at the end of the SSV time step (in that case, it will be ensured that at the end of each SSV timestep, the reservoir reaches this target.
- **Volume in the reservoir at the beginning of the study in MWh**

4.3.2.2.2.3 Run of river

Run of river is usually described as an aggregated unit per node. It can also be disaggregated, but the results will be similar (with longer computation times)

- **Capacity in MW.** (Capacity|Electricity|Run of River)
- **Investment Cost (€/MW); Can be provided as a deterministic timeseries if depending on time.** (Capital Cost|Electricity| Run of River)
- Power profiles. This must be provided as timeseries, if possible stochastic. These profiles will be multiplied by the Capacity to create Maximum Power timeseries. If stochastic profiles are used they must comply to the rule all values must be between 0 and 1, and the average of all values is the load factor of the technology.
- (optional) Minimum power in MW (=maximum pumping). Can be provided as a deterministic timeseries if depending on time. If not provided, assumed to be equal to 0
- (optional) Inertia that the unit can provide in MWs/MWA
- (optional) Share of the maximum power that can be devoted to the FCR and the AFRR

4.3.2.2.3 Variable renewable

Variable renewable includes Solar Power and Wind Power. They must be described per technology when the technologies have different characteristics.

Each technology is usually described as an aggregated unit per node. It can also be disaggregated, but the results will be similar (with longer computation times)

- **Capacity in MW.** (Capacity|Electricity|)
- **Investment Cost (€/MW); Can be provided as a deterministic timeseries if depending on time.** (Capital Cost|Electricity|)
- Power profiles. This must be provided as timeseries, if possible stochastic. These profiles will be multiplied by the Capacity to create Maximum Power timeseries. If stochastic profiles are

used they must comply to the rule all values must be between 0 and 1, and the average of all values is the load factor of the technology.
- (optional) Minimum power in MW (=maximum pumping). Can be provided as a deterministic timeseries if depending on time. If not provided, assumed to be equal to 0
- (optional) Share of the maximum power that can be devoted to the FCR and the AFRR

4.3.2.2.4 Short term storage (excluding hydro)

This category is mostly used for describing the different battery technologies, but may be used for any kind of electricity storage. This technology should be disaggregated as much as possible in order not to over estimate the flexibility that it provides.

- **Capacity in MW.** Can be provided as a deterministic timeseries if depending on time. (Capacity|Electricity|)
- (optional) if disaggregated, Maximum Power in MW or Maximum Flow (in MW) of each unit
- **Investment Cost (€/MW); Can be provided as a deterministic timeseries if depending on time.** (Capital Cost|Electricity|)
- **(optional) Minimum power or Minimum Flow in MW (=maximum pumping).** Can be provided as a deterministic timeseries if depending on time. If not provided, assumed to be equal to Capacity * pumping efficiency
- **Maximum volume of the storage in MWh.** Can be provided as a deterministic timeseries if depending on time. (Maximum Storage|Electricity|)
- (optional) As the pumped storages are managed in MWh, it is assumed that the Minimum Volume is 0, but this data can also be provided as a number or a timeseries.
- **Efficiency when charging/discharging.** Can be provided as a deterministic timeseries if depending on time. (Charging Efficiency|Electricity|) (Discharging Efficiency|Electricity|)
- (optional) Inertia that the unit can provide in MWs/MWA
- (optional) Share of the maximum power that can be devoted to the FCR
- (optional) Share of the maximum power that can be devoted to the AFRR
- (optional) Inflows to the storage in MWh or Demands from that storage (negative values). This can be used to represent e.g. the consumption of electric vehicles if modelling their batteries as a storage. This can be provided as number in MWh (which will be linearly distributed to the timesteps) or as a deterministic timeseries.
- (optional) Ramping up and down, ie maximum gradients for power increase/decrease (MW per hour)
- (optional) Cost in €/MWh
- (optional) Storage level target at the end of the SSV time step (in that case, it will be ensured that at the end of each SSV timestep, the reservoir reaches this target.
- **(optional) Volume in the reservoir at the beginning of the study in MWh**

4.3.2.3 Other flexibilities

Flexibilities such as load shifting can be represented as short term storages. The periods on which some demand can be shifted can be managed via the Minimum Storage and Maximum Storage data.

4.3.3 Demands

4.3.3.1 Power demand

The power demand must be given for each node. As much as possible it is better to provide it as stochastic timeseries, and if possible to disaggregate it per uses. The following data are necessary:

- Demand per uses
    - Demand for electric heating:
        - **Demand in MWh over the whole year** (Final Energy|Electricity|Heating)
        - **Time series , if possible stochastic. The time granularity should be hourly. The timeseries should be profiles which will be used to distribute the yearly demand in MWh over the period. These profiles must have the following characteristics: the expectancy on all scenarios of the sum of the values over one year is equal to 1. If we know the expected demand in MWh on the given year, we can then use those profiles to compute representative demand scenarios by multipliying the profiles by the total demand.**
    - Demand for cooling
        - **Demand in MWh over the whole year** (Final Energy|Electricity|Cooling)
        - **Time series, if possible stochastic. (same characteristics as for electric heating)**
    - Demand for electric transport
        - **Demand in MWh over the whole year** (Final Energy|Electricity|Transportation)
        - **Time series, usually deterministic but can be stochastic. (same characteristics as for electric heating). Timeseries over 1 day could be sufficient if there is no seasonality.**
    - Demand for electric cooking
        - **Demand in MWh over the whole year** (Final Energy|Electricity|Cooking)
        - **Time series, usually deterministic but can be stochastic. (same characteristics as for electric heating). Timeseries over 1 day could be sufficient if there is no seasonality.**
    - Demand for other uses of electricity
        - **Demand in MWh over the whole year** (Final Energy|Electricity|Other (excl. ...)
        - **Time series, usually deterministic but can be stochastic. (same characteristics as for electric heating). Timeseries over 1 day could be sufficient if there is no seasonality.**
- Or if the demand per uses is not available, total demand:
        - **Demand in MWh over the whole year** (Final Energy|Electricity)
        - **Time series, if possible stochastic. (same characteristics as for electric heating)**

4.3.3.2 (optional) Reserves

(optional) The requested level of reserves (FCR, AFFR or aggregated) per region (which could be groups of nodes) or per node, either in share of the power demand, or in MWh, as a timeseries.

4.3.4 (optional) Pollutant limits

(optional) timeseries with the maximum emissions in tons of CO2 (or other pollutant) at each timestep

4.3.5 Input data for the capacity expansion

For running the capacity expansion model, we need:

- The list of technologies that could be invested
- For each technology that could be invested, the maximum potential of the technology, ie the maximum possible added capacity per node.

This includes the generation technologies, but also the storages (except seasonal storages) and the interconnections (if they can be invested, for each interconnection, what is the maximum additional capacity)

5 Using plan4res

A dataset must first be created in P4R_DIR_LOCAL /data. The name of the sub-directory in P4R_DIR_LOCAL /data must be the name of the dataset. An example is given with P4R_DIR_LOCAL /data/toyDataset which is created during the install process.

Once a dataset is created, you can start running the different plan4res modules using the plan4res commands p4r or sp4r (if running on a cluster with SLURM[13]). The commands p4r and sp4r (see section 5.1) are installed during the installation process.

In summary, to conduct a case study, the following steps may be followed:

- **Create a plan4res dataset**:
  - This can be done from a IAMC file (created by GENeSYS-MOD or any other tool) using the command *p4r CREATE YourDataset -M simul* (or p4r CREATE YourDataset -M invest)
  - Or it can be done by editing manually the CSV files.
  - A plan4res dataset is composed of
    - a set of CSV static files in P4R_DIR_LOCAL/data/YourDataset/csv_invest (for studies with capacity expansion) or in P4R_DIR_LOCAL/data/YourDataset/csv_simul (for studies without capacity expansion)
    - AND a set of timeseries in P4R_DIR_LOCAL/data/YourDataset/TimeSeries
- **Run the formatting module to create NetCDF files** for the seasonal storage required by SMS++:
  - *p4r FORMAT YourDataset -M optim -m simul*
  - OR *p4r FORMAT YourDataset -M optim -m invest*
  - Depending if you are creating these files from the data in csv_simul or in csv_invest
- **Run the seasonal storage valuation**: *p4r SSV YourDataset*
- **Run the formatting module to create NetCDF files** for the simulation (this requires that SSV has been ran first as the results of SSV are used): *p4r FORMAT YourDataset -M simul*
- **Run the simulation**: *p4r SIM YourDataset*
- **Run the formatting module to create NetCDF files** for the capacity expansion (this requires that SSV has been ran first as the results of SSV are used): *p4r FORMAT YourDataset -M invest*
- **Run the capacity expansion**: *p4r CEM YourDataset*
- **Post-treat the results**:
  - *P4r POSTTREAT YourDataset -M simul* (for simulation results)
  - *P4r POSTTREAT YourDataset -M simul* (for simulation results)

[13] Slurm Workload Manager - Overview

The workflow is shown in Figure 4: plan4res workflow.

5.1 The plan4res commands: p4r and sp4r

p4r and sp4r are available for using them after installing and configuring plan4res. They allow to run all modules of plan4res (installed in P4R_DIR) from any directory of the user account configured by user_init_plan4res. Independently of the directory in which you are located when running p4r or sp4r, both commands will consider that the datasets are in P4R_DIR_LOCAL/data and the results will also be written in this directory.

5.1.1 The p4r command

The p4r command allows to run all modules of plan4res, without using parallelization.

Usage:

p4r <runtype> <dataset> -M <option1> -m <option2> -o <Dir> -H [save] -S [NumberIterationsFirstStep] [CheckConvEachXIter] -E [epsilon] -L [NumberOfCemIterations] [MaxNumberOfLoops] -D [distance] -U <configfile>"

where:

- **runtype** can be: CLEAN, CREATE, FORMAT, SSV, SIM, CEM, SSVandSIM, SSVandCEM, POSTTREAT:
  - CLEAN: remove all results and revert to initial csv files if necessary (ie if CEM -L was launched)
  - CONFIG: read config file in settings/plan4res_settings.yml and print to screen
  - CREATE: creates a plan4res dataset (csv files) from a IAMC dataset, including data for capacity expansion (with -M invest) or not (with -M simul)
  - FORMAT: creates netcdf files for SMS++, for further run of SSV (with -M optim) based on a dataset with capacity expansion (with -m invest) or not (with -m simul), or for further run of SIM (with -M simul) or for further run of CEM (with -M invest) or for further run of SSV in the case when CEM was already launched (with -M postinvest)
  - SSV: runs the sddp_solver for computing bellman values. With -M $option, the results of SSV will be in results_$option ; if not they will be in results_optim
  - SIM: run simulations using BellmanValues in results_simul, outputs results to results_simul
  - CEM: runs capacity expansion and simulation; outputs results in results_invest; without the option -L, CEM will use BellmanValues in results_invest; with the option -L, BellmanValues will be computed at each iteration of the loop.
  - POSTTREAT: runs posttreatment of results in results_simul with option -M simul, or in results_invest with option -M invest
  - SSVandSIM: runs in sequence SSV then SIM, outputs results in results_simul. Handles creation of NetCDF if used with option -F, and creation of csv files if used with option -C; runs the POSTTREAT in the end.
  - SSVandCEM: runs in sequence SSV then CEM, outputs results in results_invest. Handles creation of NetCDF if used with option -F, and creation of csv files if used with option -C; runs the POSTTREAT in the end.
- **dataset** is the Name of the dataset (also the name of the dataset directory in data/)
- **-o [Dir] (or --out [Dir])** is optionnal ; Dir is the name of a subdir of data/dataset where results will be written; -o is usable for SSV, CEM, SIM, SSVandSIM, SSVandCEM, POSTTREAT
- **-C or --create** is optional: in that case the dataset (csv files) will be created; -C is usable for FORMAT, SSV, SIM, CEM, SSVandCEM, SSVandSIM, CEMloopSSV

- **-F or --format** is optional: in that case the NetCDF[14] dataset for SMS++ will be created; -F is usable for SSV, SIM, CEM (without option -L), (for SSVandCEM, SSVandSIM, CEM -L, creation of netcdf files is mandatory)
- **-H or --hotstart** is optional: in that case a first SSV or CEM run must have been performed; The current run will restart from the results of the previous run ; -H is usable for SSV, CEM ; **-H save** is only used with CEM, meaning that not only the previous solution will be used but also the previous state
- **-S or --steps** is optional: in that case SSV will be ran in 2 steps (first step with convergence checks every 10 iterations, second with checks every iteration); -S is usable for SSV, CEM; The arguments **NumberIterationsFirstStep** (max number of iterations of the first step) and **CheckConvEachXIter** (convergence is checked after CheckConvEachXIter SSV iterations) can be provided either in the command line (-S NumberIterationsFirstStep CheckConvEachXIter**)** or in the plan4res_settings.yml config file (an example of this file is available in toyDataset/settings/plan4res_settings.yml). If the parameters are available in both plan4res_settings.yml and as arguments of the p4r command, the arguments of the command line will be used.
- **-L or --loopssv** is optional: in that case CEM will be launched using a loop of SSV/CEM until convergence; -L is usable for CEM; The arguments **NumberOfCemIterations** (max number of iterations in each run of CEM) and **MaxNumberOfLoops** (max number of iterations SSV/CEM) can be provided either in the command line (-L NumberOfCemIterations MaxNumberOfLoops) or in the plan4res_settings.yml config file (an example of this file is available in toyDataset/settings/plan4res_settings.yml). If the parameters are available in both plan4res_settings.yml and as arguments of the p4r command, the arguments of the command line will be used.
- **-E or --epsilon** is optional: it is the convergence criteria for CEM -L (default 0.01) ; -E is usable for CEM; The argument **epsilon** (convergence criteria of CEM) can be provided either in the command line (-E epsilon) or in the plan4res_settings.yml config file (an example of this file is available in toyDataset/settings/plan4res_settings.yml). If the parameters are available in both plan4res_settings.yml and as arguments of the p4r command, the arguments of the command line will be used.
- **-D or --distance** is optional: it allows to choose which convergence test is used when running CEM with option -L; -D is usable for CEM -L; The argument **distance** (kind of one of the convergence tests) can be provided either in the command line (-E epsilon) or in the plan4res_settings.yml config file (an example of this file is available in toyDataset/settings/plan4res_settings.yml). If the parameters are available in both plan4res_settings.yml and as arguments of the p4r command, the arguments of the command line will be used; distance can be 2 (distance between initial capacity and invested capacity is used for checking convergence) or 3 (distance between cost of investment solution at last 2 iterations is used for checking convergence); if not provided, distance 2 is used.
- **-U or –scenarios** is optional: it allows to run CEM in sequence on different lists of scenarios; -U is usable for CEM; The lists of scenarios must be in the file **configfile** , which must be a text file, and must be provided in the command line (-U configfile). An Example is available in the toyDataset/settings/list_scenarios.txt
- **-M or –mode1** is optional. The chosen option must be provided in command line (-M option1). If not provided, it will behave as if -M simul was provided. -M is useable for CREATE, FORMAT, SSV and POSTTREAT. For CREATE, option1 can be simul or invest (with simul, CREATE will compute a dataset for a simulation run, with invest, CREATE will compute a dataset for a capacity expansion run). For FORMAT, option1 can be optim (for creating NetCDF files for SSV),

[14] Unidata | NetCDF

simul (for creating NetCDF files for SIM), invest (for creating NetCDF files for CEM) or postinvest (for creating NetCDF files for SSV and updating csv files after a CEM run)

- **-m or –mode2** is optional. The chosen option must be provided in command line (-m option2). If not provided, it will behave as if -m simul was provided. -m is useable for FORMAT in the case where option1 is optim: option2 can be invest (for creating NetCDF files for SSV based on the csv files in csv_invest) or simul (for creating NetCDF files for SSV based on the csv files in csv_simul

p4r -h or --help display some help

5.1.2 The sp4r command

sp4r has the same "usage" as p4r apart from a mandatory additional argument: **-n or –nodes** followed by the number of nodes on which to run plan4res.

