plan4res use

1 Quick Run

- Create a repository (eg MY_STUDY) for your study within p4r-env/data/local/
- Upload GENeSYS-MOD results (IAMC format) to MY_STUDY/
- Upload time series (for stochastic data) to MY_STUDY/TimeSeries/
- Edit file p4r-env/scripts/python/plan4res-scripts/settingsCreatePlan4resInput.yml (change the following variables)
    - **scenarios**
    - **years**
    - **path**
    - **timeseriespath**
    - **listregionsGET** (this is the list of regions for which you want to retrieve GENeSYS-MOD results)
    - **aggregateregions** (with your own choices of aggregations)
    - **partition** (with the list of -aggregated or not aggregated- regions you want in your dataset)
    - **technos** (with the technos existing in GENeSYS-MOD resuls)
    - **StochasticScenarios** (with the list of scenarios you have in your timeseries)
    - **CouplingConstraints** (with the list of coupling constraints you wish to account for)
    - **ParametersCreate:CapacityExpansion** (with the bounds for the CEM)
    - **ParametersCreate:invest** (yes if you wish to run the CEM)
    - **ParametersCreate:InitialFillingrate** (level of the seasonal reservoirs at beginning of case study, in %)
    - **Listdatagroups (**if you need to use more than 1 data file from GENeSYS-MOD**,** update each datagroup with path and list of variables**)**
- Run plan4res without investments

    - scripts/LaunchSSVandSIM
- Or run SSV with investments
    - scripts/LaunchSSVandCEM
- You may now look at the results in p4r-env/data/local/MY_STUDY/results (for the case without investment) or results_invest (for the case with investments)

2 Plan4res launching scripts

The following scripts for running plan4res are available. They must be launched from p4r-env/scripts.

2.1 LaunchSSVandSIM

This scripts launches the following operations:

1. **Create a dataset with the plan4res data format,** out of data in the IAMC format, eg. Outputs of GENeSYS-MOD.

   Inputs: files in the IAMC format, stored in sub-directories of p4r-env/data (listed in the configuration file), and name NAMEDIR of the sub-directory to be created in p4r-env/data

   Outputs: files ZP_Zonepartition.csv, ZV_ZoneValues.csv, TU_ThermalUnits.csv, SS_SeasonalStorage.csv, STS_ShortTermStorage.csv, RES_RenewableUnits.csv

   Configuration file: settingsCreateInputPlan4res_simul.yml (stored in p4r-env/scripts/python/plan4res-scripts)

   This operation runs CreateInputPlan4res.py. Note that the csv files may also be created manually.
   The script and its configuration file are available in *https://github.com/openENTRANCE/plan4res-scripts.git*

   Command launched : python CreateInputPlan4res.py settingsCreateInputPlan4res_simul.yml NAMEDIR

2. **Convert the csv plan4res input files in netcdf4 files (inputs for SMS++)**

   Inputs: files ZP_Zonepartition.csv, ZV_ZoneValues.csv, TU_ThermalUnits.csv, SS_SeasonalStorage.csv, STS_ShortTermStorage.csv, RES_RenewableUnits.csv (listed in the configuration file), and name NAMEDIR of the sub-directory to be created in p4r-env/data

   Outputs: SDDPBlock.nc4 and Block$i.nc4 ($i are indexes of SSV timesteps), in p4r-env/data/NAMEDIR/nc4_optim/

   Configuration files: settings_format_optim.yml, settingsCreateInputPlan4res_simul.yml (stored in p4r-env/scripts/python/plan4res-scripts)

   This operation runs format.py.The script and its configuration files are available in *https://github.com/openENTRANCE/plan4res-scripts.git*

   Command launched : python format.py settings_format_optim.yml settingsCreateInputPlan4res_simul.yml NAMEDIR

3. **Run the optimisation (SSV), sddp_solver**:

Inputs: SDDPBlock.nc4 and Block$i.nc4 ($i are indexes of SSV timesteps), in p4r-env/data/NAMEDIR/nc4_optim/

Outputs: BellmanValuesOUT.csv and cuts.txt (BellmanValuesOUT.csv is created only if the sddp has converged; cuts.txt exists as soon as it has completed at least one iteration), in p4r-env/data/NAMEDIR/results/

Configuration files: sddp_solver.txt (stored in p4r-env/config/)

This operation runs sddp_solver, which is part of SMS++. The configuration file is available in https://github.com/openENTRANCE/plan4res.git

