European Commission

JRC TECHNICAL REPORTS

Integrated modelling of future EU power and heat systems

The Dispa-SET v2.2 open-source model

Konstantinos KAVVADIAS
Ignacio HIDALGO GONZALEZ
Andreas ZUCKER
Sylvain QUOILIN

2018

EUR 29085 EN

Joint Research Centre

Contact Information
Name: A. Zucker
Address: European Commission, Joint Research Centre, Westerduinweg 3, P.O. Box 2, 1755 ZG Petten, Netherlands
E-mail: andreas.zucker@ec.europa.eu
Tel.: +31 224 56 5059

JRC Science Hub
https://ec.europa.eu/jrc

JRCJRC110305

EUR 29085 EN

PDF ISBN 978-92-79-77866-7 ISSN 1831-9424 doi:10.2760/860626

Luxembourg: Publications Office of the European Union, 2018

How to cite this report: Kavvadias K., Hidalgo Gonzalez I., Zucker A., Quoilin S., Integrated modelling of future EU power and heat systems – The Dispa-SET 2.2 open-source model, EUR 29085 EN, Publications Office of the European Union, Luxembourg, 2018, ISBN:978-92-79-77866-7, doi:10.2760/860626, JRC110305

Integrated modelling of future EU power and heat systems

Abstract
This report describes the implementation of the Dispa-SET model version 2.2. It extensively describes the model equations, the model inputs, and the resolution process. This version of Dispa-SET focuses more specifically on the inclusion of the heating sector, with a new dedicated module. It allows simulating the potential interactions between heat and power and the exploitation of thermal storage as a flexible resource. The model is an open-source tool and comes with an open dataset for testing purposes. It can therefore be freely re-used or modified to fit the needs of a particular case study.

Contents

1. Introduction

The development of Dispa-SET was initiated in 2014 at the Joint Research Centre of the European Commission. The aim was to provide an open and transparent pan-european power system model, following JRC's commitment on open modeling and data transparency [Pfenninger et al., 2017]. It has now reached a mature state and allows simulating current and future EU power systems in a detailed manner. That model is currently used by a growing number of researcher in the EU.

This year's developments follow a need, driven by policies, for models describing the interrelationship between power, heat and an increasing share of non-dispatchable renewable energy sources. According to a very recent comprehensive mapping of power system models [Fernandez B. C. et al., 2017] less than half of the identified models, present a link with other sectors, mostly with gas or heat sectors. It was therefore decided to add a heat module to the JRC's unit commitment and dispatch model Dispa-SET.

This module allows the simulation of the links between heating and power sectors. The considered technologies are CHP power plants, heat pumps and electrical boilers. A practical example is provided with the model, illustrating the implementation and typical results of such simulation.

The new version maintains full compatibility with previous Dispa-SET model which is used by JRC and other researchers. This report is released together with the version 2.2 of Dispa-SET and contains both the documentation and practical user guides to run the model.

1.1 Previous reports

This report is an update of previous JRC technical reports:

- Dispa-SET 2.0: unit commitment and power dispatch model.
- Modelling Future EU Power Systems Under High Shares of Renewables.

JRC TECHNICAL REPORTS

Modelling Future EU Power Systems
Under High Shares of Renewables

The Dispa-SET 2.1
open-source model

Sylvain QUOILIN
Ignacio HIDALGO GONZALEZ
Andreas ZUCKER

2017

1.2 Dispa-SET in the scientific literature

In the past years, Dispa-SET has been used in various scientific works covering different geographical areas and with different focus points. The works for which scientific articles have been published are summarized hereunder:

- [Sánchez Pérez, 2017]: Evaluating the contribution of hydropower for flexibility services the European power system.
- [Quoilin et al., 2015, Quoilin et al., 2017b]: Ongoing work aiming at coupling the JRC-EU-TIMES model with Dispa-SET by generating simplified variable RES flexibility constraints.
- [Beltramo et al., 2017]: Evaluation of the impact of Electric Vehicle deployment in The Netherlands.
- [Pavičević et al., 2017, Tomić et al., 2017] Open-source model of the Balkans area, with some simulations involving high shares of renewables.

- [Quoilin et al., 2014]: Evaluation of the available technical flexibility to balance variable RES generation in Belgium

1.3 Downloading Dispa-SET

The public version of Dispa-SET can be downloaded in the 2 section or from its github repository (using the Clone or Download button on the right side of the screen):

```
https://github.com/energy-modelling-toolkit/Dispa-SET
```

1.4 Documentation

The model documentation is available by running sphinx in the Docs folder of the project or by consulting the online documentation. The documentation relative to the latest available public version of Dispa-SET can be consulted at:

```
http://www.dispaset.eu
```

1.5 Overview

Organization Joint Research Centre, European Commission,

Version 2.2

The Dispa-SET model is an open-source unit commitment and optimal dispatch model mainly developed within the ”Joint Research Centre” of the European Commission and focuses on the balancing and flexibility problems in European grids

The unit commitment problem consists of two parts: i) scheduling the start-up, operation, and shut down of the available generation units, and ii) allocating (for each period of the simulation horizon of the model) the total power demand among the available generation units in such a way that the overall power system costs is minimized. The first part of the problem, the unit scheduling during several periods of time, requires the use of binary variables in order to represent the start-up and shut down decisions, as well as the consideration of constraints linking the commitment status of the units in different periods. The second part of the problem is the economic dispatch problem, which determines the continuous output of each and every generation unit in the system.

The problem mentioned above can be formulated as a mixed-integer linear program (MILP). The formulation is based upon publicly available modelling approaches [Arroyo and Conejo, 2000, Carrión and Arroyo, 2006]. The goal of the model being the simulation of a large (e.g. European) interconnected power system, a tight and compact formulation has been implemented, in order to simultaneously reduce the region where the solver searches for the solution and increase the speed at which the solver carries out that search. Tightness refers to the distance between the relaxed and integer solutions of the MILP and therefore defines the search space to be explored by the solver, while compactness is related to the amount of data to be processed by the solver and thus determines the speed at which the solver searches for the optimum.

It is written in GAMS and Python (Pyomo) and uses csv files for input data handling. The optimisation is defined as a Linear Programming (LP) or Mixed-Integer Linear Programming (MILP) problem, depending on the desired level of accuracy and complexity. Continuous variables include the individual unit dispatched power, the shedded load and the curtailed power generation. The binary variables are the commitment status of each unit. The main model features can be summarized as follows:

1.6 Features

- Minimum and maximum power for each unit
- Power plant ramping limits
- Reserves up and down

- Minimum up/down times
- Load Shedding
- Curtailment
- Pumped-hydro storage
- Non-dispatchable units (e.g. wind turbines, run-of-river, etc.)
- Start-up, ramping and no-load costs
- Multi-nodes with capacity constraints on the lines (congestion)
- Constraints on the targets for renewables and/or CO2 emissions
- Yearly schedules for the outages (forced and planned) of each units
- CHP power plants and thermal storage
- Different clustering methods

The demand is assumed to be inelastic to the price signal. The MILP objective function is therefore the total generation cost over the optimisation period.