It will create the file this_sbatch_p4r.sh and run it with sbatch.

Note that the plan4res scripts will manage the configuration of the arguments requested by sbatch such as the number of tasks, …… depending on the module which is ran. Within a workflow of different modules, these parameters are adapted, and each module is launched with srun.

5.2 Creating a plan4res dataset

A plan4res dataset is composed of a number of 'static' CSV files (where static means that these data do not include timeseries) describing the power system ("static" data) and of a number of timeseries (in particular the power demand, inflows, RES potentials hourly – and stochastic- timeseries).

- **The static CSV plan4res input data files**: see the list of data required in the document plan4resInputData[15] and a description of the format of the static csv files in the document plan4resDataFormats[16].

  There are 2 ways of creating the static CSV files:

  - You can manually create the required csv_files[16] in P4R_DATA_LOCAL/data/YourDataset/csv_simul or P4R_DATA_LOCAL/data/YourDataset/csv_invest
  - You can create the csv_files from a dataset in IAMC format (see example in toyDataset/IAMC) using the plan4res command p4r CREATE YourDataset. This requires that a IAMC data excel file named YourDataset.xlsx is available in P4R_DATA_LOCAL/data/YourDataset/IAMC

- **The timeseries** may be historics or forecasts. Most of them can be found in the Copernicus Climate Change Service Data Store[17].

See section 6.3 for the formats of the data files.

[15] documentation/plan4resInputData.pdf at main · plan4res/documentation
[16] documentation/plan4resDataFormatsV2.pdf at main · plan4res/documentation
[17] Climate Data Store

5.2.1 The IAMC excel file(s)

This data file(s), which in general has(have) been created using the GENeSYS-MOD linkage scripts, and contains a number of inputs and outputs from GENeSYS-MOD, follows the common IAMC format[18]. It thus describes a number of scenarios for the transition of the energy system in a number of regions. It is composed of the following columns: Model, Scenario, Region, Variable, Unit and a number of columns corresponding to years, plus an optional column subannual.

Each row gives, for each of the years, the values of a the variable whose name is in column *Variable*, in the region named in column *Region*, for the scenario named in column *Scenario*, generated by the model named in column *Mode, l* using the unit specified in the column *Unit* for the different years defined in the columns named by year indexes. When there exist a *subannual* column, the value of the variable is valid only for the subset of the years such as defined in column *subannual*.

A typical IAMC file computed out of GENeSYS-MOD shall contain the following variables (see the definition of the variables in the nomenclature[19]), for each of the considered regions and years:

- Final Energy|Electricity
- Final Energy|Electricity|<component of final energy electricity>, where |<component of final energy electricity> is among Cooking, Cooling, Heating, Transportation, Electrolizer….. (representing the different uses of electricity)
- Network|Electricity|Maximum Flow
- Network|Electricity|Expansion Cost
- Capacity|Electricity|<Electricity generation and storage technologies>
- Variable Cost (incl. Fuel Cost)|Electricity|<Electricity generation technologies>
- Capital Cost|Electricity|<Electricity generation technologies>
- Fixed Cost|Electricity|<Electricity generation technologies>
- Lifetime|Electricity|<Electricity generation and storage technologies>
- Emission Rate|CO2|Electricity|<Electricity generation technologies>
- Secondary Energy|Electricity|<Electricity generation technologies>
- Charging Efficiency|Electricity|<Electricity storage technologies>
- Discharging Efficiency|Electricity|<Electricity storage technologies>
- Maximum Storage|Electricity|<Electricity storage technologies>

The possible electricity technologies are listed in openentrance/definitions/variable/tag_electricity_input_types.yaml at main · openENTRANCE/openentrance · GitHub, while the possible electricity storage technologies are listed in openentrance/definitions/variable/tag_storage_types.yaml at main · openENTRANCE/openentrance · GitHub.

Note that it is possible to use more than one IAMC file to create a dataset. Some other variables such as variables to define dynamic constraints for individual plants may be provided in these IAMC files, such as:

- Data related to other coupling constraints (reserves….):
    - Network|Electricity|Reserve|Requirement|Frequency Containment
    - Network|Electricity|Reserve|Requirement|Automatic Frequency Restoration
    - Network|Electricity|Requirement|Inertia

[18] Data exchange format and template

[19] openentrance/definitions/variable at main · openENTRANCE/openentrance · GitHub

- Data related to the definition of individual power plants and their dynamic constraints:
    - Maximum Active Power|Electricity|<Electricity generation and storage technologies>
    - Minimum Active Power|Electricity|<Electricity generation and storage technologies>
    - Start-Up Cost|Electricity|<Electricity generation technologies>
    - Minimum Off Duration|Electricity|<Electricity generation and storage technologies>
    - Minimum On Duration|Electricity|<Electricity generation and storage technologies>
- Data related to provision of system services by technologies
    - Inertia|Electricity|<Electricity generation technologies>
    - Frequency Containment Reserve|Electricity|<Electricity generation and storage technologies>
    - Automatic Frequency Restoration Reserve|Electricity|<Electricity generation and storage technologies>
- Data related to outages
    - Forced Outage Rate|Electricity|<Electricity generation technologies>
    - Planned Outage Rate|Electricity|<Electricity generation technologies>
    - Mean Outage Duration|Electricity|<Electricity generation technologies>
- Data necessary if the variable cost of a technology is computed from its efficiency and the price of the used fuel
    - Efficiency|<Electricity generation technologies>
    - Price|<fuels>

5.2.2 Creating a plan4res dataset from IAMC data using the CREATE module

The following operations must be done:

- **Create your dataset repo**: P4R_DIR_LOCAL/data/YourDataset/
- **Create the IAMC repo**: P4R_DIR_LOCAL/data/YourDataset/IAMC/
- **Rename your IAMC data excel file** (optional) as YourDataset.xlsx and **copy it** to P4R_DIR_LOCAL/data/YourDataset/IAMC/ ; if the dataset is created from multiple IAMC files, all must be located in P4R_DIR_LOCAL/data/YourDataset/IAMC/.
- **Create the directory for storing the timeseries**: P4R_DIR_LOCAL/data/YourDataset/TimeSeries/ , create or get the timeseries in the requested format and copy them there.
- **Create the settings directory**: P4R_DIR_LOCAL/data/YourDataset/settings
- **Create the configuration file:**
    - **settingsCreatePlan4res_simul.yml and/or settingsCreatePlan4res_invest.yml** (see section 5.2.2.1)
    - **DictTimeSeries.yml** (see section 5.2.2.2)
    - **VariablesDictionnary.yml** (see section 5.2.2.3)

  The 2 configuration files must be in P4R_DIR_LOCAL/data/YourDataset/settings ; you may start from the examples in P4R_DIR_LOCAL/data/toyDataset/settings

- **Run the p4r (or sp4r) command:**

  p4r CREATE YourDataset

  sp4r CREATE YourDataset -n $N

  the option **-M invest** must be added if you wish to create a dataset for a Capacity Expansion (CEM) run.

- **Check the resulting csv files** in P4R_DIR_LOCAL/data/YourDataset/csv_simul or in P4R_DIR_LOCAL/data/YourDataset/csv_invest. You may now **edit** those files with any text editor or with Excel (recommended) and make any wished changes.

5.2.2.1 Filling the plan4resCreatePlan4res_XXX.yml file

The configuration files settingsCreatePlan4res_simul.yml (for creation of a dataset without possibility of investments) or settingsCreatePlan4res_invest.yml (for creation of a dataset without possibility of investments) define how the plan4res static CSV data will be created out of the IAMC files.

You may edit this file with any text editor. Change the following parameters:

- **scenarios**: name of the scenario use from IAMC data from the Scenario column);
- **years**: year to use from IAMC data from the Years columns);
- **listregionsGET**: list of the regions to extract from IAMC data (Region column);
- **aggregateregions**: defines the user-requested aggregations of regions. Here you can choose to aggregate some regions together. These regions must share a common boarder. You shall choose for each aggregate a name and a list of regions. A region can belong to only one aggregate.
- **partition**: defines the partitions of the regions to be applied to the coupling constraints, defined in the 'Couplin Constraints' section. At least one partition (corresponding to the smallest regions, and to the power demand constraint) must be defined. You can choose the names of each partition and you have to provide the list of regions (which can include aggregates) for each partition. If you define more than one level, the levels must be given in the following order: first the level corresponding to the Power demand constraint, then other levels (with bigger regions). See the partition section in the plan4resInputData[20] document.
- **technos**: list of the technologies to include. These technologies must be present in the Variables in the IAMC file. Technologies are separated into subgroups:
    - **thermal** corresponds to technologies which are modelled as thermal plants (ThermalUnit in SMS++), in particular it includes the thermal fossil plants (with and without CCS), but also the nuclear plants and the geothermal plants.
    - **reservoir** corresponds to units modelled as seasonal storages (HydroUnit in SMS++). It corresponds to Hydro|Reservoir in the IAMC nomenclature.
    - **hydrostorage**: corresponds to hydro units which are modelled as short term storages (BatteryUnit in SMS++). It corresponds to Hydro|Pumped Storage in the IAMC nomenclature.
    - **battery**: corresponds to all other units which are modelled as short term storage (BatteryUnit in SMS++). It corresponds to Battery in the IAMC nomenclature.
    - **res**: corresponds to the variable renewable generation (IntermittentUnit in SMS++). It corresponds to Solar and Wind technologies in the IAMC nomenclature.
    - **runofriver**: corresponds to the Run of river technologies (modelled as IntermittentUnits in SMS++). It corresponds to Hydro|Run of River in the IAMC nomenclature.
    - **demandresponseloadshifting**: corresponds to flexible assets with load shifting capacity. They are modelled as BatteryUnits in SMS++, with additional constraints. It corresponds to residential load shifting in the current IAMC nomenclature (see

[20] plan4res/documentation: plan4res documentation

openentrance/definitions/variable/tag_residential_loadshifting_assets.yaml at main · openENTRANCE/openentrance), but some new categories may be defined in the future.

- **StochasticScenarios**: list of stochastic scenarios to include in the plan4res runs. Note that here scenario has a different meaning than in the IAMC data. It corresponds to the indexes of the different stochastic timeseries. The timeseries may be stochastic for data related to the power demand, the hydro inflows to reservoirs, the availability of thermal plants and the load factors of intermittent generation (including run of river). Each timeseries must comprise the same list of scenario indexes. The StochasticScenarios section of this setting file is meant for selecting a subset of these indexes. Indexes can be numbers or names.
- **CouplingConstraints**: (describes the coupling constraints – see the coupling constraints section in the plan4resInputData document); Coupling contraints can be : Demand, Primary, Secondary, Inertia, CO2 or any other pollutant ; **Partition** defines the level at which the coupling constraint applies (and for the Power Demand it must correspond to the first partition, which must be the partition with the smallest regions). For each constraint, the partition is the level to which this constraint is applied. **MaxPower** and **Cost** (Budget for CO2) are used to create the non-served demand fictive asset, and **SumOf** gives the different pieces of the constraint (e.g. the Power Demand can be computed out of the different uses of electricity; in the example in toyDataset, Power Demand is composed of the demands for cooking, heating, electric vehicle and other uses).
- **ParametersCreate** is a section dedicated to parameters for creation of the plan4res dataset. It includes the following:
    - **invest**: defines wether data for capacity expansion are created or not. If 'yes', create a dataset for the Capacity Expansion model, ie include technologies for which the capacity is zero with a low capacity equal to **zerocapacity.** In practtce, for this value is 'yes' in settingsCreatePlan4res_invest.yml and 'No' in settingsCreatePlan4res_simul.yml
    - **zerocapacity**: defines a capacity in MW under which it is considered to be 0. Also used to add rows for technologies which are not in the scenario data (or are there with 0 for their capacity) so that they can be included in the capacity expansion run
    - **DynamicConstraints**: if 'no' the dataset will not include dynamic constraints: MinPower is set to 0, StartUpCost, MinUp and Down duration are not used. If 'yes' you must include the data for creating these constraints in the IAMC file. You may also create the csv files without dynamic constraints and further edit them.
    - **reservoir**: defines 2 parameters for the seasonal storage:
        - **coordinated**: this value should not be changed. It will in the future allow to create datasets where the seasonal storages will be optimized independently, but this functionality is not yet present.
        - **minpowerMWh**: defines a minimum capacity under which the reservoir is considered to be too small for being managed as seasonal storage. It will be treated as a short term storage
    - **InitialFillingrate**: initial filling rates of reservoirs per regions (regions in the IAMC data file or aggregates as defined above). This is used to create the Initial Volume of reservoirs, which you may later change in the csv files if this data is available.
    - **PumpingEfficiency**: this section is used to define the efficiency of pumpig for all storage technologies when the corresponding variables (Charging Efficiency) are not available in the IAMC file.

- **Volume2CapacityRatio** : this section is used to compute the MaxPower or MaxVolume of storage units when these variables are not available in the IAMC data. In practice, if the capacity is available for a given storage technology but the maximum storage is not, the later will be computed as Maximum Storage / Volume2CapacityRatio
- **CapacityExpansion**: This section is needed only in the case where **invest** is set to 'yes'. For each category of assets -same segmentation as in the definition of **technos**- (**thermal**, **hydrostorage**, **battery**, **res**, **runofriver** – note that reservoir cannot be invested), it gives the list of technologies which can be invested. For each of those technologies, you must specify **MaxAdd** (the maximum capacity in MW that can be added) and **MaxRet** (the maximum capacity in MW that can be decommissioned); you may also specify **InvestmentCost** (in €/MW) if it is not available in the IAMC data. The **interconnections** category corresponds to possible investments in interconnection lines between regions. For this last category, the MaxAdd and MaxRet must be given as multiplication factors (eg MaxAdd=2 means that the capacity may be tripled, MaxRet=0 means that no capacity can be decommissioned).
- **conversions**: This section specify all the conversion rules which must be applied for creating the plan4res dataset (for which the convention is that the units are MW, MWh, €, €/MW, €/MWh, tons). It gives the list of units which require conversion and for each unit, it gives the unit to which it must be converted. Most conversions are already defined in the pyam[21] package which is used in the CREATE module. For those which are not, the conversion **factor** has to be specified.
- **MultFactor**: this section is used to specify the technologies for which the timeseries loadfactores must be applied to an energy instead of a capacity. In practice, this is required for run of river for which maximum power scenarios are computed by multiplying capacity*loadfactors*number of hours in the year
- **Thermal**: this section is used to define some parameters for the units of the 'thermal' category.:
  - **NbUnitsPerTechno**: If the user requires to simulate individual plants with their dynamic constraints, this value must be set to any number above 1. In that case the IAMC data must include variables defining these dynbamic constraints, in particular the Maximum Power to use to define an individual plant per technology. The real number of units will then be computed as Capacity/MaxPower ; If NbUnitsPerTechno is set to 1, aggregates per technologies will be used.
- **variablecost**: if 'Price' is set here instead of Cost, then the variable costs for the list of technologies defined under **fuel** will be computed as the product of the corresponding variables Price (for the corresponding fuel) and Efficiency (for the technology)

- **the datagroup section** lists the different IAMC files that can be used
  - **listdatagroups**: list of subsections of the datagroup section, each one corresponding to a different IAMC file
  - **datagroups** shall contain a subsection per datagroup, with the following parameters:

[21] pyam: analysis and visualization of integrated-assessment & macro-energy scenarios — pyam 3.0.0 documentation

- (optional) **inputdata**: name of the excel file corresponding to the datagroup. If only one datagroup is used with an excel file named by the name of the dataset, this entry is not required.
- **model:** name of the model (from Model column)
- **regions:** this subsection allows to define:
    - **global :** the name of the region which is used for global variables (i.e. variables which do not depend on the region. In an IAMC file, these variable may be defined for e.g. the region 'World' or for one chosen region among those available)
    - **local:** this subsection allows to list additional regions which are not in the regions list in the main section, and should be added to the list of regions for data filtering. It can also be used if the current IAMC file is using different names for the regions, in particular for the countries, such as the countries_ISO3 or countries_ISO2 names (in this case the countries names will be converted)
    - **subannual:** used to define wither the current IAMC file is composed of data at annual granularity, or of data at subannual granularity (meaning there exist a column 'subannual'.
    - **listvariables:** this subsection is used to define the variables to use in the IAMC files. It is composed of 2 parts: the variables related to the coupling constraints (**coupling** subsection) and the variables related to the technologies (**techno** subsection):
        - **coupling**: the variables are separated in categories, depending on the treatments to be done when aggregating data for a number of regions. For each category, you must provide the list of variables which you wish to use. These variables must be present in the related IAMC file. The categories may be:
            - **global**: this corresponds to global variables in the sense that their value is the same for all regions. The name of the corresponding region is the one defined above.
            - **add**: this corresponds to variables that must be added in an aggregation process (e.g. the installed capacity of a given technology)
            - **mean**: this corresponds to variables that must be averaged in an aggregation process (e.g. the variable cost per MWh of a given technology)
            - **flow**: this only applies to the **coupling** section. It corresponds to the variable applying to interconnections between regions (e.g. the maximum flow)
        - **techno**: this section is separated into sub sections corresponding to the technologies types (**thermal**, **reservoir**, **hydrostorage**, **battery**, **res** and **demandresponseloadshifting**), and each of these is separated as shown in coupling among **add**, **mean**, **global**

5.2.2.2 Defining the timeseries to be used: fill DictTimeSeries.yml

The configuration file DictTimeSeries.yml defines which timeseries will be used at which level of the plan4res dataset. Recall that the plan4res CSV files include all static data, but also the names of the timeseries. You may edit this file with any text editor.