Command launched : sddp_solver -S config/sddp_solver.txt -c config/ -p data/NAMEDIR/nc4_optim/ data/NAMEDIR/nc4_optim/SDDPBlock.nc4

4. **Create netcdf4 files (inputs for SMS++) for the simulation (SIM)**

Inputs: files ZP_Zonepartition.csv, ZV_ZoneValues.csv, TU_ThermalUnits.csv, SS_SeasonalStorage.csv, STS_ShortTermStorage.csv, RES_RenewableUnits.csv (listed in the configuration file), and name NAMEDIR of the sub-directory to be created in p4r-env/data

Outputs: SDDPBlock.nc4 and Block$i.nc4 ($i are indexes of SSV timesteps), in p4r-env/data/NAMEDIR/nc4_simul/

Configuration files: settings_format_simul.yml, settingsCreateInputPlan4res_simul.yml (stored in p4r-env/scripts/python/plan4res-scripts)

This operation runs format.py.The script and its configuration files are available in *https://github.com/openENTRANCE/plan4res-scripts.git*

Command launched : python format.py settings_format_simul.yml settingsCreateInputPlan4res_simul.yml NAMEDIR

5. **Run the simulation (SIM), sddp_solver (with different config)**

Inputs: SDDPBlock.nc4 and Block$i.nc4 ($i are indexes of SSV timesteps), in p4r-env/data/NAMEDIR/nc4_simul/

Outputs: files Demand$i.csv, Volume$i.csv, ActivePower$i.csv, Primary$i.csv, Secondary$i.csv,Demand$i.csv, MarginalCostActivePowerDemand$i.csv, MarginalCostPrimary$i.csv MarginalCostSecondary$i.csv MarginalCostInertia$i.csv MarginalCostFlows$i.csv, MaxPower$i.csv, Flows$i.csv in p4r-env/data/NAMEDIR/results/

Configuration files: sddp_greedy.txt (stored in p4r-env/config/)

This operation runs sddp_solver, which is part of SMS++. The configuration file is available in https://github.com/openENTRANCE/plan4res.git

Command launched : for each scenario $scen, sddp_solver -s -i $scen -S config/sddp_greedy.txt -c config/ -p data/NAMEDIR/nc4_optim/ data/NAMEDIR/nc4_optim/SDDPBlock.nc4
Each scenario run creates files outputOUT.csv which are converted to output$scen.csv and moved to p4r-env/data/NAMEDIR/results/ in dedicated sub-repositories per output.

6. **Posttreat results** : this converts the outputs of simulation to more readable outputs (eg per region), creates some additional and some synthetized output files (in p4r-env/data/NAMEDIR/results/OUT), creates graphs files (in p4r-env/data/NAMEDIR/results/IMG), converts the outputs in IAMC format data files (in p4r-env/data/NAMEDIR/results/IAMC) and creates a latex report files (in p4r-env/data/NAMEDIR/results/LATEX)

Inputs: files Volume$i.csv, ActivePower$i.csv, Primary$i.csv, Secondary$i.csv,Demand$i.csv, MarginalCostActivePowerDemand$i.csv, MarginalCostPrimary$i.csv MarginalCostSecondary$i.csv MarginalCostInertia$i.csv MarginalCostFlows$i.csv, MaxPower$i.csv, Demand$i.csv, Flows$i.csv in p4r-env/data/NAMEDIR/results/

Outputs:

Detailed outputs: files Demand-$region.csv, Volume-Reservoir-$region.csv, Generation-$region-$i.csv, MarginalCostActivePowerDemand-$region.csv, HistCmar-$region.csv (histogram of marginal costs), MarginalCost-$reg2$reg.csv, ImportExport-$region-$i.csv, ImportExport$i.csv in p4r-env/data/NAMEDIR/results/, in the same sub directories as the intpu files

Synthetic outputs: files Volume-Reservoir.csv, Slack-$region.csv (non served demand), nbHoursSlack.csv, InstalledCapacity.csv, AggrInstalledCapacity.csv, Generation.csv, AggrGeneration.csv, Generation-$region.csv, MonotoneCmar.csv (stochastic histogram of marginal costs), meanTimeCmar.csv (average on time of marginal costs), meanScenCmar.csv(average on scenarios of marginal costs), meanVariableCost.csv, MeanImportExport.csv, meanImportExport-$region.csv