1.7 Libraries used

- pyomo optimisation object library, interface to LP solver (e.g. CPLEX)
- pandas for input and result data handling
- matplotlib for plotting
- GAMS_api for the communication with GAMS

1.8 Public administration reference

This software has been developed initially within the Directorate C Energy, Transport and Climate, which is one of the 7 scientific directorates of the Joint Research Centre (JRC) of the European Commission. Directorate C is based both in Petten, the Netherlands, and Ispra, Italy.

1.9 Licence

Dispa-SET is a free software licensed under the ”European Union Public Licence” EUPL v1.1. It can be redistributed and/or modified under the terms of this license.

1.10 Main Developers

- Sylvain Quoilin (University of Líege)
- Konstantinos Kavvadias (European Commission, Institute for Energy and Transport)

2. Releases

Major stable releases:

- Dispa-SET v2.2
- Dispa-SET v2.1
- Dispa-SET v2.0

2.1 Changelog

2.1.1 Version 2.2

- Inclusion of CHP, power-to-heat and thermal storage (these new features can be tested by running the case study described in the config file: ”ConfigFiles/ConfigCY.xlsx”)
- Bug fixes
- Improved user interface

2.1.2 Version 2.1

- Major refactoring of the folder structure
- New data included in the database
- Inclusion of the LP formulation (in addition to the MILP)

2.1.3 Version 2.0

First public version of the Dispa-SET model.

3. Getting Started

This short tutorial describes the main steps to get a practical example of Dispa-SET running.

3.1 Prerequisites

Install Python 2.7, with full scientific stack. The Anaconda[1] distribution is recommended since it comprises all the required packages. If Anaconda is not used, the following libraries and their dependencies should be installed manually:

- numpy
- pandas (> v0.18.1)
- matplotlib
- xlrd
- pickle

This can be achieved using the pip installer (example for numpy):

```
pip install numpy
```

NB: For Windows users, some packages might require the installation of a C++ compiler for Python. This corresponds to the typical error message: ”Unable to find vcvarsall.bat”. This can be solved by installing the freely available ”Microsoft Visual C++ Compiler for Python 2.7”. In some cases the path to the compiler must be added to the PATH windows environment variable (e.g. C:Program FilesCommon FilesMicrosoftVisual C++ for Python9.0)

3.1.1 Using Dispa-SET with GAMS:

Dispa-SET is primarily designed to run with GAMS and therefore requires GAMS to be installed with a valid user licence. Currently, only the 64-bit version of GAMS is supported in Dispa-SET.

The GAMS api for python has been pre-compiled in the ”Externals” folder and is usable with most operating systems. If the pre-compiled binaries are not available or could not be loaded, the system exits with an error message. In that case, the gams python api should be compiled from the source provided in the GAMS installation folder (e.g. ”C:\GAMS\win64\24.3\apifiles\Python\api”):

```
python gdxsetup.py install
python gamssetup.py install
```

The API requires the path to the gams installation folder. The ”get_gams_path()” function of Dispa-SET performs a system search to automatically detect this path. It case it is not successful, the user is prompted for the proper installation path.

3.1.2 Using Dispa-SET with PYOMO:

- Install pyomo

```
            pip install pyomo
```

- Install a solver and add it to the PATH environment variable. If CPLEX is installed, the ”CPLEX” command should be callable from any command prompt or shell.

[1] https://www.continuum.io/downloads

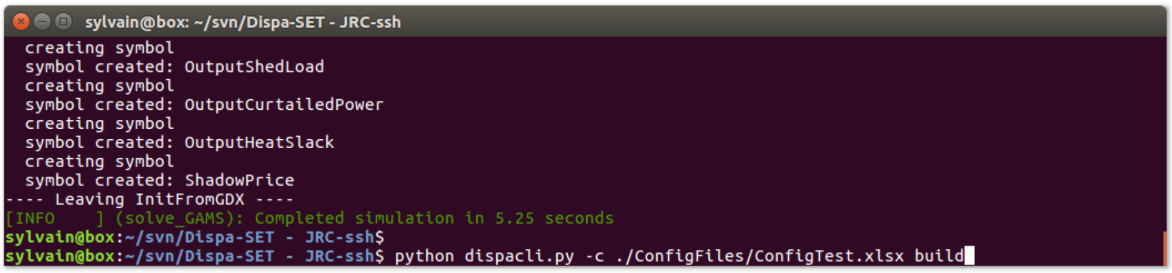
3.2 Step-by-step example of a Dispa-SET run

This section describes the pre-processing and the solving phases of a Dispa-SET run. Three equivalent methods are described in the next sections:

- Using the command line interface
- Using the Dispa-SET API
- Using GAMS

3.2.1 Using the Command Line Interface (CLI)

Dispa-SET can be run from the command line. All commands and options are introduced through the ‘dispacli.py‘ command line interface. To run ‘dispacli.py‘, open a terminal window and change the directory to the Dispa-SET root folder.



```
  creating symbol
  symbol created: OutputShedLoad
  creating symbol
  symbol created: OutputCurtailedPower
  creating symbol
  symbol created: OutputHeatSlack
  creating symbol
  symbol created: ShadowPrice
---- Leaving InitFromGDX ----
[INFO    ] (solve_GAMS): Completed simulation in 5.25 seconds
sylvain@box:~/svn/Dispa-SET - JRC-ssh$
sylvain@box:~/svn/Dispa-SET - JRC-ssh$ python dispacli.py -c ./ConfigFiles/ConfigTest.xlsx build
```

3.2.1.1 Check the configuration file

Dispa-SET runs are defined in dedicated configuration files stored in the ”ConfigFiles” folder. The configuration file ”ConfigTest.xlsx” is provided for testing purposes. It defines a 10-days optimisation problem for a fictitious power system composed of two zones (Z1 and Z2).

3.2.1.2 Pre-processing

From the command line, provide as arguments the configuration file, the solver (Pyomo or GAMS) and the actions to be performed. Within the ”Dispa-SET” folder, run:

```
python dispacli.py -c ./ConfigFiles/ConfigTest.xlsx build
```

The simulation environment folder is defined in the configuration file. In the test example it is set to ”Simulations/simulation_test”. The simulation inputs are written in three different formats: excel (34 excel files), Python (Inputs.p) and GAMS (Inputs.gdx).