This file is organized with one section per plan4res CSV file as follows:

- **ZV**: this section corresponds to the coupling constraints. It must include a subsection per component of the ActivePowerDemand such as defined in settingsPlan4resCreate_XXX.yml, under **SumOf** in the **CouplingConstraints** section. For each of those components, there must be one entry per region giving the name of the timeseries which will be used.
- **SS**: this section corresponds to the seasonal storages. In general it only includes one subsection: **Inflows**. For each subsection (if more than one), there must be one entry per region giving the name of the timeseries which will be used.
- **RES**: this section corresponds to the potentials for the variable renewable energy technologies as well as for the run of river. For each technology (corresponding to the generation technologies as defined in the IAMC nomenclature (e.g. Solar|PV|Rooftop|Commercial), there must be a subsection. For each subsection (if more than one), there must be one entry per region giving the name of the timeseries which will be used.

One timeseries can be referenced many times (for example in the case where only one timeseries will be used for all the regions, or for all the sub categories of a technology).

5.2.2.3 Defining the correspondence between IAMC variable names and plan4res variables names

The configuration file VariablesDictionnary.yml is used to define the correspondence between the native variables of plan4res and the variables defined in the IAMC nomenclature. The plan4res CSV files are using the native variable names plan4res (and this is also the case for the results). This file can be seen as a dictionary of variable names. It is composed of 2 sections:

- **Output**: this section is used in the POSTTREAT module, for converting plan4res native outputs in IAMC format files using variable names from the IAMC nomenclature. This section is described later in this document.
- **Input**: this section is organized in one subsection per CSV file:
    - VarIN corresponding to IN_Interconnections.csv. It must define the following variables:
        - MaxPowerFlow
        - InvestmentCost
    - VarZV corresponding to ZV_ZoneValues.csv. It must define the variables used for computing the different components of the power demand and may also include the variables related to system services (such as defined in SumOf in the section CouplingConstraints of settingsCreatePlan4res_XXX.yml)
    - VarTU corresponding to TU_ThermalUnits.csv. It must define the following variables:
        - Capacity
        - VariableCost
        - FixedCost
        - InvestmentCost
        - CO2Rate

        - Energy
        - LifeTime
        - And a list of optional variables depending on the needs of the user.
    - VarSS corresponding to SS_SeasonalStorage.csv. It must define the following variables:
        - MaxPower
        - MaxVolume
        - Inflows (in general the annual inflows are not available in IAMC data, which is why it is usual to use the secondary energy generated by the technology instead)
        - Energy
        - And a list of optional variables depending on the needs of the user.
    - VarSTS|Hydro corresponding to the hydropower (Pumped Storage) in STS_ShortTermStorage.csv. It must define the following variables:
        - MaxPower
        - MaxVolume
        - PumpingEfficiency or ChargingEfficiency
        - (optional) TurbineEfficiency or DischargingEfficiency or RoundTripEfficiency
        - FixedCost
        - InvestmentCost
        - Energy
        - LifeTime
        - And a list of optional variables depending on the needs of the user.
    - VarSTS|Battery corresponding to the other short term storages in STS_ShortTermStorage.csv. It must define the following variables:
        - MaxPower
        - MaxVolume
        - PumpingEfficiency or ChargingEfficiency
        - (optional) TurbineEfficiency or DischargingEfficiency or RoundTripEfficiency
        - FixedCost
        - InvestmentCost
        - Energy
        - LifeTime
        - And a list of optional variables depending on the needs of the user.
    - VarRES corresponding to RES_RenewableUnits.csv. It must define the following variables:
        - MaxPower
        - Energy
        - FixedCost
        - InvestmentCost
        - LifeTime
        - And a list of optional variables depending on the needs of the user.

Before running the script, the user must:

- Create the following subrepositories within p4r-env/data/local/MY_STUDY:
    - IAMC

    - settings
    - TimeSeries
- Copy the settings files (settings* .yml) from p4r-env/python/plan4res-scripts/ (these files were retrieved during the installation of plan4res) to p4r-env/data/local/MY_STUDY/settings and edit them as wished. The main edits to be made are the following (see details in chapter)
    - settingsCreateInputPlan4res_*.yml:
        - **scenarios:** name of the genesys-mod scenario to be used
        - **years:** selected year for the run
        - **listregionsGET :** list of the regions in genesys-mod outputs
        - **aggregateregions :** only if you wish to define aggregations
        - **partition :** usually composed of a row 'Countries' containing the list of regions in your dataset (ie aggregated regions and remaining regions which are not part of an aggregation)
        - **technos :** should be the list of technos in the genesys-mod dataset
        - **StochasticScenarios :** list of scenarios in your timeseries. If deterministic, this list should comprise only one element
        - **CouplingConstraints :** list of coupling constraints you wish to account for. The minimal requirement is ActivePowerDemand, linked to the partition with the highest granularity.
        - **ParametersCreate:CapacityExpansion :** only for runs with capcity expansion, defines for each technology for whihc it is allowed to invest, the maximum invested (or retired) capacity
        - **ParametersCreate:InitialFillingrate :** level of the seasonal reservoirs at beginning of case study, in %
    - settings_format_*.yml:
        - **Calendar**:
            - Dayfirst: True means that dates are given in the format dd/mm/yyyy HH:MM:SS, while False means they are in the format mm/dd/yyyy HH:MM:SS
        - **BeginTimeSeries** and **EndTimeSeries**: dates of start and end of the available timeseries
        - **BeginDataset** and **EndDataset**: dates of start and end of your run
        - **SSVTimeStep**: duration of the timestep for the SSV (usually one week)
        - **TimeStep**: duration of the timestep for the simulation(usually one hour)
        - **Parameters**: list of parameters for creating the dataset. In particular the list of scenarios must be checked
    - settingsPostTreatPlan4res_*.yml:
        - **BeginTreatData** and **EndTreatData**: subperiod of the simulation results on which to postreat the results
        - **Graphs**: defines the size of graphs and how they are set on each sheet (in particular for graphs composed of sub graphs per region)
        - Names of scenario and report
        - **Technos**: for each techno in the dataset, defines the colorcode
        - **technosAggr**: defines user-chosen aggregations of technologies as lonked color codes
        - **pumping**: MUST include the list of technos which may pump

- Copy the results of a genesys-mod (IAMC format) run to p4r-env/data/local/MY_STUDY/IAMC
- Get or create the necessary timeseries (generally load profiles for electricity demand, load factors for renewable generation and inflows to reservoirs....) and store them in p4r-env/data/local/MY_STUDY/TimeSeries
- Copy the SMS++ configuration files (sddp_solver.txt, BSPar-Investment.txt, sddp_greedy.txt, sddp_greedy_investment.txt and uc_solverconfig.txt) (those files were retrieved during the plan4res installation) to p4r-env/data/local/MY_STUDY/settings/

5.2.2.4 The CREATE module

This module uses the python script CreateInputPlan4res.py (in P4R_DIR/p4r-env/scripts/python/plan4res-scripts).

This script creates the csv files in plan4res data format (see section 6.3) out of results of GENeSYS-MOD in IAMC format (see section 6.1)

CreateInputPlan4res.py requires 3 configuration files:

- settingsCreateInputPlan4res.yml (see section 5.2.2.1) which is the main configuration file.
- VariablesDict.yml : contains the list of variables to retrieve and the correspondence between the plan4res and IAMC variable names (see section 5.2.2.3)
- TimeSeriesDict: contains the list of timeseries to use for the different stochastic variables (described in the main configusation file) and regions (see section 5.2.2.2)

The following operations are done for creating a plan4res dataset corresponding to one scenario and one year (multiple datasets can be created in sequence):

1- Read annual data for the given year and scenario, using the list of variables and regions to retrieve, given in the configuration file (settingsCreateInputPlan4res.yml), together with their locations (the data can be separated in multiple data sources).

2- Data conversion following conversion rules in the configuration file

3- Regional aggregation for both annual data and timeseries

4- Creation of all csv files in pla4res data format.

5.3 Creating NetCDF files out of the csv files before running the plan4res calculation modules: the FORMAT module

The FORMAT module can be used as follows:

p4r FORMAT YourDataset -M option1 -m option2

option1 allows to choose which kind of NetCDF files will be created: for running SSV (option1=optim), for running SIM (option1=simul), for running CEM (option1=invest), or in the case where CEM has already be ran and we want to create NetCDF files for SSV, based on the new installed capacities as computed by CEM, option1=postinvest. (this is in particular used in CEM -L, see below)

option1 is used only when option1=optim. It allows to specify which csv_XXX to use for creating the NetCDF files for SSV.

FORMAT uses the python script format.py (in P4R_DIR/p4r-env/scripts/python/plan4res-scripts).

format.py reads the csv plan4res input files (see section 6.3 for a description of the format), and creates a serie of NetCdF files (in the SMS++ format, see section 6.4) :

- SDDPBlock.nc4: this file, used for optimisation, investment and simulation, describes the full problem, apart from investment ;
- InvestmentBlock.nc4: this file, used for investment only, describes the investment problem.
- N files Block_i.nc4: each one describes the assets for the SSV timestep i

When used with option -M postinvest, format.py also updates the data in csv_simul and in csv_invest, and creates backup files for the original data.

format.py requires 4 configuration files:

- settings_format_XXX.yaml (see section 5.3.1.1)
- settingsCreateInputPlan4res.yml (see section 5.2.2.1) which is also the main configuration file of CREATE.
- VariablesDict.yml : contains the list of variables to retrieve and the correspondence between the plan4res and IAMC variable names (see section 5.2.2.3)
- TimeSeriesDict: contains the list of timeseries to use for the different stochastic variables (described in the main configusation file) and regions (see section 5.2.2.2)

5.3.1.1 Filling the settings_format_XXX.yml file

The configuration files settings_format_optim.yml (for creation of NetCDF files for SSV), settings_format_simul.yml (for creation of NetCDF files for SIM), settings_format_invest.yml (for creation of NetCDF files for CEM), or settings_format_postinvest.yml (for creation of NetCDF files for SSV in the case of running SSV after CEM) define how the NetCDF files will be created out of the plan4res CSV static data and timeseries.

You may edit this file with any text editor. Change the following parameters:

- (do not change) **outputDir**: sub-directory of p4r-env/data/local/MY_STUDY/ where nc4 files are created (in practice nc4_optim, nc4_simul or nc4_invest)
- (do not change) **inputDir**: sub-directory of p4r-env/data/local/MY_STUDY/ where csv files are (in practice csv_simul or csv_invest)
- (do not change) (only in settings_format_postinvest.yml) **resultsDir**: sub-directory of p4r-env/data/local/MY_STUDY/ where the results of the capacity expansion (CEM) are located (in practice results_invest)
- (do not change) (only in settings_format_postinvest.yml) **investDir**: sub-directory of p4r-env/data/local/MY_STUDY/ where the input csv files for the capacity expansion are (in practice csv_invest)
- (do not change) **FormatMode**: defines which kinds of blocks are created: SingleUC : generates ONE UCBlock for the first period (first SSVTimestep) of the dataset, UC : generates a serie of UCBlocks for each period (each SSVTimestep) of the dataset, SDDP : generates only the SDDPBlock, SDDPandUC : generates the SDDPBlock and all the UCBlocks, INVEST : generates the InvestmentBlock, INVESTandSDDP : generates the InvestmentBlock and the SDDPBlock, INVESTandSDDPandUC : generates the InvestmentBlock and the SDDPBlock and all the UCBlocks; in practice modes SDDPandUC is used when running without investment and mode INVESTandSDDPandUC when running with investments.

- (do not change) **FormatVU** defines the kind of bellman values used as inputs: None means that the bellman values are not included in the NetCDF files (they are passed as a separate file), PerReservoir means that 1 belmanvalue file per reservoir is used, and Polyhedral means that a single belmannvalues file for all reservoirs (with coefficients per reservoir) is used.
- (do not change) **IncludeVU** defines at which ssv timestep bellman values are included: None means that no bellman values are used, Last means that they are used only at the last timestep and All means they are used at all timesteps.
- **Invest** defines the additional constraints for investment optimization: Simple = no additional constraints are created, NRJ= regions are autonomous in energy (ie the amount of energy is sufficient for each node), TargetRES= each region has a target of renewable energies
- **Calendar**: (see **Erreur ! Source du renvoi introuvable.**) this section is dedicated to the time data.
    - **dayfirst**: True if the format is giving the day first (01/07/2050 means first of july)
    - **BeginTimeSeries** : start of the available timeseries (in case the timeseries are available eg only for year 2050 and the dataset is created for 2030, the formatting tool will use anyway the timeseries and translate the dates)
    - **EndTimeSeries** : end of the timeseries
    - **BeginDataset** : first hour of the dataset in the NetCDF files
    - **EndDataset**: end of the dataset in the NetCDF files
    - **SSVTimeStep**: duration of the SSV timestep (eg 1 week); This duration is expressed as a number of XXX, where this number is given under **Duration** and XXX is the chosen "unit" (can be hours, days or weeks)
    - **TimeStep**: duration of the timestep (eg 1 hour) (TimeStep is always lower or equal than SSVTimeStep). This duration is expressed as a number of XXX, where this number is given under **Duration** and XXX is the chosen "unit" (can be hours, days or weeks)
- (do not change) **IncludeScenarisedData**: True or False wether scenarised data should be included in the Blocks (not necessary for running plan4res, but can be useful for running simulations on a single UCBlock). The data in the blocks will correspond to those of the first scenario.
- **ParametersFormat:** this section gives a number of parameters
    - **DownReservoirVolumeMultFactor**: multiplicative factor for computing the size of virtual downstream reservoir from the maximum volume of the upstream reservoir
    - **DownDeltaRampUpMultFactor**: multiplicative factor for computing the maximum ramp up of the virtual downstream reservoir from the ramp up of upstream reservoir
    - **DownDeltaRampDownMultFactor**: multiplicative factor for computing the maximum ramp down of the virtual downstream reservoir from the ramp down of the upstream reservoir
    - **NumberHoursInYear**: number of hours in a year (8760). This is used for converting power to energy.
    - **InertiaMultFactor**: multiplicative factor for computing the inertia contributions of the assets
    - **Scenarios**: list of the scenarios to include in the instance (indexes of scenarios, among those available in the timeseries). It can include all scenarios that are available in all stochastic timeseries. It can also be only a subset.
    - **ScenarisedData**: this section contains the list of scenarised data, among 'ActivePowerDemand','Hydro:Inflows','Renewable:MaxPowerProfile' and 'Thermal:MaxPowerProfile'; For each it must be specified wether the profile

available in timeseries should be multiplied by an energy or a power in order to create the data. For the Renewable:MaxPowerProfile subsection, this information must be given for **res** (i.e. variable renewable energy, usually Wind and PV power) and for **runofriver.**

- **ThermalMaxPowerTimeSpan**: frequency of the data for scenarised thermal max power profiles. This is expressed as a duration, defined as a number of XXX, where this number is given under **Duration** and XXX is the chosen "unit" (can be hours, days or weeks). For example, 168 hours means that the same value of the scenarios for 'Thermal:MaxPowerProfile will be used each week. These scenarios thus do not need to have a higher granularity.
- **CoeffSpillage**: it is allowed to spill CoeffSpillage*Maximum Flow
- **LowerBound**: optional: lower bound for the SDDP algorithm in SSV. This is optional.
- (do not change) **RecomputeCSV**: True if it is required that format.py will compute new CSV files in csv_simul and csv_invest to account for new installed capacities as computed by CEM. This is used only in settings_format_postinvest.yml. If it is not present it is assumed to be False.
- **RoundDecimals**: number of decimals to keep for rounding values.