Configuration files: settingsPostTreatPlan4res_simul.yml, settingsCreateInputPlan4res_simul.yml (stored in p4r-env/scripts/python/plan4res-scripts)

This operation runs PostTreatPlan4res.py.The script and its configuration files are available in *https://github.com/openENTRANCE/plan4res-scripts.git*

Command launched : python PostTreatPlan4res.py settingsPostTreatPlan4res_simul.yml settings_format_simul.yml settingsCreateInputPlan4res_simul.yml NAMEDIR

2.2 LaunchSSVandCEM

This scripts is very close to the LaunchSSVandCEM except that after computing the bellmanvalues and before running the simulation it optimizes the investments. In the first steps the only difference is in the creation of the dataset, which includes technologies with 0 capacity as they could be invested. It launches the following operations:

1. **Create a dataset with the plan4res data format,** out of data in the IAMC format, eg. Outputs of GENeSYS-MOD.

   Inputs: files in the IAMC format, stored in sub-directories of p4r-env/data (listed in the configuration file), and name NAMEDIR of the sub-directory to be created in p4r-env/data

   Outputs: files ZP_Zonepartition.csv, ZV_ZoneValues.csv, TU_ThermalUnits.csv, SS_SeasonalStorage.csv, STS_ShortTermStorage.csv, RES_RenewableUnits.csv

Configuration file: settingsCreateInputPlan4res_invest.yml (stored in p4r-env/scripts/python/plan4res-scripts)

This operation runs CreateInputPlan4res.py. Note that the csv files may also be created manually.
The script and its configuration file are available in
https://github.com/openENTRANCE/plan4res-scripts.git

Command launched : python CreateInputPlan4res.py settingsCreateInputPlan4res_invest.yml NAMEDIR

2. **Convert the csv plan4res input files in netcdf4 files (inputs for SMS++)**

Inputs: files ZP_Zonepartition.csv, ZV_ZoneValues.csv, TU_ThermalUnits.csv, SS_SeasonalStorage.csv, STS_ShortTermStorage.csv, RES_RenewableUnits.csv (listed in the configuration file), and name NAMEDIR of the sub-directory to be created in p4r-env/data

Outputs: SDDPBlock.nc4 and Block$i.nc4 ($i are indexes of SSV timesteps), in p4r-env/data/NAMEDIR/nc4_optim/

Configuration files: settings_format_optim.yml, settingsCreateInputPlan4res_invest.yml (stored in p4r-env/scripts/python/plan4res-scripts)

This operation runs format.py.The script and its configuration files are available in
https://github.com/openENTRANCE/plan4res-scripts.git

Command launched : python format.py settings_format_optim.yml settingsCreateInputPlan4res_invest.yml NAMEDIR

3. **Run the optimisation (SSV), sddp_solver**:

Inputs: SDDPBlock.nc4 and Block$i.nc4 ($i are indexes of SSV timesteps), in p4r-env/data/NAMEDIR/nc4_optim/

Outputs: BellmanValuesOUT.csv and cuts.txt (BellmanValuesOUT.csv is created only if the sddp has converged; cuts.txt exists as soon as it has completed at least one iteration), in p4r-env/data/NAMEDIR/results/

Configuration files: sddp_solver.txt (stored in p4r-env/config/)

This operation runs sddp_solver, which is part of SMS++. The configuration file is available in
https://github.com/openENTRANCE/plan4res.git

Command launched : sddp_solver -S config/sddp_solver.txt -c config/ -p data/NAMEDIR/nc4_optim/ data/NAMEDIR/nc4_optim/SDDPBlock.nc4

4. **Create netcdf4 files (inputs for SMS++) for the investment (CEM)**

Inputs: files ZP_Zonepartition.csv, ZV_ZoneValues.csv, TU_ThermalUnits.csv, SS_SeasonalStorage.csv, STS_ShortTermStorage.csv, RES_RenewableUnits.csv (listed in the configuration file), and name NAMEDIR of the sub-directory to be created in p4r-env/data

Outputs: SDDPBlock.nc4 and Block$i.nc4 ($i are indexes of SSV timesteps), in p4r-env/data/NAMEDIR/nc4_invest/

Configuration files: settings_format_invest.yml, settingsCreateInputPlan4res_invest.yml (stored in p4r-env/scripts/python/plan4res-scripts)

This operation runs format.py.The script and its configuration files are available in *https://github.com/openENTRANCE/plan4res-scripts.git*

Command launched : python format.py settings_format_invest.yml settingsCreateInputPlan4res_invest.yml NAMEDIR