3.2.1.3 Run the optimisation using GAMS

Using the GAMS API, the simulation can be started directly from the main Dispa-SET Python file after the pre-processing phase. From the ”Dispa-SET” folder, run:

```
python dispacli.py -g -c ./ConfigFiles/ConfigTest.xlsx simulate
```

This generates the simulation environment, runs the optimisation, and stores the results in the same folder. Note that this can only work if the simulation has been pre-processed before (step 3.2.1.2). It is possible to combine the pre-processing and simulation step in one command:

```
python dispacli.py -g -c ./ConfigFiles/ConfigTest.xlsx build simulate
```

3.2.1.4 Run the optimisation using PYOMO

The same action can be performed using the PYOMO solver. In that case, the ”-g” argument must be changed into ”-p”:

```
python dispacli.py -p -c ./ConfigFiles/ConfigTest.xlsx build simulate
```

3.2.2 Using the Dispa-SET API

The steps to run a model can be also performed directly in Python, by importing the Dispa-SET module. An example file (”build_and_run.py”) is available in the ”scripts/” folder. After checking the configuration file ”ConfigTest.xlsx” (in the ”ConfigFiles” folder), use the following commands in a Python script:

3.2.2.1 Import the Dispa-SET module:

```
import DispaSET as ds
```

3.2.2.2 Load the configuration file:

```
config = ds.load_config_excel('ConfigFiles/ConfigTest.xlsx')
```

3.2.2.3 Build the simulation environment:

```
SimData = ds.build_simulation(config)
```

3.2.2.4 Run the solver:

Using PYOMO:

```
out = ds.solve_pyomo(config['SimulationDirectory'])
```

Using GAMS:

```
out = ds.solve_GAMS(config['SimulationDirectory'], config['GAMS_folder'])
```

A more detailed description of the Dispa-SET functions in available in the API section.

3.2.3 Using GAMS

It is sometimes useful to run Dispa-SET directly in the GAMS IDE(e.g. for debugging purposes). In that case, the pre-processing must be run first (steps 3.2.1.2 or 3.2.2.1, 3.2.2.2 and 3.2.2.3) and the GAMS file generated in the simulation folder can be used to run the optimisation.

3.2.3.1 Using the GAMS graphical user interface:

From the simulation folder (defined in the config file), the Dispa-SET model can be run following the instructions below:

1. Open the UCM.gpr project file in GAMS
2. From GAMS, open the UCM_h.gms model file
3. Run the model in GAMS.

The result file is written to the 'Results.gdx' file which is stored in the 'Simulation' folder, together with all input files.

3.2.3.2 Using the GAMS command line:

GAMS can also be run from the command line (this is the only option for the Linux and MacOS versions).

1. Make sure that the GAMS binary is in the system PATH
2. From the simulation environment folder, run:

```
            gams UCM_h.gms
```

3.3 Postprocessing

Various functions and tools are provided within the "PostProcessing.py" module to load, analyse and plot the simulation results. The use of these functions is illustrated in the "Read_results_notebook.ipynb" Notebook or in the "read_results.py" script. The type of results provided by the post-processing is illustrated hereunder.

In case the simulation was performed in GAMS, the results should be loaded from the Results.gdx file and properly formatted. This can be achieved using the dedicated functions:

```
import DispaSET as ds
inputs,results = ds.GetResults(path='Simulations/simulation_test/')
datain = ds.ds_to_df(inputs)
```

The power dispatch can then be plotted for each simulated zone. In this plot, the units are aggregated by fuel type. The power consumed by storage units and the exportations are indicated as negative values (cfr Figure 1). It is also interesting to display the results at the unit level to gain deeper insights regarding the dispatch. In that case, a plot is generated, showing the commitment status of all units in a zone at each timestep (Figure 2). Both the dispatch plot and the commitment plot can be called using the *CountryPlots* function.

```
ds.CountryPlots(inputs,results,'DE')
```

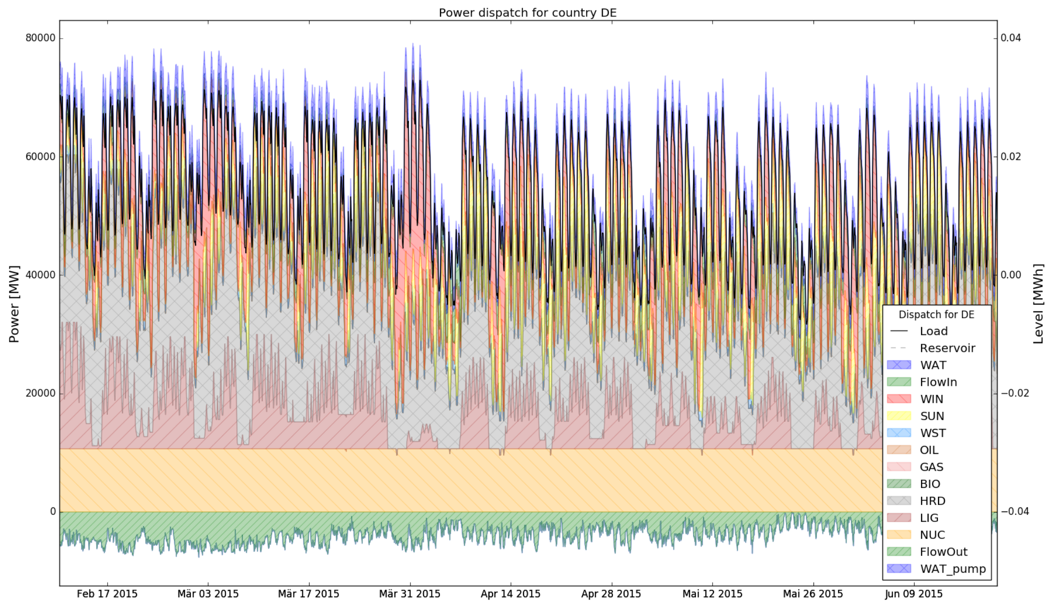


Figure 1: Example result: Power Dispatch for Germany, disaggregated by fuel type

Some aggregated statistics on the simulations results can also be obtained, including the number of hours of congestion in each interconnection line, the yearly energy balances for each zone, the amount of lost load, etc (Figure 3):