5.4 Running the plan4res optimization modules SSV, SIM and CEM

These 3 modules are the main components of plan4res. They correspond to executables of the SMS++ library. They are all launched using p4r MODULE YourDataSet. A number of options can be added to this command (see section 5.1.1), in particular -C (to force use of CREATE first) and -F (to force use of FORMAT first) and -o Dir (to write the results in a sub-directory Dir of results_XXX/) . Some of these options (see below) allow to pass parameters, which can also be included in the plan4res_settings.yml configuration file. Whenever a parameter is present in the configuration file and via an option of p4r, the value which will be used is the one in the option of p4r.

5.4.1 Running the SSV module

The SSV module can be launched as follows:

```
p4r SSV YourDataset
```

This module will compute optimal strategies for the management of seasonal storages, which are defined as Bellman Values. The FORMAT module must have been ran before (to produce the NetCDF files)

Inputs: SDDPBlock.nc4 and Block$i.nc4 ($i are indexes of SSV timesteps), in nc4_optim/

Outputs: BellmanValuesOUT.csv and cuts.txt (BellmanValuesOUT.csv is created only if the sddp has reached convergence; cuts.txt exists as soon as it has completed at least one iteration), in results_optim/. See the format of these files in section

SSV requires the following **configuration files**:

- Plan4res.yml (see
- The SMS++ configuration files sddp_solver.txt and uc_solverconfig.txt. These configuration file is updated by the SSV module. See section **Erreur ! Source du renvoi introuvable.** for more details.

SSV can be re-ran in hot-start (ie using the current solution as a starting point) when adding option -H: p4r SSV YourDataSet -H

SSV can also be ran in 2 steps, using option -S. this allows to decrease the computation time. The parameters of -S (**NumberIterationsFirstStep** and **CheckConvEachXIter**) can be passed either via the p4r command: p4r SSV YourDataSet -S NumberIterationsFirstStep CheckConvEachXIter, or in the plan4res_settings configuration file (p4r SSV YourDataSet -S). The 2 steps of SSV are: a first step with maximum NumberIterationsFirstStep iterations, in which the convergence is checked each **CheckConvEachXIter** iteration, and a second step which is using the parameters defined in sddp_solver.txt. Checking the convergence takes some time, as it means simulating some scenarios, and it may not be necessary to do it often in the first iterations, which is why this option may save computation time.

5.4.2 Running the SIM module

The SIM module can be launched as follows:

p4r SIM YourDataset

This module will simulate the optimal operation of the power system over the whole period. The SSV module and the FORMAT module must have been ran before (to produce the NetCDF files and the Bellman Values). It will compute the optimal schedules for each asset in all the regions for all the scenarios, as well as the marginal costs of all the coupling constraints in all the regions and for the interconnections.

Inputs:

- SDDPBlock.nc4, Block$i.nc4, and InvestmentBlock.nc4 ($i are indexes of SSV timesteps), in nc4_simul/
- BellmanValuesOUT.csv or cuts.txt in results_optim/

Outputs: (in results_simul/)

- Optimal schedules of all assets: files ActivePower$i.csv, Primary$i.csv, Secondary$i.csv,
- Marginal costs: MarginalCostActivePowerDemand$i.csv, MarginalCostPrimary$i.csv MarginalCostSecondary$i.csv MarginalCostInertia$i.csv MarginalCostFlows$i.csv
- Volumes of all reservoirs (seasonal and short-term): Volume$i.csv,
- Flows in all lines: Flows$i.csv

SIM requires the following **configuration files**:

- Plan4res.yml (see
- The SMS++ configuration files BSPar-investment.txt, sddp_greedy.txt, sddp_greedy_investment.txt and uc_solverconfig.txt. These configuration file is updated by the SSV module. See section **Erreur ! Source du renvoi introuvable.** for more details.

5.4.3 Running the CEM module

5.4.3.1 Simple run of CEM

The CEM module can be launched as follows:

p4r CEM YourDataset

This module will compute an optimized installed capacity (including also adapting the capacity of short term storages and of interconnections) and then simulate the optimal operation of this adapted power system over the whole period. The SSV module and the FORMAT module must have been ran before (to produce the NetCDF files and the Bellman Values).

Inputs:

- InvestmentBlock.nc4, SDDPBlock.nc4, Block$i.nc4, and ($i are indexes of SSV timesteps), in nc4_invest/
- BellmanValuesOUT.csv or cuts.txt in results_optim/
- the csv files in csv_invest and csv_simul when running with option -L

Outputs: (in results_invest/)

- Adapted system: Solution_OUT.csv
- Optimal schedules of all assets: files ActivePower$i.csv, Primary$i.csv, Secondary$i.csv,
- Marginal costs: MarginalCostActivePowerDemand$i.csv, MarginalCostPrimary$i.csv MarginalCostSecondary$i.csv MarginalCostInertia$i.csv MarginalCostFlows$i.csv
- Volumes of all reservoirs (seasonal and short-term): Volume$i.csv,
- Flows in all lines: Flows$i.csv

CEM requires the following **configuration files**:

- Plan4res.yml (see
- The SMS++ configuration files BSPar-investment.txt and uc_solverconfig.txt. These configuration file are updated by the CEM module. See section **Erreur ! Source du renvoi introuvable.** for more details.

CEM can be re-ran in hot-start (ie using the current solution as a starting point) when adding othe ption -H: p4r CEM YourDataSet -H

The option -E epsilon allows to specify the convergence criteria of CEM. Epsilon may also be passed in plan4res_settings.yml.

5.4.3.2 Complex runs of CEM

CEM may also be ran with additional options which allow to:

- **Recompute the bellman values every N iterations of the capacity expansion algorithm**: p4r CEM YourDataset -L

  CEM -L is equivalent to running an algorithm composed of a number of iterations where in each iteration CEM is ran, its outputs are used to compute a new dataset (i.e. new files in csv_invest and csv_simul accounting for the results of the CEM run, and new NetCDF files in nc4_optim based on the new CSV files), and SSV is ran to compute new Bellman Values which will be used in the next iteration.

  The parameters of -L (**NumberOfCemIterations** and **MaxNumberOfLoops**) can be passed either via the p4r command: p4r CEM YourDataSet -L NumberOfCemIterations MaxNumberOfLoops, or in the plan4res_settings configuration file (p4r CEM YourDataSet -L). **NumberOfCemIterations** is the maximum number of iterations in each run of CEM and **MaxNumberOfLoops** is the maximum number of iterations SSV/CEM). If the parameters are available in both plan4res_settings.yml and as arguments of the p4r command, the arguments of the command line will be used.
  CEM -L may also use the parameter **Distance** which can be passed via the option -D distance, or via the plan4res_settings file. This parameter allows to choose which convergence test is used when running CEM with option -L: **distance** can be 2 (distance between initial capacity and invested capacity is used for checking convergence) or 3 (distance between cost of investment solution at last 2 iterations is used for checking convergence); if not provided, distance 2 is used.

- **Launch in sequence the algorithm on different subsets of the scenarios**. This may be useful in case the computation time is too long with all scenarios. It will allow to compute a first result on e.g. 1 scenario, then hot-start the algorithm on a bigger subset of scenarios.

  p4r CEM YourDataset -U list_scenarios.txt

  This requires the additional configuration file: list_scenarios.txt. This file gives on its first row the number of subsets of scenarios to be used (it 2, it means that CEM will be launched twice), and on the following rows, one subset is to be described in each row, with the format ['scenario1','scenario2'] (Here we assume that we are defining a subset with 2 scenarios, and that the scenarios are named 'scenario1' and 'scenario2'. Don't forget the quotes…..

5.4.4 The configuration file plan4res_settings.txt

This configuration file contains parameters for the plan4res modules SSV and CEM. Some of those parameters can be passed as options of the p4r command (in this case the value passed in option will always be used).

This file contains the following parameters:

- **SSV parameters**
  - **NumberSSVIterationsFirstStep**: maximum number of iterations of the first step, when using option -S
  - **NumberSSVIterations**: maximum number of iterations when not using option -S, or maximum number of iterations in the final step when using option -S
  - **CheckConvEachXIterInFirstStep**: convergence is checked after X iterations of the sddp algorithm in the first step, when ran with option -S
  - **CheckConvEachXIter**: convergence is checked after X iterations of the sddp algorithm in the unique SSV step when ran without option -S, and in final SSV step when ran with option -S
  - **EpsilonSSV**: convergence criteria of SSV

- **CEM parameters**
  - **NumberOfCemIterationsInLoopCEM**: maximum number of iterations in each run of CEM when CEM is launched with option -L
  - **NumberOfCemIterations**: maximum number of iterations in each run of CEM when CEM is launched without option -L or in the last CEM run when launched with -L
  - **MaxNumberOfLoops**: maximum number of iterations SSV/CEM when CEM is launched with option -L
  - **EpsilonCEM**: convergence criteria for CEM when launched with option -L
  - **Distance**: choice of the distance computation for CEM when launched with option -L. distance can be 2 or 3 -default 2- 2 means that the distance between the initial capacity and the invested capacity is used for checking convergence; 3 means that the distance between the cost of the investment solution at the last 2 iterations is used for checking convergence
  - **ScenariosLists**: name of the configuration file for running CEM with option -U

5.4.5 Other modules: SSVandSIM and SSVandCEM

These modules include complete workflows.

p4r SSVandSIM YourDataset does the following:

- If used with option -C: p4r CREATE YourDataset

- p4r FORMAT YourDataset -M optim
- p4r SSV YourDataset
- p4r FORMAT YourDataset -M simul
- p4r SIM YourDataset
- p4r POSTTREAT YourDataset

p4r SSVandCEM YourDataset does the following:

- If used with option -C: p4r CREATE YourDataset -M invest
- p4r FORMAT YourDataset -M optim -m invest
- p4r SSV YourDataset -M invest
- p4r FORMAT YourDataset -M invest
- p4r CEM YourDataset
- p4r POSTTREAT YourDataset -M invest

5.5 Post-treating the results: the POSTTREAT module

The POSTTREAT module can be launched as follows:

p4r POSTTREAT YourDataset -M simul

p4r POSTTREAT YourDataset -M invest

POSTTREAT uses the python script PostTreatPlan4res.py (in P4R_DIR/p4r-env/scripts/python/plan4res-scripts).

PostTreatPlan4res.py:

- converts the outputs of SIM and CEM to more readable outputs (eg per region),
- creates some additional and some synthetized output files (in results_simul/OUT or results_invest/OUT),
- creates graphs (in results_simul /IMG or results_invest/IMG),
- converts the outputs in IAMC format files (in results_simul /IAMC or results_invest/IAMC)
- creates a latex report files (in results_simul /LATEX or results_invest/LATEX)

Inputs:

- results of simulation in results_simul/ or in results_invest/
- results of capacity expansion (if -M invest) in results_invest/
- Bellman values in results_optim/

Outputs:

- *Detailed outputs*: Inputs and outputs of plan4res are converted in more readable files per regions:
  - Power demand: Demand-$region.csv,
  - Volumes in seasonal storage reservoirs: Volume-Reservoir-$region.csv,
  - Generation schedules of all assets: Generation-$region-$i.csv
  - Marginal costs of the demand constraint: MarginalCostActivePowerDemand-$region.csv, Histograms of the marginal costs: HistCmar-$region.csv
  - Marginal costs of the interconnections: MarginalCost-$reg2$reg.csv,
  - Imports and exports per region: ImportExport-$region-$i.csv and ImportExport$i.csv

- *Synthetic outputs*: Synthetised results are produced:

- (initial) Installed Capacity: InitialInstalledCapacity.csv, (new) aggregated installed capacity: AggrInstalledCapacity.csv, (new) Installed Capacity: InstalledCapacity.csv and invested Capacity: InvestedCapacity.csv,
- Non served energy per region: Slack-$region.csv, and number of hours with non served energy: nbHoursSlack.csv,
- Generation per technology: Generation.csv and Generation-$region.csv , and per aggregated technologies: AggrGeneration.csv
- Marginal costs: Histogram of scenarios of marginal costs: MonotoneCmar.csv, average on time of marginal costs: meanTimeCmar.csv, average on scenarios of marginal costs : meanScenCmar.csv,
- Average imports and exports: MeanImportExport.csv and meanImportExport-$region.csv

PostTreatPlan4res.py requires 4 configuration files:

- settingsPostTreatPlan4res_XXX.yml (see section)
- settings_format_XXX.yaml (see section 5.3.1.1)
- settingsCreateInputPlan4res_XXX.yml (see section 5.2.2.1) which is also the main configuration file of CREATE.
- VariablesDict.yml : contains the list of variables to retrieve and the correspondence between the plan4res and IAMC variable names (see section 5.2.2.3)

5.5.1 Filling the settingPostTreatPlan4res_XXX.yml configuration file

The configuration files settingsPostTreatPlan4res_simul.yml (for posttreating results of SIM) or settingsPostTreatPlan4res_invest.yml (for posttreating results of CEM) define how the results will be post-treated by POSTTREAT.