5. **Run the investment model (CEM), investment_solver**

Inputs: InvestmentBlock.nc4, SDDPBlock.nc4 and Block$i.nc4 ($i are indexes of SSV timesteps), in p4r-env/data/NAMEDIR/nc4_invest/

Outputs: files Demand$i.csv, Volume$i.csv, ActivePower$i.csv, Primary$i.csv, Secondary$i.csv,Demand$i.csv, MarginalCostActivePowerDemand$i.csv, MarginalCostPrimary$i.csv MarginalCostSecondary$i.csv MarginalCostInertia$i.csv MarginalCostFlows$i.csv, MaxPower$i.csv, Flows$i.csv in p4r-env/data/NAMEDIR/results_invest/

The results of the investment is given directly in the command line, in the form of a serie of multiplicative factors which shall be applied to the invested technologies.

Configuration files: BSPar-Investment.txt (stored in p4r-env/config/)

This operation runs investment_solver, which is part of SMS++. The configuration file is available in https://github.com/openENTRANCE/plan4res.git

Command launched : investment_solver -l data/NAMEDIR/results_invest/BellmanValuesOUT.csv -o -e -S config/BSPar-Investment.txt -c config/ -p data/NAMEDIR/nc4_invest/ data/NAMEDIR/nc4_invest/InvestmentBlock.nc4
Investment_solver also launches the simulation on all scenarios
Each scenario run creates files outputOUT.csv which are converted to output$scen.csv and moved to p4r-env/data/NAMEDIR/results/ in dedicated sub-repositories per output.

6. **Posttreat results** : this converts the outputs of simulation to more readable outputs (eg per region), creates some additional and some synthetized output files (in p4r-env/data/NAMEDIR/results_invest/OUT), creates graphs files (in p4r-env/data/NAMEDIR/ results_invest /IMG), converts the outputs in IAMC format data files (in p4r-env/data/NAMEDIR/ results_invest /IAMC) and creates a latex report files (in p4r-env/data/NAMEDIR/ results_invest /LATEX)

Inputs: files Volume$i.csv, ActivePower$i.csv, Primary$i.csv, Secondary$i.csv,Demand$i.csv, MarginalCostActivePowerDemand$i.csv, MarginalCostPrimary$i.csv MarginalCostSecondary$i.csv MarginalCostInertia$i.csv MarginalCostFlows$i.csv, MaxPower$i.csv, Demand$i.csv, Flows$i.csv in p4r-env/data/NAMEDIR/results/

Outputs:

Detailed outputs: files Demand-$region.csv, Volume-Reservoir-$region.csv, Generation-$region-$i.csv, MarginalCostActivePowerDemand-$region.csv, HistCmar-$region.csv (histogram of marginal costs), MarginalCost-$reg2$reg.csv, ImportExport-$region-$i.csv, ImportExport$i.csv in p4r-env/data/NAMEDIR/ results_invest /, in the same sub directories as the intpu files

Synthetic outputs: files Volume-Reservoir.csv, Slack-$region.csv (non served demand), nbHoursSlack.csv, InstalledCapacity.csv, AggrInstalledCapacity.csv, Generation.csv, AggrGeneration.csv, Generation-$region.csv, MonotoneCmar.csv (stochastic histogram of

marginal costs), meanTimeCmar.csv (average on time of marginal costs), meanScenCmar.csv(average on scenarios of marginal costs), meanVariableCost.csv, MeanImportExport.csv, meanImportExport-$region.csv

Configuration files: settingsPostTreatPlan4res_invest.yml, settings_format_nvest.yml settingsCreateInputPlan4res_invest.yml (stored in p4r-env/scripts/python/plan4res-scripts)

This operation runs PostTreatPlan4res.py.The script and its configuration files are available in *https://github.com/openENTRANCE/plan4res-scripts.git*

Command launched : python PostTreatPlan4res.py settingsPostTreatPlan4res_investl.yml settings_format_invest.yml settingsCreateInputPlan4res_invest.yml NAMEDIR

2.3 Other scripts

Scripts running individual operations are also available:

- LaunchCREATE: this script runs operation 1, ie creates plann4res csv input data only
- LaunchSSV: this script runs operations 2 and 3 ie creates netcdf input files of SMS++ and runs sddp_solver to compute Bellman values. Note that if sddp_solver did not converge it can be hot-started by adding -l cuts.txt as an option (the already computed cuts from cuts.txt will be used to restart)
- LaunchCEM: this script creates netcdf input files for the CEM and runs investment_solver
- LaunchSIM: this script creates netcdf input files for the SIM and runs the simulation with sddp_solver