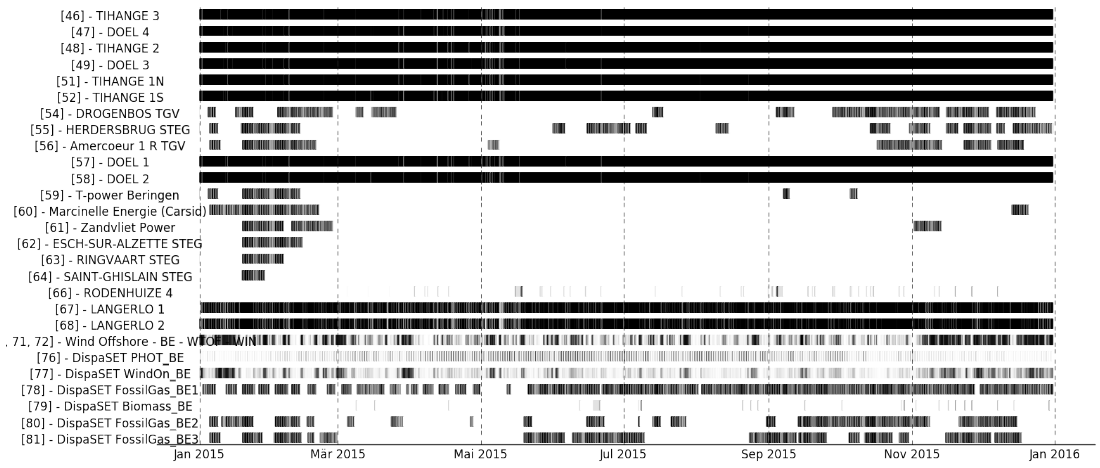
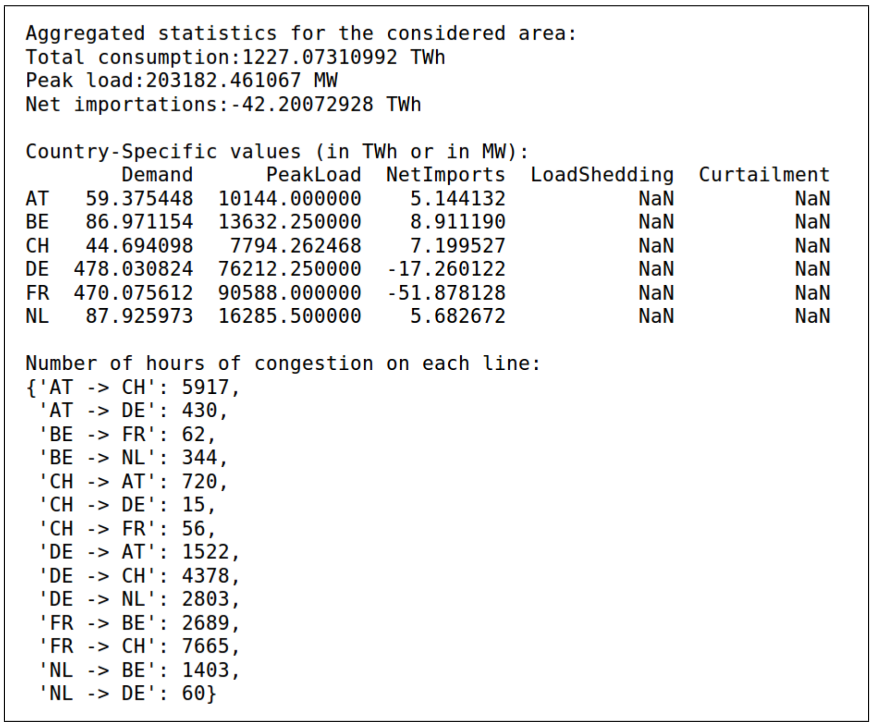


Figure 2: Example result: Commitment and power level status of each unit in Belgium (including Luxemburg)

```
r = ds.ResultAnalysis(inputs,results)
```



```
Aggregated statistics for the considered area:
Total consumption:1227.07310992 TWh
Peak load:203182.461067 MW
Net importations:-42.20072928 TWh

Country-Specific values (in TWh or in MW):
        Demand      PeakLoad  NetImports  LoadShedding  Curtailment
AT   59.375448  10144.000000    5.144132           NaN          NaN
BE   86.971154  13632.250000    8.911190           NaN          NaN
CH   44.694098   7794.262468    7.199527           NaN          NaN
DE  478.030824  76212.250000  -17.260122           NaN          NaN
FR  470.075612  90588.000000  -51.878128           NaN          NaN
NL   87.925973  16285.500000    5.682672           NaN          NaN

Number of hours of congestion on each line:
{'AT -> CH': 5917,
 'AT -> DE': 430,
 'BE -> FR': 62,
 'BE -> NL': 344,
 'CH -> AT': 720,
 'CH -> DE': 15,
 'CH -> FR': 56,
 'DE -> AT': 1522,
 'DE -> CH': 4378,
 'DE -> NL': 2803,
 'FR -> BE': 2689,
 'FR -> CH': 7665,
 'NL -> BE': 1403,
 'NL -> DE': 60}
```

Figure 3: Example output of the *ResultAnalysis* function

The yearly energy balance per fuel or per technology can be also plotted in order to compare the energy mix in each zone. This plot can be created using the EnergyBarPlot function, the output being shown in Figure 4.

```
PPindicators = ds.PerPowerPlantIndicators(inputs,results)
ax = ds.EnergyBarPlot(datain,results,PPindicators)
```

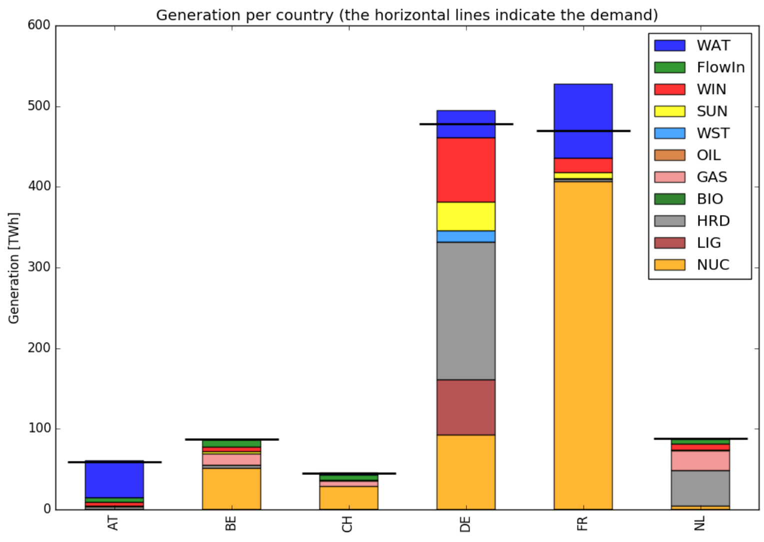


Figure 4: Example result: Energy balance per simulated country

4. Model Description

4.1 Approach

The aim of this model is to represent with a high level of detail the short-term operation of large-scale power systems solving the so-called unit commitment problem. To that aim we consider that the system is managed by a central operator with full information on the technical and economic data of the generation units, the demands in each node, and the transmission network.