You may edit this file with any text editor. Change the following parameters:

- **BeginTreatData**: optional , use only if you want to posttreat on a subset of the time horizon
- **EndTreatData**: optional , use only if you want to posttreat on a subset of the time horizon
- (do not change) **Resultsdir**: sub-repository where all results are (in practice results_simul or results_invest)
- (do not change) **ScenarioIdentifier**: postfix of the output files of SIM and CEM as defined in SMS++
- **map**: yes if you wish to output graphs on maps
- **geopandas**: if you are not using the p4r-env environment, you may not have installed the python package geopandas (which is quite tricky to install) in that case choose no
- **private_map**: CSV file defining the boarders of the map, contains gdp data. It must include the countries you are working on.
- **PostTreat**:
  - DrawMean: True if you wish to include the average in the stochastic graphs
  - The following subsections: **Volume**, **Flows**, **Power**, **MarginalCost**, **MarginalCostFlows**, **Demand**, **InstalledCapacity** and **SpecifiedPeriods** are used to specify which treatment should be done for each category. The list of possible treatments is given for each category and you must write yes for allowing the treatment, or no.
    - **read**: read results and creates the posttreated csv files
    - **draw**: create graphs
    - **latex**: create latex report

        - **iamc**: converts results to IAMC format
        - (do not change) **Dir**: name of sub dir in results_invest or results_simul
    - SpecifiedPeriods allows the user to choose a list of scenarios and subperiods which will be analysed in details. It includes additional parameters:
        - **Scenarios**: subset of scenarios on which to perform detailed analysis
        - A subsection for each period to analyse, with a name chosen by the user. For each period, you need to fill **begin** with the first timestep of the period and **end** with the last timestep
- **Graphs**: this section allows to specify the dimensions of the graphics. It include one subsection for each of the following categories: **Volume**, **Power**, **MarginalCost**, **MarginalCostFlows** and **Demand.** For each category, the following parameters are required:
    - **nbcols**: number of columns in the graph which will be composed of subgraphs per regions or interconnections
    - **nblines**: number of rows in the graph which will be composed of subgraphs per regions or interconnections
    - **SizeCol**: size of the column in the aforementioned graph
    - **SizeRow**: size of the row in the aforementioned graph
    - **TitleSize**: size of the title in the aforementioned graph
    - **LabelSize**: size of the labels (columns and rows) in the aforementioned graph
    - (only for Power) **treat**: list of aggregated technologies to be included in the generation graphs
    - (only for Power) **ChloroGraph**: parameters for the graph composed of one map per aggregated technology, with shade varying depending on the intensity of this technology in each region:
        - **nbcols**: number of columns
        - **nblines**: number of rows
        - **dpi**: the default dpi produces very big jpeg files. Lowering it lowers the quality of the graph but also its size.
- **scenario**: name of Scenario to be used in Scenario column of the IAMC files containing outputs of plan4res (should be the name of the scenario used in CreateInputPlan4res followed by '|' and an additional identifier if necessary)
- **model**: name of Model to be used in Scenario column of the IAMC files containing outputs of plan4res (should be plan4res 2.0)
- **namereport**: name of the latex file which can be created
- **titlereport**: title of the latex report
- **usevu**: if yes, the cost of water used will be computed using the Bellman values computed by SSV
- **timestepvusms**: index of the SSV timestep to use for computation of the value of the water at the end of the period analysed by POSTTREAT
- **arrondi**: 1 if you wish to round all figures, else 0
- **marginalcostlimits**: allows to limit the graphs of marginal costs to a certain limit. You must provide a maximum (**max**) and a minimum (**min**)
- **pielimit**: POSTTREAT create graphs with pies. For very small quantities, the share of the pie will not be visible. This allows to not include shares of less than X in the graphs
- **Technos**: this section gives a color identifier for each technologies. All technologies in settingsCreatePlan4res_XXX.yml must be present, but this section can include additional

technologies. For each techno X that has pumping capacity, an additional techno X_PUMP must be added.

- **technosAggr**: this section allows to define aggregates for the technologies (e.g. gas or WindPower). These aggregates are used for creating additional graphs as well as synthetic outputs. This section must list the different aggregates and provide the following for each aggregate:
  - **technos**: list of the technologies included in this aggregate
  - **color**: color used in e.g. bargraphs
  - **colors**: code of a color range to use in the chloromaps
- **pumping**: provides the list of all technologies with storage and pumping capacity
- **nopumping**: provides the list of all technologies with storage but WITHOUT pumping capacity
- **graphVolumes**: parameters for creation of the graphs related to storage for the 'SpecificPeriods'. It includes one subsection per category (**Reservoir**, **PumpedStorage**, **Battery**), and for each category:
  - **Name**: title of the graph
  - **Technos**: list of technos to include

6 plan4res data formats

This section presents the format of plan4res datasets:

- IAMC format csv data
- plan4res csv input data
- plan4res NetCDF input data
- plan4res outputs

Details about the structure of plan4res and the workflow can be found in plan4resStructure.pdf (in your documentation directory or plan4res/documentation: plan4res documentation), in particular the location of the different files.

6.1 plan4res IAMC input data

The IAMC input data should be composed of one csv file whose name should be Dataset.xlsx. This file can be an output of the model GEneSYS-MOD. It will be used by the plan4res CREATE module to create plan4res csv input data files. It is also possible to adapt the configuration file of CREATE in order to use more than one IAMC file for retrieving the different variables (take some variables from one IAMC file, and other variables from another IAMC file).

These files have to follow the IAMC data format, described in the deliverable D4.2[22] (data exchange format and template) of open ENTRANCE, using the variables and regions defined in the open ENTRANCE nomenclature[23].

[22] Available at https://doi.org/10.5281/zenodo.5521098

[23] openENTRANCE/openentrance: Definitions of common terms (variables, regions, etc.) for the openENTRANCE project

Model	Scenario	Region	Variable	Unit	2018	2025	2030	2035	2040	2045	2050
GENeSYS-MOD.jl	Gondor_globalLimit_364	Harad	Capacity|Electricity|Hydro|Run of River	GW	7,6627	7,6627	7,6627	7,6627	7,6627	7,6627	7,6627
GENeSYS-MOD.jl	Gondor_globalLimit_364	Harad	Capacity|Electricity|Nuclear	GW						26,208	59,567
GENeSYS-MOD.jl	Gondor_globalLimit_364	Harad	Capacity|Electricity|Oil|w/o CCS	GW	14,456	10,69	10,41	0,5857	0,1376	0,0039	0,002
GENeSYS-MOD.jl	Gondor_globalLimit_364	Harad	Capacity|Electricity|Solar|PV|Utility	GW	20,384	57,374	72,403	84,514	96,33	128,44	128,44
GENeSYS-MOD.jl	Gondor_globalLimit_364	Harad	Capacity|Electricity|Wind|Onshore	GW	10,92	10,204	7,8154	4,5544	12,649	20,203	20,203
GENeSYS-MOD.jl	Gondor_globalLimit_364	Harad	Capital Cost|Electricity|Battery|Lithium-Ion	MEUR/GW	0,01	0,01	0,01	0,01	0,01	0,01	0,01
GENeSYS-MOD.jl	Gondor_globalLimit_364	Harad	Capital Cost|Electricity|Battery|Redox Flow	MEUR/GW	0,01	0,01	0,01	0,01	0,01	0,01	0,01
GENeSYS-MOD.jl	Gondor_globalLimit_364	Harad	Capital Cost|Electricity|Biomass|w/ CCS	MEUR/GW	0,01	0,01	0,01	0,01	0,01	0,01	0,01

Table 1: example of IAMC input data used by script `CreateInputPlan4res.py` to create the plan4res files

6.2 Time series

- **A set of CSV files containing scenarized time serie**s: those files follow a common format for time series: the first row is the header, the first column contains the UCT timestamp (in any format readable by Python pandas) and the following columns are different scenarios of the current timeseries. One CSV file contains only timeseries for One variable. Scenarios are identified by the corresponding column name in the header line (for example, past years, eg 1970, 1971, 1972…);

Table 2: Example of stochastic timeserie

Timestamp [UTC]	Base	PVminus10	Demandplus10
01/01/2050 00:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 01:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 02:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 03:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 04:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 05:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 06:00	3.31E-05	3.31E-05	3.64E-05
01/01/2050 07:00	4.91E-05	4.91E-05	5.40E-05
01/01/2050 08:00	6.51E-05	6.51E-05	7.16E-05
01/01/2050 09:00	8.11E-05	8.11E-05	8.92E-05

- **One (optional) unique CSV file containing all deterministic timeseries**: deterministic timeseries have to be present in a single csv file, whose name has to be given in the settingsCreateInputPlan4res_*xxx*.yml or in the settings_format_XXX.yml configuration file. This file follows the common format for time series: the first row is the header, the first column contains the UCT timestamp (in any format readable by Python pandas) and the following columns correspond to each deterministic timeseries, the name of each deterministic timeseries being in the header.

Table 3: Deterministic time series

Timestamp [UTC]	**TS_XX**	**TS_YY**	**TS_ZZ**
01/01/2050 00:00	2539999.79	235940.523	1846508.33
01/01/2050 01:00	2330728.6	202206.466	1743163.67
01/01/2050 02:00	2400311.81	203914.562	1762293.74
01/01/2050 03:00	2373790.71	200641.82	1772480.46
01/01/2050 04:00	2242430.18	178585.198	1578006.06
01/01/2050 05:00	2139597.06	167443.384	1418207.69
01/01/2050 06:00	2524088.67	198991.4	1717258.86
01/01/2050 07:00	2466338.43	200655.869	1819359.65

01/01/2050 08:00	2339799.73	202049.705	1839468.73
01/01/2050 09:00	2360725.92	201430.798	1834015.73
01/01/2050 10:00	2311079.56	195961.933	1815442.03

6.3 Plan4res csv Input Data

This section describes the format of plan4res input data. A dataset is composed of 7 csv files representing the "fixed" data and of a number timeseries (described in section 6.2).

All the CSV input files are following.

1. One row containing column labels
2. Serie of Rows containing the data (consistent with column labels)
3. Each row contains a name, a zone and various values of variables. Zone can be any level of geographical partition (see below).

Columns that are not used may be skipped.

Important notice: within plan4res, the convention for the units is that all data and results are in MW, MWh, €, €/MW, €/MWh depending on the variable (see **Erreur ! Source du renvoi introuvable. Erreur ! Source du renvoi introuvable.**)

The dataset is composed of:

- *ZP_ZonePartition.csv*: contains the description of the different regions;
- *ZV_ZoneValues.csv*: contains the data linked to the regions (in particular the demand);
- *IN_Interconnections.csv*: contains the description of the network;
- *TU_ThermalUnits.csv*: contains the description of the thermal power plants (including nuclear) and of any asset which can be modeled in plan4res as a thermal plant (even if not using any fossil fuel)
- *SS_SeasonalStorage.csv*: contains the description of hydropower with seasonal storages and other long-term seasonal storages;
- *STS_ShortTermStorage.csv*: contains the description of all the short-term storages, including eg. pumped hydro, batteries but it can also be used for demand-response mechanisms such as load-shifting;
- *RES_RenewableUnits.csv*: contains the description of PV, wind power and run-of-river technologies.

6.3.1 Geography and coupling constraints : files ZP_ZonePartition, ZV_ZoneValues, IN_Interconnections

This first group of data files is used to describe the regions in the dataset, the interconnections between the regions, and the constraints at region level, such as the power demand, but also the system services.

6.3.1.1 File ZP_ZonePartition.csv

The file ZP_ZonePartition.csv describes the different regions that are used in the dataset. In the example below, the lower level (on the left) corresponds to countries.

The lower level is used to define **Nodes**. Each generation unit (defined in TU_ThermalUnits.csv, RES_RenewableUnits.csv, STS_ShortTermUnits.csv and SS_SeasonalStorage.csv) belongs to one **Node**. Interconnections (defined in IN_Interconnections.csv) are connecting nodes.

The higher level corresponds to continents (with a unique continent called 'MiddleEarth'). There could exist intermediate levels. The different levels are used to define "coupling constraints" (see ***Erreur ! Source du renvoi introuvable. Erreur ! Source du renvoi introuvable.***), such as the power demand which is always linked to a Node. Other coupling constraints (such as system services) may apply to different levels. (See ***Erreur ! Source du renvoi introuvable. Erreur ! Source du renvoi introuvable.*** for more details about regions)

Table 4: ZP_ZonePartition.csv

Countries	Continent
RoGonDor	MiddleEarth
DolAmroth	MiddleEarth
Harad	MiddleEarth

The different regions can be defined by the user in the settingsCreateInputPlan4res_*xxx*.yml configuration file (Regions section).

6.3.1.2 File ZV_ZoneValues.csv

This file contains the values of all coupling constraints (in particular the demand at each Node, which is mandatory). It may also optionally contain the parameters for defining the costs associated to imbalance (for each coupling constraint). These parameters can also be defined in the settingsCreateInputPlan4res_*xxx*.yml configuration file (Coupling constraints section), in particular if the user wishes to use the same parameters for all the regions. If the user wishes to use different values, they must appear in ZV_ZoneValues.csv.

Table 5: ZV_ZoneValues.csv

Type	Zone	Value	Profile_Timeserie
Cooking	DolAmroth	7178950.652	TS_COOKING_DolAmroth.csv
ElecHeating	DolAmroth	54635950.13	TS_HEAT_LOW_DolAmroth.csv
ElecVehicle	DolAmroth	13541851.26	TS_MOBILITY_PSNG_DolAmroth.csv
OtherExclHeatTranspCooking	DolAmroth	51369219.81	TS_LOAD_DolAmroth.csv
CostActivePowerDemand	DolAmroth	10000	
MaxActivePowerDemand	DolAmroth	1500000	

Table 5 shows the content of ZV_ZoneValues.csv for the region "DolAmroth". In this example, tere is only one type of coupling constraint per region, which is the power demand. The power demand is composed of 4 parts: power demand for cooking, for heating (ElecHeating), for transportation (ElecVehicle) and for other uses (OtherExclHeatTranspCooking). The unit of the values in the column 'value' is MWh or €/MWh (for the variables of Type Cost*). The row 'Cooking' gives in column 'value' the power demand for cooking in the region DolAmroth for the year, in MWh, as well as the name of the timeseries to use to compute the hourly power demand for cooking (here TS_COOKING_DolAmroth.csv). Whenever the timserie would be deterministic, the Profile_Timeserie column would contain the name of the column corresponding to this timeseries in the csv file containing all deterministic timeseries. Table 6 shows the first 6 hours of this timeseries, for the 3 scenarios 'Base', 'PVminus10' and 'Demandplus10'). The (here scenarized) hourly power demand for cooking is computed by multiplying the timeseries by the numerical value in ZV_ZoneValue.csv.

Table 6: TS_COOKING_DolAmroth.csv

Timestamp [UTC]	Base	PVminus10	Demandplus10
01/01/2050 00:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 01:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 02:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 03:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 04:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 05:00	1.60E-05	1.60E-05	1.76E-05
01/01/2050 06:00	3.31E-05	3.31E-05	3.64E-05

The total power demand (scenarized) timeseries of the region is computed by adding the 4 timeseries corresponding to the 4 parts of the power demand. The way of computing the power demand is defined in the settingsCreateInputPlan4res_*xxx*.yml configuration file. Figure 7 shows the total power demand in the region DolAmroth, as it is computed by plan4res out of the values in column value of ZV_ZoneValues.csv and the different timeseries in column Profile_Timeseries.

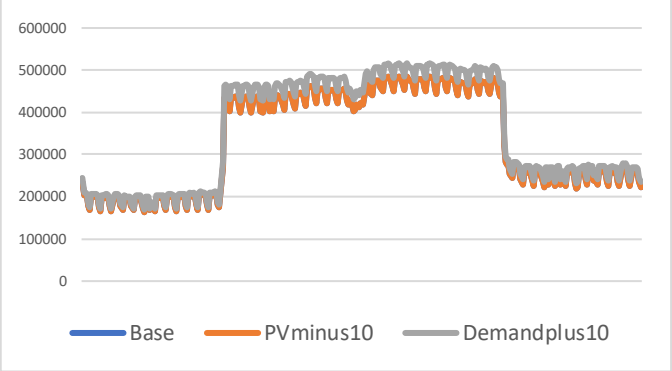


Figure 7: Power demand scenarios in region DolAmroth

6.3.1.3 File IN_Interconnections.csv

This file contains the description of the network, ie the set of **lines** connecting the different **nodes**. (remind that the nodes are described in the first column of ZP_ZonePartition.csv).