3 Configuration files

3.1 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

3.1.1 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              100 # 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 #
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
```

3.1.2 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
```

3.1.3 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
```

3.1.4 Config file for Unit Commitment

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

3.1.4.1 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
```

3.1.4.2 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 .
```

3.2 Python scripts configuration files

Those scripts are available in https://github.com/openENTRANCE/plan4res-scripts.git

They are written using yaml.

3.2.1 settingsCreateInputPlan4res_XXX.yaml

XXX stands for simul or invest

This configuration file contains the following parameters (editable parameters are in green):

- scenarios: => name of the scenario to use from GENeSYS-MOD data;
- years: => year to use from GENeSYS-MOD data
- USEPLAN4RESROOT: don't change, used to tell python where the root path is
- path: path where to retrieve data, automatically filled when the python script is passed NAMEDIR
- timeseriespath: location of Timeseries
- configDir: location of this configuration file
- nomenclatureDir: 'location of openentrance definitions
- csvfiles: list of csv files to create in the plan4res excel file
- listregionsGET: list of regions to extract from GENeSYS-MOD output
- aggregateregions: defines aggregations of regions
- partition: defines the partitions of the regions to be applied to coupling constraints
- technos: list of technologies to include. Technologies are separated into subgroups:
    - thermal= modelled as ThermalUnit (TU),
    - reservoir= modelled as HydroUnit (SS),
    - hydrostorage/battery=modelled as ShortTermStorage (STS),
    - res/runofriver=modelled as IntermittentUnit (RES))
- StochasticScenarios: list of scenarios to include in runs (scenarios index are those of the timeseries)
- CouplingConstraints: (describes the coupling constraints – see 3.2.4); Coupling contraints can be : Demand, Primary, Secondary, Inertia, CO2 ; Partition defines the level at which the coupling constraint applies. 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
- PumpingEfficiency: used to replace data when non available in dataset
- ParametersCreate:

    - CapacityExpansion: list of technologies which can be invested with the investment bounds
    - conversions: => Conversions rules:
    - zerocapacity: capacity in MW under which it is considered to be 0
    - DynamicConstraints: if 'no' the dataset will not include dynamic constraints: MinPower is set to 0, StartUpCost, MinUp and Down duration are not used
    - invest: 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
    - MultFactor: list of technos where MaxPower is computed as energy/(mean load profile*8760)
    - Volume2CapacityRatio : used to compute the MaxPower or MaxVolume of storage units
    - reservoir: minpowerMWh : minimum capacity under which the reservoir is converted to short term storage
    - NbUnitsPerTechno: if >1, the data must include unit max power for the technology and the number of units is computed as Capacity/MaxPower ; if 1: 1 asset per technology
    - variablecost: defines a list of technologies for which the variable cost is computed as the product price*efficiency
    - InitialFillingrate: initial filling rates of reservoirs per countries
    - ExistsNuts: used only if some data are given at lower resolution than countries (NUTS in Europe)
    - DemandResponse parameters are only used when accounting for residential demand response
- the datagroup section lists the different 'sources' to get the data; for each source it gives
    - the name of the directory and file where to find data;
    - The name of model, scenario, year to filter
    - (optional) A list of additionnal regions to retrieve (compared to listregionsGET above, among nuts1, nuts2, nuts3, countries, ehighway
    - the list of variables to retrieve ; variables are separated into
        - global : global variables, not depending on technologies (eg demand, carbon price)