The unit commitment problem considered in this report is a simplified instance of the problem faced by the operator in charge of clearing the competitive bids of the participants into a wholesale day-ahead power market. In the present formulation the demand side is an aggregated input for each node, while the transmission network is modelled as a transport problem between the nodes (that is, the problem is network-constrained but the model does not include the calculation of the optimal power flows).

The unit commitment problem consists of two parts: i) scheduling the start-up, operation, and shut down of the available generation units, and ii) allocating (for each period of the simulation horizon of the model) the total power demand among the available generation units in such a way that the overall power system costs is minimized. The first part of the problem, the unit scheduling during several periods of time, requires the use of binary variables in order to represent the start-up and shut down decisions, as well as the consideration of constraints linking the commitment status of the units in different periods. The second part of the problem is the so-called economic dispatch problem, which determines the continuous output of each and every generation unit in the system. Therefore, given all the features of the problem mentioned above, it can be naturally formulated as a mixed-integer linear program (MILP).

Since our goal is to model a large European interconnected power system, we have implemented a so-called tight and compact formulation, in order to simultaneously reduce the region where the solver searches for the solution and increase the speed at which the solver carries out that search. Tightness refers to the distance between the relaxed and integer solutions of the MILP and therefore defines the search space to be explored by the solver, while compactness is related to the amount of data to be processed by the solver and thus determines the speed at which the solver searches for the optimum. Usually tightness is increased by adding new constraints, but that also increases the size of the problem (decreases compactness), so both goals contradict each other and a trade-off must be found.

4.2 Structure

The Dispa-SET model contains a mixed integer linear problem (MILP) and a simplified linear program (LP). Alternative implementations (for both the MILP and LP models) exist in both GAMS and PYOMO. The formulation of the MIP is based on publicly available modelling approaches [Arroyo and Conejo, 2000, Carrión and Arroyo, 2006].

In the MILP, continuous variables include the individual units' dispatched power, the shed load and the curtailed power generation in every time step. The binary variables represent the commitment status of each unit.

The main model features inputs consists of:

- Minimum and maximum power output for each unit
- Power plant ramping limits
- Reserves up and down
- Minimum up/down times
- Pumped-hydro storage
- Non-dispatchable units (e.g. wind turbines, run-of-river, etc.)
- Start-up, ramping and no-load costs
- Multi-nodes with capacity constraints on the lines (congestion)
- Constraints on the targets for renewables and/or CO2 emissions
- Yearly schedules for the outages (forced and planned) of each units
- Yearly schedules for the levels of large water reservoirs

The main model output is given by:

- Commitment status of each unit
- Dispatched power of each unit
- Load Shedding
- Curtailment

4.3 Data

As an optimisation model, Dispa-SET is making use of three data types: Sets, Parameters and optimisation variables.

4.3.1 Sets

The sets are the basic building blocks of the optimisation model, corresponding exactly to the indices in the algebraic representations of models. The sets defined in Dispa-SET are listed in Table 1.

4.3.2 Parameters

Parameters are constant coefficients of the optimisation mode. They correspond to the exogenous data provided to the model. In GAMS parameters are explicitly declared as "Parameter" and in Pyomo by creating a Param() object inside the model object. The list of Dispa-SET parameters is provided in Table 2.

Table 1: List of the sets defined in the optimisation model

Name	Description
f	Fuel types
h	Hours
i	Time step in the current optimisation horizon
l	Transmission lines between nodes
mk	{DA: Day-Ahead, 2U: Reserve up, 2D: Reserve Down}
n	Zones within each country (currently one zone, or node, per country)
p	Pollutants
t	Power generation technologies
tr	Renewable power generation technologies
u	Units
s(u)	Storage units (including hydro reservoirs)
chp(u)	CHP units

4.3.3 Optimisation variables

The optimisation (or decision) variables are those variables that need to be adjusted to minimise the objective function. Decision variables are explicitly declared as "Variable: in GAMS and in Pyomo by defining an Var() object inside the model object. They are defined in Table 3.

4.4 Equations

4.4.1 Objective Function

The goal of the unit commitment problem is to minimise the total power system costs (expressed in EUR in equation 1), which are defined as the sum of different cost items, namely: start-up and shut-down, fixed, variable, ramping, transmission-related and load shedding (voluntary and involuntary) costs.

$$\begin{aligned}
&min \sum_{u,n,i} \\
&\Big[CostStartUp_{u,i} + CostShutDown_{u,i} + CostFixed_u \cdot Committed_{u,i} \\
&+ CostVariable_{u,i} \cdot Power_{u,i} + CostRampUp_{u,i} + CostRampDown_{u,i} \\
&+ PriceTransimission_{i,l} \cdot Flow_{i,l} + (CostLoadShedding_{i,n} \cdot ShedLoad_{i,n}) \\
&+ CostHeatSlack_{chp(u),i} \cdot HeatSlack_{chp(u),i}) \\
&+ CostVariable_{chp(u),i} \cdot CHPPowerLossFactor_{chp(u)} \cdot Heat_{chp(u),i}) \\
&+ VOLL_{Power} \cdot (LostLoadMaxPower_{i,n} + LostLoadMinPower_{i,n}) \\
&+ VOLL_{Reserve} \cdot (LostLoadReserve2U_{i,n} + LostLoadReserve2D_{i,n}) \\
&+ VOLL_{Ramp} \cdot (LostLoadRampUp_{u,i} + LostLoadRampDown_{u,i})\Big]
\end{aligned} \tag{1}$$

The costs can be broken down as:

- Fixed costs: independent on whether the unit is on or off.
- Variable costs: proportional to the power output of the units (e.g. fuel costs)
- Start-up costs: due to the start-up of a unit.
- Shut-down costs: due to the shut-down of a unit.
- Ramp-up: emerging from the ramping up of a unit.
- Ramp-down: emerging from the ramping down of a unit.
- Shed load: due to necessary load shedding.
- Transmission: depending of the flow transmitted through the lines.