Table 7: IN_Interconnectionscsv

Name	StartLine	EndLine	MaxPowerFlow	MinPowerFlow
DolAmroth>Harad	DolAmroth	Harad	71	-100
RoGonDor>DolAmroth	RoGonDor	DolAmroth	2200	-2000
RoGonDor>Harad	RoGonDor	Harad	866	-1160

In_Interconnections.csv may contain the following columns (optional columns are highlighted): (*Note that all numerical values may be replaced by the name of a deterministic Timeserie*):

- **Name**: name of the line (used for processing results)
- **StartLine** and **EndLine** must be nodes defined in the first column of ZP_ZonePartition.csv.
- **MaxPowerFlow** and **MinPowerFlow** are the bounds in MW on the flows for this line (one way or the other); *MaxPowerFlow* is the maximal flow from Start to End, while *MinPowerFlow* (which can be negative) is the minimum flow from Start to End. *(-*

*1)*MinPowerFlow* is also the maximum flow between End and Start while *(-1)*MaxPowerFlow* is the minimum flow between Start and End.

- **Impedance** (optional, default 0): Impedance of the line
- **Cost** (optional, default 0): Cost of the line in €/MWh

6.3.2 Description of generation assets : files TU_ThermalUnits, SS_SeasonalStorage, STS_ShortTermStorage and RES_RenewableUnits

These data files allow to describe the different kinds of generation units. It also allows to describe some load control mechanisms, such as load shifting, or management of electric vehicle charging.

6.3.2.1 File TU_ThermalUnits.csv

This files gives the characteristics of all thermal power plants (and of all the power plants which are modelled as thermal power plants in plan4res, such as Geothermal plants).

Table 8: TU_ThermalUnits.csv

Zone	Name	NumberUnits	MaxPower	VariableCost	CO2Rate
RoGonDor	Biomass|w/o CCS	1	19573.5	3.816	0
DolAmroth	Biomass|w/o CCS	1	367.12	3.816	0
Harad	Biomass|w/o CCS	1	469.141614	3.816	0
RoGonDor	Coal|Hard coal|w/o CCS	1	16445.9549	3	0.7512
DolAmroth	Coal|Hard coal|w/o CCS	1	11.923392	3	0.7512
Harad	Coal|Hard coal|w/o CCS	1	1783.37195	3	0.7512
RoGonDor	Coal|Lignite|w/o CCS	1	5339.78409	3	0.94971429
RoGonDor	Gas|CCGT|w/o CCS	1	83694.0023	81.6648775	0.34634483
DolAmroth	Gas|CCGT|w/o CCS	1	2000	82.0429943	0.34634483
Harad	Gas|CCGT|w/o CCS	1	15000	81.517916	0.34634483
RoGonDor	Gas|OCGT|w/o CCS	1	22000	113.409441	0.52863158
Harad	Gas|OCGT|w/o CCS	1	25000	113.205328	0.52863158
RoGonDor	Nuclear	1	50783.1831	8	0
DolAmroth	Nuclear	1	5000	8	0
Harad	Nuclear	1	40000	8	0
RoGonDor	Oil|w/o CCS	1	8652.37627	0.036	0.70105263
Harad	Oil|w/o CCS	1	3.94960304	0.036	0.70105263

It contains the following data (*Note that some of the numerical values may be replaced by the name of a deterministic Timeserie*. In that case the timeseries must be present in the Deterministic timeseries CSV file (see):

- **Name**: name of the technology, or name of the plant. The list of names must be present in the settingsCreateInputPlan4res_*xxx*.yml configuration file (technos section, thermal), as this list is used in particular for post-treatments of results. Whenever the user wishes to add a new technology or plant, it must be added to the list in the settingsCreateInputPlan4res_*xxx*.yml configuration file.
- **Zone**: the values in this column must be included in the list of **nodes**, which is given in the first column of ZP_ZonePartition.csv, and corresponds to the lowest granularity of regions.
- **NumberUnits**: number of units of the given characteristics.
- **MaxPower**: maximum power of a single unit in MW. When NumberUnits=1, MaxPower is equal to Capacity.
- **Profile_Timeserie** (optional): deterministic or stochastic timeseries which will be applied to MaxPower. This allows accounting for eg. Maintenances and outages.

- **MinPower** (optional, 0 by default)[24] : minimum power of a single unit in MW. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **Pauxiliary** (optional, 0 by default): Power (MW) taken from the system when off for each unit. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **VariableCost** (optional, 0 by default): proportional cost in €/MWh. Can be a deterministic timeseries.
- **FixedCost** (optional, 0 by default): fixed cost in €. Can be a deterministic timeseries.
- **Quadterm** (optional, 0 by default): quadratic cost. Can be a deterministic timeseries.
- **StartUpCost** (optional, 0 by default): cost (€) for starting the unit after a shut down. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **MinUpTime** (optional, 1 hour by default): minimum duration when the plant is on (number of hours). *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **MinDownTime** (optional, 1 hour by default): minimum duration when the plant is off (number of hours). *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **Inertia** (optional, 0 by default): maximum inertia that can be provided by a unit in MWs/MWA. Can be a deterministic timeseries.
- **PrimaryRho** (optional, 0 by default): this parameter, multiplied by Maxpower, gives the maximum share of the active power that can be used as primary reserve. Can be a deterministic timeseries.
- **SecondaryRho** (optional, 0 by default): this parameter, multiplied by MaxPower, gives the maximum share of the active power that can be used as secondary reserve (optional, 0 by default). Can be a deterministic timeseries.
- **DeltaRampDown** (optional, MaxPower by default): maximum gradient when the power is decreased from one time step to the other, MW per hour. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **DeltaRampUp** (optional, Maxpower by default): maximum gradient when the power is increased from one time step to the other, MW per hour. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **CO2** (optional, 0 by default): emission rate in tons CO2 per MWh
- **Any other pollutant** which is defined in the coupling constraints section of the settingsCreateInputPlan4res_*xxx*.yml configuration file.

6.3.3 File SS_SeasonalStorage.csv

This file gives the characteristics of all seasonal storages. Only aggregated seasonal storages (ie one with one reservoir) can be described in this file (although SMS++ is capable of handling complex hydrovalleys).

[24] Note that data consistency is ensured: if MinPower > MaxPower at some point, then MinPower = MaxPower. For example, this is useful if MaxPower = 0 during a period to model a maintenance or an outage. For thermal units, MinPower constraint applies only if the unit is started.

Table 9: SS_SeasonalStorage.csv

Name	Zone	Hydro System	Number Units	Max Power	Min Power	Max Volume	Min Volume	Inflows	Inflows Profile	Initial Volume	Turbine Efficiency	Pumping Efficiency
Hydro|Reservoir	RoGonDor	0	1	25814.05	0	51628100	0	7.5E+07	Inflows_RoGonDor.csv	34418733.3	1	0
Hydro|Reservoir	DolAmroth	0	1	5524.2394	0	11048478.9	0	1.6E+07	Inflow_DolAmroth.csv	6629087.33	1	0
Hydro|Reservoir	Harad	0	1	6000	0	12000000	0	2.8E+07	Inflow_Harad.csv	6000000	1	0

It contains the following columns (*Note that all numerical values may be replaced by the name of a deterministic Timeserie*):

- **Name**: name of the technology, or name of the plant. The list of names must be present in the settingsCreateInputPlan4res_*xxx*.yml configuration file (technos section, reservoir), as this list is used in particular for post-treatments of results. Whenever the user wishes to add a new technology or plant, it must be added to the list in the settingsCreateInputPlan4res_*xxx*.yml configuration file.
- **Zone**: the values in this column must be included in the list of **nodes**, which is given in the first column of ZP_ZonePartition.csv, and corresponds to the lowest granularity of regions.
- **NumberUnits**: number of units of the same type at the same location.
- **MaxPower** : maximum power of a single unit in MW. When NumberUnits=1, MaxPower is equal to Capacity. Can be a deterministic timeserie.
- **MinPower**[25] (optional, 0 by default) [26] : minimum power of a single unit in MW. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **DeltaRampDown** (optional, MaxPower by default): maximum gradient when the power is decreased from one time step to the other, MW per hour. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **DeltaRampUp** (optional, Maxpower by default): maximum gradient when the power is increased from one time step to the other, MW per hour. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **MaxVolume** : maximum volume of the reservoir (MWh). Can be a deterministic timeseries.
- **MinVolume**[27] (optional, 0 by default) : minimum volume of the reservoir (MWh). Can be a deterministic timeseries.
- **TurbineEfficiency**: (optional, 1 by default). This value, multiplied by the flow, gives the generated power.
- **PumpingEfficiency**: (optional, 1 by default, but should be lower than TurbineEfficiency in practice). This value, multiplied by the flow, gives the generated power.
- **Inflows**: (optional, 0 by default; mandatory if column InflowsProfile is present). Inflows to the upstream reservoir (energy per year in MWh).
- **InflowsProfile**: (optional): time series profile for Inflows. Multiplied by the Inflows in energy, gives the inflows time series. These timeseries may be stochastic (in that case the value has to be XXX.csv), or deterministic.
- **InitialVolume**: (optional, 0 by default). Initial Volume of the upstream reservoir (MWh)

[25] As in ThermalUnits, consistency between MinPower and MaxPower is ensured such that Minpower can never be greater than MaxPower.

[26] Note that for hydro storages, MinPower forces the plant to run: it aims at representing hydro operational constraints such as minimal river flows. It is different from the MinPower constraint applied to thermal units.

[27] Consistency between MinVolume and MaxVolume is ensured, such that MinVolume can never be greater than MaxVolume.

- **Inertia** (optional, 0 by default): maximum inertia that can be provided by a unit in MWs/MWA. Can be a deterministic timeseries.
- **PrimaryRho**: (%) this parameter, multiplied by MaxPower, gives the maximum primary reserve that can be provided by a unit (optional, 0 by default). Can be a deterministic timeserie.
- **SecondaryRho**: (%) this parameter, multiplied by MaxPower, gives the maximum secondary reserve that can be provided by a unit (optional, 0 by default). Can be a deterministic timeserie.
- **WaterValues**: optional; used to specify water values as input data from a file, if they are/were not computed by the current run (for example, coming from a previous SSV run). Contains the name of the file/sheet where the water values are stored. When using the simulation mode with one Bellman values (BV) file for all units (ie a polyhedral function), this file is only given in the first line; when using the simulation mode with 1 BV per unit (ie one function per reservoir), this file is given at each line; in optimization mode (= when running the SSV), this is not used as computed by the model. This column is always optional as the BV file (in case it is a polyhedral function) can be passed to SMS++ when calling the solver.

6.3.4 File STS_ShortTermStorage.csv

This file is used for:

- Pumped hydro
- Batteries
- Other flexibilities such as residential flexibilities (in particular load shifting) or electric vehicles (EV), which are modelled as short-term storages.

Table 10: STS_ShortTermStorage.csv

Name	Zone	NumberUnits	MaxPower	MinPower	MaxVolume	MinVolume	TurbineEfficiency	PumpingEfficiency
Hydro|Pumped Storage	RoGonDor	1	12150.54	-12150.54	1215054	0	0.866	0.866
Hydro|Pumped Storage	DolAmroth	1	3302	-3302	330200	0	0.866	0.866
Hydro|Pumped Storage	Harad	1	5920.72	-5920.72	592072	0	0.866	0.866
Battery|Lithium-Ion	RoGonDor	1	142775.466	-14275.47	571101.864	0	0.95	0.95
Battery|Lithium-Ion	DolAmroth	1	3915.34909	-391.3491	15661.3964	0	0.95	0.95
Battery|Lithium-Ion	Harad	1	27246.1193	-27246.119	108984.477	0	0.95	0.95

This file contains the following data:

- **Name**: name of the technology, or name of the plant. The list of names must be present in the settingsCreateInputPlan4res_*xxx*.yml configuration file, as this list is used in particular for post-treatments of results. Whenever the user wishes to add a new technology or plant, it must be added to the list in the settingsCreateInputPlan4res_*xxx*.yml configuration file.
- **Zone**: the values in this column must be included in the list of **nodes**, which is given in the first column of ZP_ZonePartition.csv, and corresponds to the lowest granularity of regions.
- **NumberUnits**: number of units of the same type at the same location.
- **MaxPower**: maximum power of a single unit in MW. When NumberUnits=1, MaxPower is equal to Capacity.
- **MaxPowerProfile** (optional): deterministic timeseries, which will be multiplied by MaxPower to obtain the MaxPower timeseries in MW.
- **MinPower** (optional, 0 by default) [28] [29]: minimum power of a single unit in MW. Can be a deterministic timeserie. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*

[28] Note that for hydro storages, MinPower forces the plant to run: it aims at representing hydro operational constraints such as minimal river flows. It is different from the MinPower constraint applied to thermal units.
[29] Consistency between MinPower and MaxPower is ensured, MinPower can never be greater than MaxPower.

- **MaxVolume**[30]: maximum volume of the reservoir (MWh)..
- **MaxStorageProfile** (optional): deterministic timeseries, which will be multiplied by MaxVolume to obtain the MaxVolume timeserie in MWh.
- **MinVolume** (optional, 0 by default): minimum volume of the reservoir (MWh). Can be a deterministic timeserie.
- **TurbineEfficiency** (optional, 1 by default): This value, multiplied by the flow, gives the generated power. Can be a deterministic timeserie.
- **PumpingEfficiency** (optional, 1 by default, but should be lower than TurbineEfficiency in practice): This value, multiplied by the flow, gives the generated power. Can be a deterministic timeserie.
- **Inflows** (optional, 0 by default):
- **InitialPower** (optional, 0 by default): Initial Power of the unit (MW)
- **InitialStorage** (optional, 0 by default): Initial Volume of the upstream reservoir (MWh)
- **Inertia** (optional, 0 by default): maximum inertia that can be provided by a unit in MWs/MWA. Can be a deterministic timeserie.
- **MaxPrimaryPower** (optional, 0 by default): maximum primary reserve that can be provided by a unit. Can be a deterministic timeserie.
- **MaxSecondaryPower** (optional, 0 by default): maximum secondary reserve that can be provided by a unit (optional, 0 by default). Can be a deterministic timeserie.
- **DeltaRampDown** (optional, MaxPower by default): maximum gradient when the power is decreased from one time step to the other, MW per hour. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **DeltaRampUp** (optional, Maxpower by default): maximum gradient when the power is increased from one time step to the other, MW per hour. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **Cost** (optional, 0 by default): proportional cost (€/MWh). Can be a deterministic timeseries.
- **Inflows**: (optional, 0 by default). Inflows to the upstream reservoir (MWh). If negative it is considered to be a consumption linked to the unit (eg consumption of EV). Can be a deterministic timeseries.
- **VolumeLevelTarget**[31]: (optional) used to force the optimization to reach this volume at the end of each stage. If there is a VolumeLevelTarget, the minimum volume constraint is replaced by this value at the first and last time-steps of each stage.

6.3.4.1 File RES_RenewableUnits.csv

This file gives the characteristics of the variable renewable units: windpower, PV power and run-of-river units.

Table 11: RES_RenewableUnits.csv

Name	Zone	NumberUnits	MaxPower	MinPower	MaxPowerProfile
Wind|Onshore	DolAmroth	1	9923.33	0	ONSHORE_OPT_DolAmroth.csv
Wind|Offshore	RoGonDor	1	18113.33	0	Offshore-Deep_RoGonDor.csv

It contains the following data:

[30] Consistency between MinVolume and MaxVolume is ensured. MinVolume can never be greater than MaxVolume

[31] If an InitialStorage is not provided, VolumeLevelTarget will be used as InitialStorage.