        - techno : variables for which the treatment is different depending on the kind of technology, among thermal, reservoir, hydrostorage, battery, res and demandresponseloadshifting

  for each group, variables are separated into 3 categories, depending on the method to use for aggregating regions: add (values for regions will be added in case of aggregation), global (the same value is used for all regions) and mean (in case of aggregation, the average value will be used)

- listdatagroups: gives the list of datasources (as the different variables required may be retrieved from different datasets)

3.2.2 settings_format_XXX.yaml

XXX stands for optim, simul or invest

This configuration file contains the following parameters (editable parameters are in green):

- outputDir: sub-directory of data/NAMEDIR where nc4 files are created (in practice nc4_optim, nc4_simul or nc4_invest
- 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.
- FormatVU defines the kind of bellman values used as inputs: None is the more usual used value, meaning that no bellman values are passed to sddp_solver, 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.
- 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 3.2.5)
    - 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 : beginning of the dataset to be created
    - EndDataset: end of the dataset to be created
    - SSVTimeStep: duration of the timestep in the SSV (eg 1 week)
    - TimeStep: duration of the timestep (eg 1 hour) (TimeStep is always lower or equal than SSVTimeStep=
- IncludeScenarisedData: True or False wether scenarised data should be included in the Blocks (not necessary for running plan4res, useful only for running single UC)
- ParametersFormat
    - DownReservoirVolumeMultFactor: mult factor for computing the size of virtual downstream reservoir from max volume of upstream reservoir
    - DownDeltaRampUpMultFactor: mult factor for computing the max ramp up of virtual downstream reservoir from ramp up of upstream reservoir
    - DownDeltaRampDownMultFactor: mult factor for computing the max ramp down of virtual downstream reservoir from ramp down of upstream reservoir
    - NumberHoursInYear: number of hours in a year (8760)
    - InertiaMultFactor: mult factor for computing inertia contributions
    - Scenarios: list of scenarios to include in the instance (indexes of scenarios, among those available in the timeseries)
    - ScenarisedData 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
    - ThermalMaxPowerTimeSpan: frequency of the data for scenarised thermal max power profiles
    - CoeffSpillage: it is allowed to spill CoeffSpillage*Maximum Flow
    - LowerBound: optional: lower bound for SDDP

3.2.3 settingsPostTreatPlan4res_XXX.yaml

XXX stands for simul or invest

This configuration file contains the following parameters (editable parameters are in green):

- BeginTreatData: optional – start of the analysis
- EndTreatData: optional – end of the analysis
- Resultsdir: sub-repository in data/NAMEDIR where all results are
- Map: defines wether maps will be created
- PostTreat: list of treatments to be done per category of results among Volume, Flows, Power, MarginalCost, MarginalCostFlows, Demand, InstalledCapacity and SpecifiedPeriods; The sub-repository of the results directory must be specified for each category (aprt from InstalledCapacity and SpecifiedPeriods). For each category the following operations can be done:
    - read: read results and creates additional results files
    - draw: create graphs
    - latex: create latex report
    - iamc: converts results to IAMC format

  SpecifiedPeriods allows the user to choose a list of scenarios and subperiods which will be analysed in details

- Dimensions of graphics: the number of columns and rows allows to produce graphs composed of sub graphs for each regions. It must ne chosen such as all regions can be included. The size of columns and rows must be chosen accordingly. TitleSize is the dimension of the title of the grap and LabelSize the simension of the labels
- Identifiers to use for creation of IAMC files: name of the scenario (should be the name of the scenario used in CreateInputPlan4res followed by '|' and an additional identifier if necessary. Model should be plan4res v1.0 (or following versions)