Table 2: Dispa-SET Parameters

Name	Units	Description
AvailabilityFactor(u,i)	%	Percentage of nominal capacity available
CHPPowerLossFactor(u)	%	Power loss when generating heat
CHPPowerToHeat(u)	%	Nominal power-to-heat factor
CHPMaxHeat(chp)	MW	Maximum heat capacity of chp plant
CHPType	n.a.	CHP Type
CommittedInitial(u)	n.a.	Initial commitment status
CostFixed(u)	EUR/h	Fixed costs
CostLoadShedding(n,h)	EUR/MWh	Shedding costs
CostRampDown(u)	EUR/MW	Ramp-down costs
CostRampUp(u)	EUR/MW	Ramp-up costs
CostShutDown(u)	EUR/h	Shut-down costs
CostStartUp(u)	EUR/h	Start-up costs
CostVariableH(u,i)	EUR/MWh	Variable costs
Curtailment(n)	n.a.	Curtailment {binary: 1 allowed}
Demand(mk,n,i)	MW	Hourly demand in each zone
Efficiency(u)	%	Power plant efficiency
EmissionMaximum(n,p)	EUR/tP	Emission limit per zone for pollutant p
EmissionRate(u,p)	tP/MW	Emission rate of pollutant p from unit u
FlexibilityDown(u)	MW/h	Available fast shut-down ramping capacity
FlexibilityUp(u)	MW/h	Available fast start-up ramping capacity
Fuel(u,f)	n.a.	Fuel type used by unit u {binary: 1 u uses f}
LineNode(l,n)	n.a.	Line-zone incidence matrix {-1,+1}
LoadMaximum(u,h)	%	Maximum load for each unit
LoadShedding(n,h)	MW	Load that may be shed per zone in 1 hour
Location(u,n)	n.a.	Location {binary: 1 u located in n}
OutageFactor(u,h)	%	Outage factor (100 % = full outage) per hour
PartLoadMin(u)	%	Percentage of minimum nominal capacity
PowerCapacity(u)	MW	Installed capacity
PowerInitial(u)	MW	Power output before initial period
PowerMinStable(u)	MW	Minimum power for stable generation
PowerMustRun(u)	MW	Minimum power output
PriceTransmission(l,h)	EUR/MWh	Price of transmission between zones
RampDownMaximum(u)	MW/h	Ramp down limit
RampShutDownMaximum(u)	MW/h	Shut-down ramp limit
RampStartUpMaximum(u)	MW/h	Start-up ramp limit
RampUpMaximum(u)	MW/h	Ramp up limit
Reserve(t)	n.a.	Reserve provider {binary}
StorageCapacity(s)	MWh	Storage capacity (reservoirs)
StorageChargingCapacity(s)	MW	Maximum charging capacity
StorageChargingEfficiency(s)	%	Charging efficiency
StorageDischargeEfficiency(s)	%	Discharge efficiency
StorageInflow(s,h)	MWh	Storage inflows
StorageInitial(s)	MWh	Storage level before initial period
StorageMinimum(s)	MWh	Minimum storage level
StorageOutflow(s,h)	MWh	Storage outflows (spills)
StorageProfile(u,h)	MWh	Storage long-term level profile
Technology(u,t)	n.a.	Technology type {binary: 1: u belongs to t}
TimeDownInitial(u)	h	Hours down before initial period
TimeDownLeftInitial(u)	h	Time down remaining at initial time
TimeDownLeftJustStopped(u,i)	h	Time down remaining if started at time i
TimeDownMinimum(u)	h	Minimum down time
TimeDown(u,h)	h	Number of hours down
TimeUpInitial(u)	h	Number of hours up before initial period
TimeUpLeftInitial(u)	h	Time up remaining at initial time
TimeUpLeftJustStarted(u,i)	h	Time up remaining if started at time i
TimeUpMinimum(u)	h	Minimum up time
TimeUp(u,h)	h	Number of hours up
VOLL ()	EUR/MWh	Value of lost load

Table 3: Dispa-SET Variables

Name	**Units**	**Description**
Committed(u,h)	n.a.	Unit committed at hour h {1,0}
CostStartUpH(u,h)	EUR	Cost of starting up
CostShutDownH(u,h)	EUR	Cost of shutting down
CostRampUpH(u,h)	EUR	Ramping cost
CostRampDownH(u,h)	EUR	Ramping cost
CurtailedPower(n,h)	MW	Curtailed power at node n
Flow(l,h)	MW	Flow through lines
Heat(chp,h)	MW	Heat output by chp plant
HeatSlack(chp,h)	MW	Heat satisfied by other sources
MaxRamp2U(u,h)	MW/h	Maximum 15-min Ramp-up capbility
MaxRamp2D(u,h)	MW/h	Maximum 15-min Ramp-down capbility
Power(u,h)	MW	Power output
PowerMaximum(u,h)	MW	Power output
PowerMinimum(u,h)	MW	Power output
ShedLoad(n,h)	MW	Shed load
StorageInput(s,h)	MWh	Charging input for storage units
StorageLevel(s,h)	MWh	Storage level of charge
Spillage(s,h)	MWh	Spillage from water reservoirs
SystemCostD	EUR	Total system cost for one optimisation period
LostLoadMaxPower(n,h)	MW	Deficit in terms of maximum power
LostLoadRampUp(u,h)	MW	Deficit in terms of ramping up for each plant
LostLoadRampDown(u,h)	MW	Deficit in terms of ramping down
LostLoadMinPower(n,h)	MW	Power exceeding the demand
LostLoadReserve2U(n,h)	MW	Deficit in reserve up

- Loss of load: power exceeding the demand or not matching it, due to insufficient ramping capability or loss of reserves.

The variable production costs (in EUR/MWh), are determined by fuel and emission prices corrected by the thermal efficiency (which is considered to be constant for all levels of output in this version of the model) and the emission rate of the unit:

$$\begin{gathered} CostVariable_{u,h} = \\ Markup_{u,h} + \sum_{n,f}\left(\frac{Fuel_{u,f}\cdot FuelPrice_{n,f,h}\cdot Location_{u,n}}{Efficiency_u}\right) \\ + \sum_p \left(EmissionRate_{u,p}\cdot PermitPrice_p\right) \end{gathered} \tag{2}$$

The variable cost includes an additional mark-up parameter that can be used for calibration and validation purposes.

$$\begin{gathered} i = 1: \\ CostStartUp_{u,i} \geq CostStartUp_u \cdot (Committed_{u,i} - CommittedInitial_u) \\ CostShutDown_{u,i} \geq CostShutDown_u \cdot (CommittedInitial_u - Committed_{u,i}) \\ i > 1: \\ CostStartUp_{u,i} \geq CostStartUp_u \cdot (Committed_{u,i} - Committed_{u,i-1}) \\ CostShutDown_{u,i} \geq CostShutDown_u \cdot (Committed_{u,i-1} - Committed_{u,i}) \end{gathered} \tag{3}$$

In the previous equation, as in some of the following, a distinction is made between the equation for the first and subsequent periods. The equation for the first period takes into account the commitment status of the unit before the beginning of the simulation, which is part of the information fed into the model.