- **Name**: name of the technology, or name of the plant. The list of names must be present in the settingsCreateInputPlan4res_*xxx*.yml configuration file (technos section, thermal), as this list is used in particular for post-treatments of results. Whenever the user wishes to add a new technology or plant, it must be added to the list in the settingsCreateInputPlan4res_*xxx*.yml configuration file.
- **Zone**: the values in this column must be included in the list of **nodes**, which is given in the first column of ZP_ZonePartition.csv, and corresponds to the lowest granularity of regions.
- **NumberUnits**: number of units of the same type at the same location.
- **MaxPower**: capacity for PV and WindPower; yearly average energy for run-of-river) – This is due to the fact that the available load factor timeseries for Wind and PV Power and for run-of-river were computed with different methods. The maximum potential production for Wind or PV is equal to the capacity multiplied by the load factor timeseries while the maximum potential production for a run-of-river is equal to the average yearly energy multiplied by the timeseries.)
- maximum power of a single unit in MW. When NumberUnits=1, MaxPower is equal to Capacity.
- **MaxPowerProfile** (optional): deterministic or stochastic timeseries which will be applied to MaxPower. This allows accounting for in particular the variability of the potential power, given its correlation with climate variables (sun, wind…).
- **MinPower** (optional, 0 by default)[32] : minimum power of a single unit in MW. Can be a deterministic timeseries. *This constraint is relaxed in SSV and CEM, as well as in SIM when the user requires to use continuous relaxation (see parameter in SMS++ configuration files)*
- **Inertia** (optional, 0 by default): maximum inertia that can be provided by a unit in MWs/MWA. Can be a deterministic timeserie.
- **Gamma** (optional, 1 by default): this parameter is used by the model to determine the maximum available primary and secondary reserve. In practice, the model accounts for the following constraint: at each timestep, the sum of the primary and secondary reserves is lower than the maximum power minus the generated power.

6.3.5 Additional columns for dealing with Investments (CEM)

To run the Capacity Expansion Model (CEM), 4 columns need to be added to the files corresponding to assets in which one wants to invest (see below), ie to the files describing the generation units (apart from Seasonal Storages) as well as to the file describing the interconnection (see):

Table 12: additional columns for capacity expansion

Zone	Name	….	InvestmentCost	DecommissionCost	MaxAddedCapacity	MaxRetCapacity
DolAmroth	Biomass|w/o CCS		1980000	100	5000	0
RoGonDor	Biomass|w/o CCS		1980000	100	4000	0
Harad	Biomass|w/o CCS		1980000	100	500	0
DolAmroth	Coal|Hard coal|w/o CCS		1082250	100	0	2000

- **MaxAddedCapacity**: this is the maximum capacity that may be added, in MW.
- **MaxRetCapacity**: this is the maximum capacity that may be decommissioned, in MW.

[32] Note that data consistency is ensured: if MinPower > MaxPower at some point, then MinPower = MaxPower. For example, this is useful if MaxPower = 0 during a period to model a maintenance or an outage. For thermal units, MinPower constraint applies only if the unit is started.

- **InvestmentCost**: this is the cost for investing into the given capacity in the given zone, in €/MW. Note that these are yearly costs, computed as Capital Cost / LifeTime (in years) + Fixed Cost.
- **DecommissionCost**: this is the cost for decommissioning the given capacity in the given zone, in €/MW.

Costs are not discounted.

These columns may then be added to the following files:

- IN_Interconnections.csv
- TU_ThermalUnits.csv
- STS_ShortTermStorage.csv
- RES_RenewableUnits.csv

Note that investment in new seasonal storages is not available at present.

6.4 SMS++ input data: the NetCDF files

The CSV static data files and timeseries are converted into the following NetCDF files, following the specifications[33] of those files defined by SMS++.

3 different kinds of NetCDF files are created:

- Block_$i.nc4: these files (one per SSV timestep) are describing the power system in the current timestep
- SDDPBlock.nc4: this file is describing the SSV problem, including all stochastic scenarios
- InvestmentBlock.nc4: this file is describing the CEM problem, with in particular identification of the assets which can be invested and related constraints

6.5 Plan4res outputs

The results of plan4res are detailed in the following subsections

6.5.1 Results of SSV

The SSV runs a Stochastic Dynamic Dual Programming (SDDP) algorithm to compute Bellman values for all the seasonal storages.

Note: Bellman values represent the *cost-to-go functions* = the expected economic value associated to the various levels of the seasonal storages at each time stages. They are usually represented as sets of hyperplanes called *cuts*.

The results are:

- When the convergence criteria of the SDDP algorithm is met[34]: *
    - BellmanValuesOUT.csv: redundant cuts have been pruned out.
    - BellmanValuesAllOUT.csv: contains all the cuts found by the algorithm.

[33] https://gitlab.com/smspp/smspp-project/-/raw/develop/doc/SMS++%20File%20Format%20Manual/ffm.pdf?ref_type=heads

[34] See the plan4res run guide for more information.

- In any case, ie even without convergence: cuts.txt, which contains all the cuts already found by the SDDP algorithm. This is useful to help reach convergence since it is possible to launch the SSV again using the cuts from a previous run stored in cuts.txt as a hot start.

The format of this file is, as shown in Table 13, aiming to represent a polyhedral function at each SSV timestep. The first column (Timestep) gives the index of the SSV Timestep. There are a number of rows for each timestep, representing the different 'cuts' of the function at this timestep. The function at timestep $i is: $\max_{cuts}\left(\sum_{j=0}^{R} a\_j + b\right)$, where a_j and b are the values in the corresponding columns, and R is the number of reservoirs (the number of columns a_j).

Table 13: BellmanValuesOUT.csv

Timestep	a_0	a_1	a_2	b
0	0	0	0	3.9825E+10
0	-113.40944	-10000	-10000	7.4137E+10
0	-113.40944	-10000	-81.517916	1.2033E+11
0	-81.786066	-3	-150.94929	5.6955E+10
0	-0.036	-3.816	-8	5.3972E+10
0	-8	-3	-113.20533	5.5282E+10
0	-0.03249	-0.03249	-8	5.4175E+10
0	-0.036	-10000	-81.517916	1.1965E+11
0	-0.036	-8	-81.517916	5.5241E+10
1	0	0	0	3.9637E+10
1	-113.40944	-10000	-10000	1.3613E+11

6.5.2 Results of SIM

The simulation produces 2 sets of results:

- Raw results of the simulation. They correspond to the outputs of the model, without any post-treatment (the post-treatment is performed when running POSTTREAT).
- Post-treated results. These are results computed by the POSTTREAT function of plan4res.

6.5.2.1 Raw results of SIM

All row results follow the same format: the index column gives the indexes of the simulation timesteps, while the other columns give the values corresponding to the name in the header (which can be a region, a technology….)

For each scenario $i, the following files are produced.

- Demand_Scen$i_OUT.csv: 1 column per node with the power demand in the node.
- ActivePower_Scen$i_OUT.csv: generation schedules. 1 column per unit (apart from the Seasonal Storage units which are duplicated in 2 columns: one for the generation part, and the other one for the pumping part). Note that all other units with storages comprise a unique column with both positive and negative (pumping) values.
- Primary_Scen$i_OUT.csv: primary reserve schedules. 1 column per unit (apart from the Seasonal Storage units which are duplicated in 2 columns: one for the generation part, and the other one for the pumping part).

- Secondary_Scen$i_OUT.csv: secondary reserve schedules. 1 column per unit (apart from the Seasonal Storage units which are duplicated in 2 columns: one for the generation part, and the other one for the pumping part)
- Inertia_Scen$i_OUT.csv : inertia provided by each unit. 1 column per unit (apart from the Seasonal Storage units which are duplicated in 2 columns: one for the generation part, and the other one for the pumping part).
- Volume_Scen$i_OUT.csv: volumes of each storage (seasonal storages and short-term storages).
- Flows_Scen$i_OUT.csv: flows between zones. 1 column per transmission line.
- MarginalCostActivePowerDemand_Scen$i_OUT.csv: marginal costs of the demand constraint. 1 column per node (ch the *ActivePowerDemand* constraint is always applied to Nodes).
- MarginalCostPrimary_Scen$i_OUT.csv: marginal costs of the primary reserve constraint. 1 column per region (at the partition level of the *PrimaryDemand* constraints, note that these constraints may apply to different regional partitions than the Demand. See section **Erreur ! Source du renvoi introuvable.**).
- MarginalCostSecondary_Scen$i_OUT.csv: marginal costs of the secondary reserve constraint. 1 column per zone (at the partition level of the *SecondaryDemand* constraints).
- MarginalCostInertia_Scen$i_OUT.csv: marginal costs of the Inertia constraint (at the partition level of the *InertiaDemand* constraints). 1 column per zone.
- MarginalCostFlows_Scen$i_OUT.csv: marginal costs of the lines. 1 column per line.
- MaxPower_Scen$i_OUT.csv: maximum available power. 1 column per technology.

All files share the same format: a header containing the names of the series, an index with the time steps.

6.5.2.2 Post-Treated resuls of SIM

6.5.2.2.1 Installed Capacity

InstalledCapacity.csv and AggrInstalledCapacity.csv: installed capacities (aggregated installed capacity, with aggregations such as defined in `settingsPostTreatPlan4res.py`) in the different available technologies per zone. 1 column per technology, 1 row per node. (the generation units are attached to nodes)

Table 14: InstalledCapacity.csv

	Hydro|Reservoir	Biomass|w/o CCS	Coal|Hard coal|w/o CCS	Coal|Lignite|w/o CCS	Gas|CCGT|w/o CCS
RoGonDor	25814.05	19573.5	16445.95488	5339.784093	83694.0023
DolAmroth	5524.239444	367.12	11.923392	0	2000
Harad	6000	469.1416144	1783.371951	0	15000

6.5.2.2.2 Generation

Generation.csv and AggrGeneration.csv: total average (on scenarios) generation on the whole simulation time horizon (aggregated generation, with aggregations such as defined in `settingsPostTreatPlan4res_XXX.py`) from the different available technologies per node. 1 column per technology, 1 row per node. Note that the Non Served column corresponds to the non-served electricity.

Table 15: AggrGeneration.csv

	Hydro	WindPower	PV	Biomass	Nuclear
RoGonDor	126850389	578839017	185813311	0	249852103
DolAmroth	29932305.8	26161752.2	11697614.5	2944912.11	29463408.7
Harad	37280301.6	10847753.5	117436876	4098421.14	346617336

Generation-NODE.csv and ***AggrGeneration-NODE.csv***: average generation (aggregated generation) from the different available technologies per node. 1 file per node, 1 column per technology, 1 row per simulation time step.

Table 16: Generation-Harad.csv

	Hydro|Reservoir	Biomass|w/o CCS	Coal|Hard coal|w/o CCS
02/07/2030	886.9210309	11259.39875	42800.92681
03/07/2030	0	11259.39875	42800.92681
04/07/2030	0	11259.39875	42800.92681
05/07/2030	0	11259.39875	42800.92681
06/07/2030	0	11259.39875	42800.92681
07/07/2030	0	11259.39875	42800.92681
08/07/2030	0	11259.39875	42800.92681
09/07/2030	19775.91482	11259.39875	42800.92681

Generation-NODE-$i.csv: generation from the different available technologies in the node for scenario $i. 1 file per zone and scenario, 1 column per technology, 1 row per time step.

Slack-ZONE.csv: electricity not served per node. 1 file per node. 1 column per scenario, 1 row per time step.

Table 17: Slack-Harad.csv

	Harad-0	Harad-1	Harad-2	Harad-3
02/07/2030	0	0	0	0
03/07/2030	0	0	0	0
04/07/2030	0	0	0	0
05/07/2030	0	0	0	0
06/07/2030	0	0	0	0
07/07/2030	0	0	0	0

nbHoursSlack.csv: number of hours with non-served electricity per node and scenario.

6.5.2.2.3 Costs

MeanVariableCost.csv: average variable costs. 1 column per node, 1 row per technology.

Table 18: MeanVariableCost.csv

	RoGonDor	DolAmroth	Harad
Hydro|Reservoir	0	0	0
Biomass|w/o CCS	0	11237784.6	15639575.1
Coal|Hard coal|w/o CCS	330248836	305191.142	46738612.1
Coal|Lignite|w/o CCS	107785456	0	0

VariableCostPerScenario.csv: variable costs. 1 column per scenario and node, 1 row per techno.

Table 19: VariableCostPerScenario.csv

	RoGonDor-0	RoGonDor-1	RoGonDor-2	RoGonDor-3	DolAmroth-0
Hydro|Reservoir	0	0	0	0	0
Biomass|w/o CCS	0	0	0	0	11073780.1

6.5.2.3 Marginal Costs

meanScenCmar.csv: average over all scenarios of the marginal costs. 1 column par node, 1 row per time step.

Table 20: meanScenCmar.csv

	RoGonDor	DolAmroth	Harad
02/07/2030	0.036	8	81.517916
03/07/2030	0.036	7.415	81.517916
04/07/2030	0.036	8	81.517916
05/07/2030	0.036	5.999648	81.517916
06/07/2030	0.036	5.999648	81.517916
07/07/2030	0.036	5.999648	77.5439176
08/07/2030	0.036	5.999648	81.517916
09/07/2030	0.036	8	81.517916
10/07/2030	0.036	8	81.517916
11/07/2030	0.036	5.999648	81.517916

meanTimeCmar.csv: average over all timesteps of the marginal costs. 1 column per node, 1 row per scenario.

Table 21: meanTimeCmar.csv

	RoGonDor	DolAmroth	Harad
Base	256.72732	3439.75537	1521.54793
PVminus10	257.017259	3583.34174	1793.57264
Demandplus10	442.220462	4878.66198	3086.87072
PVminus10AndDemandplus10	451.32022	4977.57685	3259.88472

sortedCmar.csv: average of the marginal costs histogram. 1 column par node, 1 row per sorted time step index.

Table 22: SortedCmar.csv

	RoGonDor	DolAmroth	Harad
0	10000	10000	10000
1	10000	10000	10000
2	10000	10000	10000
3	10000	10000	10000
4	9512.5	10000	10000
5	9154.88089	10000	10000
6	9154.88089	10000	10000
7	8749.78	10000	10000
8	8506.03	10000	10000

MarginalCostActivePowerDemand-NODE.csv: marginal costs of the node (in this case, the nodes on which the *ActivePowerDemand* constraints apply). 1 column par scenario, 1 row per time step.

HistCmar-NODE.csv : histogram of the marginal costs of the zone. 1 column par scenario, 1 row per sorted time step index.

6.5.2.3.1 Flows

MeanImportExport-NODE.csv: average import and exports to/from the node. 1 column for Import, 1 column for export, 1 row per time step.

Table 23: meanImportExport-NODE.csv

	Export	Import
02/07/2030	0	22488
03/07/2030	0	22488
04/07/2030	0	22488
05/07/2030	0	22488
06/07/2030	0	22488
07/07/2030	0	22488

ImportExport-NODE-$i.csv: import and exports to/from the node for scenario $i. 1 column for import, 1 column for export, 1 row per time step.

MeanImportExport.csv: average flows. 1 column for import, 1 column for line, 1 row per time step.

Table 24: MeanImportExport.csv

	DolAmroth>Harad	RoGonDor>DolAmroth	RoGonDor>Harad
02/07/2030	1704	52800	20784
03/07/2030	1704	52800	20784
04/07/2030	1704	52800	20784
05/07/2030	1704	52800	20784
06/07/2030	1704	52800	20784
07/07/2030	1704	52800	20784

6.6 Appendix

7 SMS++ Configuration files

SMS++ configuration files may need to be edited in case optimization doesn't behave well.