- Identifiers for the latex report: name of the latex file and title of the report
- Marginalcostlimits allows to specify limits for the graphs for the marginal costs
- Colors of the graphs: for each technology, a color identified (HEX code) is given
- Aggregated technologies allows to aggregate technologies ; A color identifier needs to be defined.
- Pumping/no pumping lists the storage technologies with/without pumping
- graphVolumes gives information for creation of the graphs for storage technologies: name of graphs, technos to include

3.2.4 Coupling Constraints

this lists the different coupling constraints that may be used are given in configuration files (see below). The following constraints may be included :

- ActivePowerDemand: this is related to the equilibrium constraint Active power = Active Power demand at each node of the network. The Active Power Demand is computed as the sum of different time-series that are listed in the corresponding line (this means that the power demand is computed as the sum of the electric heating demand, the electric cooling demands….)
- PrimaryDemand: this is related to the primary reserve, in each “zone” (can be a country….)
- SecondaryDemand: this is related to the secondary reserve.
- InertiaDemand: this is related to the inertia requirement.
- PollutantBudget: this constraint is not used in the current version of the tool. It is related to the maximum amount of polluting emissions, for a specific pollutant.

3.2.5 Time Parameters :

Time is discretized in:

- Time Sets (usually 1 week)
- Time Steps (usually 1 hour)

The time horizon is described in by the following variables in the ‘calendar’ block of the format configuration file. These parameters give both the periods where timeseries are available and the period when we want to run the model. When timeseries are available only on a given year, they are used anyway. If the dataset is longer than the timeseries period, the timeseries is extended. Also timeseries may be given with a different frequency. The scripts will adapt and recompute the timeseries with the required frequency.

- Timestep is the timestep duration, used for solving the unit commitment.
- SSVTimeStep gives the duration of the Time Sets (which is the seasonal storage timestep). In the current example, it is 168 hours (1 week). SSVTimeStep has to be at least bigger than UCTimeStep.

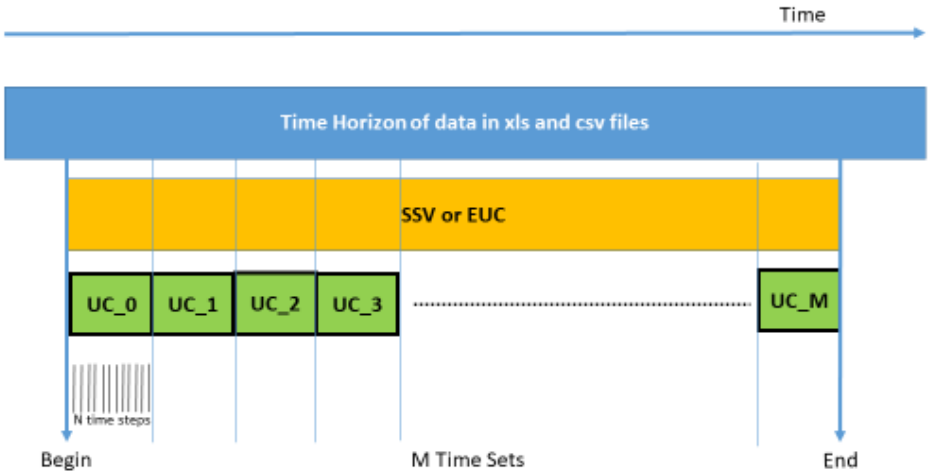

Figure 1: Time Management

3.2.6 Partitions

Description of 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.

Those partitions are described in files or sheets named ZP_ZonePartition, that follow the same rules:

The first column 'Level1' lists the lowest level partition. The partition 'level2' is a higher level partition, meaning that the regions in Level2 are larger than the regions in level1, and each region in level2 is composed of a list of regions in level1. Level3 partitions again are bigger than level2. At each level we may have different partitions obtained by regrouping regions differently (here we have 2 different ways of grouping the regions of level2: level3part1 and level3part2).

- Level1 = identifier of the first level zone (e.g., datazone) - string
- Level2= identifier of the second level zone (e.g., cluster) - string
- Level3Part1 = identifier of the third level zone (e.g., region), in a first partition – string
- Level3Part2 = identifier of the third level zone (e.g., region), in a second partition – string
- Etc.

A row with values L1, L2, L31, L32 … means that zone L1 belongs to second level zone L2, and this zone belongs to third level zones L31 and L32, …; Each L1 belongs to a unique L2; Each L2 belongs to a unique L3 for the first partition, and to a unique L3 – that can be different- for the second partition, etc…

The Figure 2 below illustrates this. In Blue: Level1; in Green: level2, in red and orange: level3 (partition1 and partition2), and in black level4. Here there are 7 zones in level1, that are aggregated in 4 level 2 clusters: BE, FR1, FR2 and ES. Each level1 zone belongs to a unique level2 cluster. At level3 there are 2 partitions: Level3part1 partitions the 4 clusters as North=BE+FR1, South=FR2+ES while Level3part2 partitions the same 4 clusters as BE=BE, FR=FR1+FR2 and ES=ES.