Ramping costs are computed in the same manner:

$$
\begin{gathered}
i = 1: \\
CostRampUpH_{u,i} \geq CostRampUp_u \cdot (Power_{u,i} - PowerInitial_u) \\
CostRampDownH_{u,i} \geq CostRampDown_u \cdot (PowerInitial_u - Power_{u,i}) \\
i > 1: \\
CostRampUpH_{u,i} \geq CostRampUp_u \cdot (Power_{u,i} - Power_{u,i-1}) \\
CostRampDownH_{u,i} \geq CostRampDown_u \cdot (Power_{u,i-1} - Power_{u,i})
\end{gathered} \tag{4}
$$

It should be noted that in case of start-up and shut-down, the ramping costs are added to the objective function. Using start-up, shut-down and ramping costs at the same time should therefore be performed with care.

In the current formulation, all other costs (fixed and variable costs, transmission costs, load shedding costs) are considered as exogenous parameters.

As regards load shedding, the model considers the possibility of voluntary load shedding resulting from contractual arrangements between system operators and consumers. Additionally, in order to facilitate tracking and debugging of errors, the model also considers some variables representing the situation when the capacity of the system is not able to provide the minimum/maximum power, reserve, or when the ramping constraints are reached. These lost loads are a very expensive last resort of the system used when there is no other choice available. The different lost loads are assigned very high costs (with respect to any other costs). This allows running the simulation without infeasibilities, thus helping to detect the origin of the loss of load. In a normal run of the model, without errors, the LostLoad variables are expected to be equal to zero.

4.4.2 Demand-related constraints

The main constraint to be met is the supply-demand balance, for each period and each zone, in the day-ahead market. According to this restriction, the sum of all the power produced by all the units present in the node (including the power generated by the storage units), and the power injected from neighbouring nodes is equal to the load in that node, plus the power consumed for energy storage, minus the shed load.

$$
\begin{gathered}
\sum_u (Power_{u,i} \cdot Location_{u,n}) \\
+ \sum_l (Flow_{l,i} \cdot LineNode_{l,n}) \\
= Demand_{DA,n,h} + \sum_r \left(StorageInput_{s,h} \cdot Location_{s,n}\right) \\
- ShedLoad_{n,i} \\
- LostLoadMaxPower_{n,i} + LostLoadMinPower_{n,i}
\end{gathered} \tag{5}
$$

Besides that balance, the reserve requirements (upwards and downwards) in each node must be met as well. In Dispa-SET, the reserve requirements are defined as an aggregation of secondary and tertiary reserves, which are typically brought online in periods shorter than an hour, the time step of this model. Therefore, additional equations and constraints are defined to account for the up/down ramping requirements, by computing the ability of each unit to adapt its power output within a period of 15 min.

For each power plant, the ability to increase its power (in MW/h) is the ramp-up capability if it is already committed or the nominal power if its starting time is lower than 15 minutes. This is to take into account that fast starting units could provide reserve (e.g. hydro units for secondary reserve, gas turbine for tertiary reserve).

$$
\begin{gathered}
MaxRamp2U_{u,i} \\
\leq RampUpMaximum_u \cdot Committed_{u,i} \\
+ FlexibilityUp_u \cdot (1 - Committed_{u,i})
\end{gathered} \tag{6}
$$

where FlexibilityUp is the maximum flexibility (in MW/h) that can be provided by the unit in 15 min in case of cold start:

$$\begin{gathered}\text{If } RampStartUpMaximum_u \geq PowerMinStable_u \cdot 4 \\ \text{Then } FlexibilityUp_u = RampStartUpMaximum_u \\ \text{Else } FlexibilityUp_u = 0\end{gathered} \tag{7}$$

where the factor 4 is used to convert the ramping rate from MW/15min to MW/h.

The maximum ramping rate is also limited by the available capacity margin between current and maximum power output:

$$\begin{gathered}MaxRamp2U_{u,i} \leq (PowerCapacity_u \cdot AvailabilityFactor_{u,i} \\ \cdot (1 - OutageFactor_{u,i}) - Power_{u,i}) \cdot 4\end{gathered} \tag{8}$$

The same applies to the 15 min ramping down capabilities:

$$\begin{gathered}MaxRamp2D_{u,i} \\ \leq max\left(RampDownMaximum_u, FlexibilityDown_u\right) \cdot Committed_{u,i}\end{gathered} \tag{9}$$

The parameter FlexibilityDown is defined as the maximum ramp down rate at which the unit can shut down in 15 minutes. In case the unit cannot be shut-down in 15 minutes (and only in this case) the maximum ramping down capability is limited by the capacity margin between actual and minimum power:

$$\begin{gathered}\text{If} RampShutDownMaximum_u < PowerMinStable_u \cdot 4 \\ \text{Then} MaxRamp2D_{u,i} \leq (Power_{u,i} - PowerMinStable_u \cdot Committed_{u,i}) \cdot 4 \\ Else: \\ MaxRamp2D_{u,i} \leq Power_{u,i} \cdot 4\end{gathered} \tag{10}$$

The reserve requirements are defined as a time series by the users. In case no input is provided a default formula is used to evaluate the needs for secondary reserves as a function of the maximum expected load for each day. The default formula is described by:

$$Demand_{2U,n,i} = \sqrt{10 \cdot \max_h\left(Demand_{DA,n,h}\right) + 150^2} - 150 \tag{11}$$

Downward reserves are defined as 50% of the upward margin:

$$Demand_{2D,n,h} = 0.5 \cdot Demand_{2U,n,h} \tag{12}$$

The reserve demand should be fulfilled at all times by all the plants allowed to participate in the reserve market:

$$\begin{gathered}Demand_{2U,n,h} \\ \leq \sum_{u,t}\left(MaxRamp2U_{u,i} \cdot Technology_{u,t} \cdot Reserve_t \cdot Location_{u,n}\right) \\ + LostLoadReserve2UH_{n,i}\end{gathered} \tag{13}$$

The same equation applies to downward reserve requirements (2D).