Those files all share the same format:

- First part with global information about the solvers used and compute configurations (not to be edited)
- Integer parameters : a first row relling the number of integer parameters followed by the list of names and values of those parameters
- double parameters : a first row relling the number of integer parameters followed by the list of names and values of those parameters
- string parameters : a first row relling the number of integer parameters followed by the list of names and values of those parameters

In the following, we highlight in green the parameters which may be edited

7.1 Config files for SSV : sddp_solver.txt

```
BlockSolverConfig     # exact type of the Configuration object
1  # the BlockSolverConfig is a "differential" one
1  # number of (the names of) Solver in this BlockSolverConfig
# now all the names of the Solver
#SDDPSolver    # name of 1st Solver
ParallelSDDPSolver    # name of 1st Solver
1  # number of ComputeConfig in this BlockSolverConfig
# now all the ComputeConfig
# 1st ComputeConfig
# ComputeConfig of the SDDPSolver
ComputeConfig # exact type of the ComputeConfig object
1  # f_diff == 0 ==> all non-provided parameters are set to the default value
   # f_diff == 1 ==> all non-provided parameters are not changed
6 # number of integer parameters
# now all the integer parameters
intLogVerb                1  # log verbosity
intNStepConv             10  # Frequency at which the convergence is checked
intPrintTime              1  # Indicates whether computational time should be displayed
intNbSimulForward        10  # Number of simulations considered in the forward pass
intOutputFrequency        1  #
IntNbSimulCheckForConv   20  # THis parameter is automatically made equal to the number of
scenarios by the Launch* scripts
1 # number of double parameters
# now all the double parameters
dblAccuracy 0.01 # Relative accuracy for declaring a solution optimal
2 # number of string parameters
# now all the string parameters
strInnerBSC   uc_solverconfig.txt   # BlockSolverConfig for the UCBlock
strOutputFile cuts.txt              # name of the output file
```

7.2 Config files for simulation in SIM or CEM: sddp_greedy_investment.txt (CEM) / sddp_greedy.txt (SIM)

```
BlockSolverConfig     # exact type of the Configuration object
1  # the BlockSolverConfig is a "differential" one
1  # number of (the names of) Solver in this BlockSolverConfig
# now all the names of the Solver - - - - - - - - - - - - - - - - - - - - -
SDDPGreedySolver    # name of 1st Solver
```

```
1  # number of ComputeConfig in this BlockSolverConfig
# now all the ComputeConfig
# 1st ComputeConfig- - - - - - - - - - - - - - - - - - - - - - - - - - - -
# ComputeConfig of the SDDPGreedySolver; it basically is the
# ComputeConfig of the inner Solver, which is a BundleSolver
ComputeConfig # exact type of the ComputeConfig object
1  # f_diff == 0 ==> all non-provided parameters are set to the default value
   # f_diff == 1 ==> all non-provided parameters are not changed
2 # number of integer parameters
# now all the integer parameters
intLogVerb 1      # log verbosity
intUnregisterSolver 1 # unregister the Solver of the inner Block after solving it
0 # number of double parameters
# now all the double parameters
1 # number of string parameters
# now all the string parameters
strInnerBSC      uc_solverconfig.txt   # BlockSolverConfig for the UCBlock
```

7.3 Config file for CEM: BSPar-Investment.txt

```
BlockSolverConfig     # exact type of the Configuration object
1  # the BlockSolverConfig is a "differential" one
1  # number of (the names of) Solver in this BlockSolverConfig
# now all the names of the Solver - - - - - - - - - - - - - - - - - - - -
BundleSolver    # name of 1st Solver
1  # number of ComputeConfig in this BlockSolverConfig
# now all the ComputeConfig
# 1st ComputeConfig- - - - - - - - - - - - - - - - - - - - - - - - - - - -
# ComputeConfig of the SDDPGreedySolver; it basically is the
# ComputeConfig of the inner Solver, which is a BundleSolver
ComputeConfig # exact type of the ComputeConfig object
1  # f_diff == 0 ==> all non-provided parameters are set to the default value
   # f_diff == 1 ==> all non-provided parameters are not changed
23 # number of integer parameters
# now all the integer parameters
intDoEasy  0      # do easy components for (some) LagBFunction
intMaxIter 10000  # max number of iterations for each call
intMaxThread 6    # MaxThread, max number of new tasks to spawn
intLogVerb 5      # log verbosity of main Bundle algorithm
intWZNorm   2     # which norm to use in the norm-based stopping condition
intBPar1   20     # discard items when they have been useless for <this> iterations
intBPar2 1000     # max bundle size per component
intBPar3    1     # max n. of items to fetch from Fi() at each iteration
intBPar4    1     # min n. of items to fetch from Fi() at each iteration
intBPar6    0     # second parameter for dynamic max n. of items per iteration
intBPar7   11     # how to deal with the global pools
intMnSSC    0     # min number of consecutive SS with the same t for a t increase
intMnNSC    1     # min number of consecutive NS with the same t for a t decrease
inttSPar1  12     # long-term t-strategy (0 = none, 4 = soft, 8 = hard, 12 = balancing)
intMaxNrEvls 2    # maximum number of function evaluations at each iteration
intMPName  15     # MP solver: 0 = QPP, 7 = Cplex with quadratic stabilization
                  # + 8 = check for duplicate linearizations
# QPPenalty's parameters : - - - - - -
intMPlvl   4      # log verbosity of Master Problem
intQPmp1   0      # MxAdd, how many variables can be added to the base at each
                  # iteration in BMinQuad (0 = at will)
intQPmp2   0      # MxRmv, how many variables can be removed from the base at each
                 # iteration in BMinQuad (0 = at will)
# OSiMPSolver's parameters : - - - - - -
intOSImp1  4      # CPLEX algorithm
intOSImp2  1      # pre-processing (reduction)
intOSImp3  1      # threads
intRstAlg   2     # parameter to handle the reset of the algorithm
16 # number of double parameters
# now all the double parameters
```

```
dblRelAcc  1e-4   # relative accuracy required to solution
dblNZEps   1e-4   # stopping parameter: threshold to declare 0 the || residual ||
dbltStar   -100   # stopping parameter: multiplied to || residual || to estimate gap
dblBPar5   4      # first parameter for dynamic max n. of items per iteration
dblm1      -0.30  # a NS is possible if (~) Fi( Lambda1 ) >= Fi( Lambda ) + | m1 | v*
dblm2      0.90   # a SS is possible if Fi( Lambda1 ) <= Fi( Lambda ) + ( 1 - m2 ) v*
dblm3      0.99   # a NR is computed if \sigma^* < - t * m3 * || z* ||
dblmxIncr  10     # max increase of t
dblmnIncr  1.5    # min increase of t (each time it is increased)
dblmxDecr  0.1    # max decrease of t
dblmnDecr  0.66   # min decrease of t (each time it is decreased)
dbltMaior  1e+6   # maximum value for t
dbltMinor  1e-10  # minimum value for t
dbltInit   1e+0   # initial value for t
dbltSPar2  1e-2   # parameter for the long-term t-strategy
dblCtOff   0.01   # cut-off factor for pricing in MinQuad
```

7.4 Config file for Unit Commitment

2 types of files can be used : for solving via decomposition or via LP

7.4.1 Config file decomposition

Main file : BSPar-greedy-LD.txt :

```
BlockSolverConfig     # exact type of the Configuration object
1  # the BlockSolverConfig is a "differential" one
1  # number of (the names of) Solver in this BlockSolverConfig
# now all the names of the Solver - - - - - - - - - - - - - - - - - - - - - -
LagrangianDualSolver    # name of 2nd Solver
1  # number of ComputeConfig in this BlockSolverConfig
# now all the ComputeConfig
# 1st ComputeConfig- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# ComputeConfig of the LagrangianDualSolver; it mostly is the
# ComputeConfig of the inner Solver, which is a [Parallel]BundleSolver
ComputeConfig # exact type of the ComputeConfig object
1  # f_diff == 0 ==> all non-provided parameters are set to the default value
   # f_diff == 1 ==> all non-provided parameters are not changed
23 # number of integer parameters
# now all the integer parameters
intDoEasy  0      # do easy components for (some) LagBFunction
intMaxIter 10000  # max number of iterations for each call
intMaxThread 36   # MaxThread, max number of new tasks to spawn
intLogVerb 2      # log verbosity of main Bundle algorithm
intWZNorm   2     # which norm to use in the norm-based stopping condition
intBPar1   20     # discard items when they have been useless for <this> iterations
intBPar2 1000     # max bundle size per component
intBPar3    1     # max n. of items to fetch from Fi() at each iteration
intBPar4    1     # min n. of items to fetch from Fi() at each iteration
intBPar6    0     # second parameter for dynamic max n. of items per iteration
intBPar7   11     # how to deal with the global pools
intMnSSC    0     # min number of consecutive SS with the same t for a t increase
intMnNSC    1     # min number of consecutive NS with the same t for a t decrease
inttSPar1  12     # long-term t-strategy (0 = none, 4 = soft, 8 = hard, 12 = balancing)
intMaxNrEvls 2    # maximum number of function evaluations at each iteration
intMPName  15     # MP solver: 0 = QPP, 7 = Cplex with quadratic stabilization
                  # + 8 = check for duplicate linearizations
# QPPenalty's parameters : - - - - - -
intMPlvl   2      # log verbosity of Master Problem
intQPmp1   0    # MxAdd, how many variables can be added to the base at each
                 # iteration in BMinQuad (0 = at will)
intQPmp2   0      # MxRmv, how many variables can be removed from the base at each
                 # iteration in BMinQuad (0 = at will)
# OSiMPSolver's parameters : - - - - - -
intOSImp1  4      # CPLEX algorithm
```

```
intOSImp2  1      # pre-processing (reduction)
intOSImp3  1      # threads
intRstAlg   2     # parameter to handle the reset of the algorithm
16 # number of double parameters
# now all the double parameters
dblRelAcc  1e-4   # relative accuracy required to solution
dblNZEps   1e+4   # stopping parameter: threshold to declare 0 the || residual ||
dbltStar   -100   # stopping parameter: multiplied to || residual || to estimate gap
dblBPar5   4      # first parameter for dynamic max n. of items per iteration
dblm1      -0.30  # a NS is possible if (~) Fi( Lambda1 ) >= Fi( Lambda ) + | m1 | v*
dblm2      0.90   # a SS is possible if Fi( Lambda1 ) <= Fi( Lambda ) + ( 1 - m2 ) v*
dblm3      0.99   # a NR is computed if \sigma^* < - t * m3 * || z* ||
dblmxIncr  10     # max increase of t
dblmnIncr  1.5    # min increase of t (each time it is increased)
dblmxDecr  0.1    # max decrease of t
dblmnDecr  0.66   # min decrease of t (each time it is decreased)
dbltMaior  1e+6   # maximum value for t
dbltMinor  1e-10   # minimum value for t
dbltInit   1e+0   # initial value for t
dbltSPar2  1e-2   # parameter for the long-term t-strategy
dblCtOff   0.01   # cut-off factor for pricing in MinQuad
1 # number of string parameters
# now all the string parameters
str_LDSlv_ISName     ParallelBundleSolver  # the inner Solver used by LagrangianDualSolver
#str_LagBF_BSCfg            LPBSCfg.txt   # BlockSolverConfig for the LagBFunctions
# not needed if intDoEasy >= 1 && USE_BundleSolver > 0, this is done "by hand"
# note that we could eof() the file here since the rest is all empty
```

7.4.2 Config file without decomposition

```
BlockSolverConfig # The name of the configuration
1  # The BlockSolverConfig is "differential"
1  # The number of Solvers
# Now all the names of the Solvers
CPXMILPSolver
1  # The number of ComputeConfigs
# Now all the ComputeConfigs
# 1st -----------------------------------
ComputeConfig  # Type of the object
1  # differential
2  # Number of integer parameters
intLogVerb 0
intRelaxIntVars 1 # 1 =relax integer constraints
0  # Number of double parameters
# dblMaxTime 3600
3  # Number of string parameters
strOutputFile uc.lp
CPXPARAM_CPUmask auto
CPXPARAM_WorkDir .
```

8 Direct use of SMS++ executables

8.1 Running the Unit Commitment (UC)

The unit commitment can be run with the following command :

```
ucblock_solver [options] Block.nc4
```

with the following set of options :

```
  -B, --blockcfg <file>    Block configuration.
```

```
  -S, --solvercfg <file>   Solver configuration.
  -n, --nc4problem <file>  Write nc4 problem on file.
  -v, --verbose            Make the solver verbose.
  -o, --output <type>      Solution output type (0 none, 1 screen, 2 files,
3 both).
  -h, --help               Print help.
```

`Block.nc4` is the netcdf input file ; it can be generated by the formatting tool, mode UC.

ucblock_solver requires specific configuration files for SMS++: BSPar-greedy-LD.txt, TUBSCfg.txt, OUBSCfg.txt, OUBSCfg.txt, HSUBSCfg.txt if using decomposition or uc_solverconfig.txt if not (see 7.4.)

Detailed documentation as well as examples of configuration files can be found on the SMS++ repository : SMS++ / Tools · GitLab

8.2 Running the SSV or the SIM

`sddp_solver` [options] `SDDPBlock.nc4`

with the following set of options :

```
  -B, --blockcfg <file>           Block configuration.
  -c, --configdir <path>          The prefix for all config filenames.
  -e, --eliminate-redundant-cuts  Eliminate given redundant cuts.
  -h, --help                      Print this help.
  -i, --scenario <index>          The index of the scenario.
  -l, --load-cuts <file>          Load cuts from a file.
  -m, --num-simulations <number>  Number of simulations to be performed.
  -n, --num-blocks <number>       Number of sub-Blocks per stage.
  -p, --prefix <path>             The prefix for all Block filenames.
  -s, --simulation                Simulation mode.
  -S, --solvercfg <file>          Solver configuration.
  -t, --stage <stage>             Stage from which initial state is taken.
```

The option -s corresponds to the SIM, it requires to choose a scenario with option -i ; without this option the SSV will be run. For running the SIM, the option -l is also needed (for loading BellmanValues)

`SDDPBlock.nc4` is the netcdf input file. `Block$i.nc4` (at all $i stages) are also needed (they must be in the directory given with option -p); they can be generated by the formatting tool, mode SDDPandUC.

sddp_solver requires a specific configuration file for SMS++: sddp_solver.txt for solving the SSV, and sddp_greedy.txt for solving the SIM (see 7.17.4, 7.2), as well as the configuration file of ucblock_solver (see 7.4.)

Detailed documentation as well as examples of configuration files can be found on the SMS++ repository : SMS++ / Tools · GitLab

8.3 Running the CEM

`investment_solver` [options] `InvestmentBlock.nc4`
with the following set of options :

```
  -a, --save-state <prefix>       Save states of the InvestmentBlock
solver.
```

```
  -B, --blockcfg <file>           Block configuration.
  -b, --load-state <file>         Load a state for the InvestmentBlock
solver.
  -c, --configdir <path>          The prefix for all config filenames.
  -e, --eliminate-redundant-cuts  Eliminate given redundant cuts.
  -h, --help                      Print this help.
  -l, --load-cuts <file>          Load cuts from a file.
  -n, --num-blocks <number>       Number of sub-Blocks per stage.
  -o, --output-solution           Output the solutions.
  -p, --prefix <path>             The prefix for all Block filenames.
  -S, --solvercfg <file>          Solver configuration.
  -s, --simulate                  Simulate the given investment.
  -x, --initial-investment <file> Initial investment.
```

Detailed documentation as well as examples of configuration files can be found on the SMS++ repository : SMS++ / Tools · GitLab

`InvestmentBlock.nc4` is the netcdf input file. `SDDPBlock.nc4` and `Block$i.nc4` (at all $i stages) are also needed (they must be in the directory given with option -p); they can be generated by the formatting tool, mode INVESTandSDDPandUC.

investment_solver requires specific configuration file for SMS++: BSPar-Investment.txt, as well as a configuration file for the SIM: sddp_greedy_investment.txt (see 7.2), and the configuration file of ucblock_solver (see 7.4.)