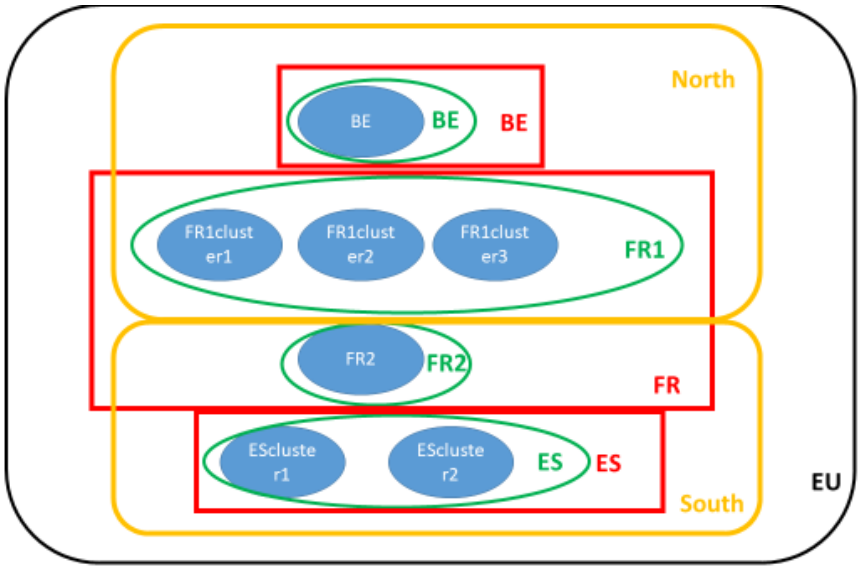

Figure 2: partitions

3.2.7 VariablesDict.yml

This file contains 2 sections, one for the outputs and one for the inputs:

- Output: contains the list of plan4res output variables with the corresponding IAMC format name (see https://github.com/openENTRANCE/openentrance/tree/main/definitions)
- Input: contains the names of the plan4res input variables and the corresponding IAMC names. It is divided in sections corresponding to each csv file of the plan4res input (see **Erreur ! Source du renvoi introuvable.**)

3.2.8 DictTimeSeries.yml

This file contains the names of the timeseries to use for the stochastic data. It contains the following sections:

- ZV for timeseries related to the demand (which may be created by parts, such as AirCOndition, ElecHeating…. Or at once). For each subsection it gives the nbame of the timeseries to use for each region.

4 Use of SMS++ executables

4.1 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 3.1.4.)

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

4.2 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 3.1.13.1.4, 3.1.2), as well as the configuration file of ucblock_solver (see 3.1.4.)

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

4.3 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 3.1.2), and the configuration file of ucblock_solver (see 3.1.4.)

5 plan4res python scripts

5.1 CreateInputPlan4res.py

This script creates the csv files in plan4res data format (see **Erreur ! Source du renvoi introuvable.**) out of results of GENeSYS-MOD in IAMC format.

CreateInputPlan4res.py requires 3 configuration files:

- settingsCreateInputPlan4res.yml 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
- TimeSeriesDict: contains the list of timeseries to use for the different stochastic variables (described in the main configusation file) and regions

The following operations are done for creating a plan4res dataset corresponding to one GENeSYS-MOD scenario and one GENeSYS-MOD 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.

5.2 Format.py

The script format.py reads the csv plan4res input files (see section **Erreur ! Source du renvoi introuvable.** for a description of the format), and creates a serie of NetCdF files (SMS++ format) :

- 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 (usually week) ;

format.py requires 4 configuration files:

- Settings_format_XXX.yaml which is the main configuration file for creating the netcdf files for the required run (SSV, SIM or CEM)
- (optionnal) settingsCreateInputPlan4res.yml which is configuration file used for creating the dataset

- VariablesDict.yml : contains the list of variables to retrieve and the correspondence between the plan4res and IAMC variable names
- TimeSeriesDict: contains the list of timeseries to use for the different stochastic variables (described in the main configusation file) and regions

Format.py can be run either by using the script LaunchXXX or by the command :

Python -W ignore format.py settings_format_xxx.yml settingsCreateInputPlan4res.yml

A full description of the SMS++ NetCdF input files is available in the SMS++ GitLab:
https://gitlab.com/smspp/smspp-project/-/blob/develop/doc/SMS++%20File%20Format%20Manual/ffm.pdf

5.3 PostTreatPlan4res.py

The script PostTreatPlan4res.py reads the csv plan4res output files, after run of CEM or SIM (see section **Erreur ! Source du renvoi introuvable.** for a description of the format), and creates :

- Post-treated output files
- IAMC format output files
- Graphs
- A latex report

PostTreatPlan4res.py requires 5 configuration files:

- SettingsPostTreatPlan4res.yaml which is the main configuration file
- (optionnal) Settings_format_XXX.yaml which is the configuration file used for creating thenetcdf files for the required run (SIM or CEM)
- (optionnal) settingsCreateInputPlan4res.yml which is configuration file used for creating the dataset
- VariablesDict.yml : contains the list of plan4res output variables and the correspondence between the plan4res and IAMC variable names

PostTreatPlan4res.py can be run either by using the script LaunchXXX or by the command :

Python -W ignore PostTreatPlan4res.py settingsPostTreatPlan4res.yml settings_format_xxx.yml settingsCreateInputPlan4res.yml