4.4.3 Power output bounds

The minimum power output is determined by the must-run or stable generation level of the unit if it is committed:

$$\begin{gathered}PowerMustRun_{u,i} \cdot Committed_{u,i} \\ \leq Power_{u,i}\end{gathered} \tag{14}$$

On the other hand, the output is limited by the available capacity, if the unit is committed:

$$\begin{gathered}Power_{u,i} \\ \leq PowerCapacity_u \cdot AvailabilityFactor_{u,i} \\ \cdot(1 - OutageFactor_{u,i}) \cdot Committed_{u,i}\end{gathered} \tag{15}$$

The availability factor is used for renewable technologies to set the maximum time-dependent generation level. It is set to one for the traditional power plants. The outage factor accounts for the share of unavailable power due to planned or unplanned outages.

The power output in a given period also depends on the output levels in the previous and the following periods and on the ramping capabilities of the unit. If the unit was down, the ramping capability is given by the maximum start up ramp, while if the unit was online the limit is defined by the maximum ramp up rate. Those bounds are given with respect to the previous time step by the equation:

$$\begin{gathered}
i = 1: \\
Power_{u,i} \leq \\
PowerInitial_u \\
+ CommittedInitial_u \cdot RampUpMaximum_u \\
+ (1 - CommittedInitial_u) \cdot RampStartUpMaximum_u \\
+ LostLoadRampUp_{u,i} \\
i > 1: \\
Power_{u,i} \leq \\
Power_{u,i-1} \\
+ Committed_{u,i-1} \cdot RampUpMaximum_u \\
+ (1 - Committed_{u,i-1}) \cdot RampStartUpMaximum_u \\
+ LostLoadRampUp_{u,i}
\end{gathered} \tag{16}$$

Where the LoadMaximum parameter is calculated taking into account the availability factor and the outage factor:

$$LoadMaximum_{u,h} = AvailabilityFactor_{u,h} \cdot (1 - OutageFactor_{u,h}) \tag{17}$$

Similarly, the ramp down capability is limited by the maximum ramp down or the maximum shut down ramp rate:

$$\begin{gathered}
i = 1: \\
PowerInitial_u - Power_{u,i} \leq \\
Committed_{u,i} \cdot RampDownMaximum_u \\
+ (1 - Committed_{u,i}) \cdot RampShutDownMaximum_u \\
+ LostLoadRampDown_{u,i} \\
i > 1: \\
Power_{u,i-1} - Power_{u,i} \leq \\
Committed_{u,i} \cdot RampDownMaximum_u \\
+ (1 - Committed_{u,i}) \cdot RampShutDownMaximum_u \\
+ LostLoadRampDown_{u,i}
\end{gathered} \tag{18}$$

4.4.4 Minimum up and down times

The operation of the generation units is also limited as well by the amount of time the unit has been running or stopped. Due to technical limitations and in order to avoid excessive ageing of the generators, once a unit is started up, it cannot be shut down immediately. Reciprocally, if the unit is shut down it may not be started immediately.

That is, the value of the time counter with respect to the minimum up time and down times determines the commitment status of the unit. In order to model theses constraints linearly, it is necessary to keep track of the number of hours the unit must be online at the beginning of the simulation:

$$\begin{gathered}
TimeUpLeftInitial_u = \\
min\left\{N, (TimeUpMinimum_u - TimeUpInitial_u) \cdot CommittedInitial_u\right\}
\end{gathered} \tag{19}$$

where N is the number of time steps in the current optimisation horizon.

If the unit is initially started up, it has to remain committed until reaching the minimum up time:

$$\sum_{i=1}^{TimeUpLeftInitial_u} (1 - Committed_{u,i}) = 0 \tag{20}$$

If the unit is started during the considered horizon, the time it has to remain online is TimeUpMinimum, but cannot exceed the time remaining in the simulated period. This is expressed in the next equation and is pre-calculated for each time step of the period.

$$\begin{gathered} TimeUpLeftJustStarted_{u,i} = \\ min\left\{N - i + 1, TimeUpMinimum_u\right\} \end{gathered} \tag{21}$$

The equation imposing the unit to remain committed is written:

$$\begin{gathered} i = 1 : \\ \sum_{j=i}^{i+TimeUpLeftJustStarted_{u,i}-1} Committed_{u,j} \geq \\ TimeUpLeftJustStarted_{u,i} \cdot (Committed_{u,i} - CommittedInitial_u) \\ i > 1 : \\ \sum_{ii=i}^{i+TimeUpLeftJustStarted_u-1} Committed_{u,ii} \geq \\ TimeUpLeftJustStarted_{u,i} \cdot (Committed_{u,i} - Committed_{u,i-1}) \end{gathered} \tag{22}$$

The same method can be applied to the minimum down time constraint:

$$\begin{gathered} TimeDownLeft_u = \\ min\{N, (TimeDownMinimum_u - TimeDownInitial_u) \\ \cdot(1 - CommittedInitial_u)\} \end{gathered} \tag{23}$$

Related to the initial status of the unit:

$$\sum_{i=1}^{TimeDownLeftInitial_u} Committed_{u,i} = 0 \tag{24}$$

The TimeDownLeftJustStopped parameter is computed by:

$$\begin{gathered} TimeDownLeftJustStopped_{u,i} = \\ min\left\{N - i + 1, TimeDownMinimum_u\right\} \end{gathered} \tag{25}$$

Finally, the equation imposing the time the unit has to remain de-committed is defined as:

$$\begin{gathered} i = 1 : \\ \sum_{ii=i}^{i+TimeDownLeftJustStopped_{i,u}-1} (1 - Committed_{u,ii}) \geq \\ TimeDownLeftJustStopped_{u,i} \cdot (CommittedInitial_u - Committed_{u,i}) \\ i > 1 : \\ \sum_{ii=i}^{i+TimeDownLeftJustStopped_u-1} (1 - Committed_{u,i}) \geq \\ TimeDownLeftJustStopped_{u,i} \cdot (Committed_{u,i-1} - Committed_{u,ii}) \end{gathered} \tag{26}$$

These expressions allow the formulation of the unit commitment problem without the use of additional binary variables to describe the start-up and shut-down of each unit.

4.4.5 Storage-related constraints

Storage units are considered to be subject to the same constraints as non-storage power plants. In addition to those constraints, storage-specific restrictions are added for the set of storage units (i.e. a subset of all units), e.g. large hydro reservoirs and pumped hydro storage units or batteries. These restrictions include the storage capacity, inflow, outflow, charging, charging capacity, charge/discharge efficiencies, etc. Discharging is considered as the standard operation mode and is therefore linked to the Power variable, common to all units.

The first constrain imposes that the energy stored by a given unit is bounded by a minimum value:

$$StorageMinimum_s \leq StorageLevel_{s,i} \tag{27}$$

In the case of a storage unit, the availability factor applies to the charging/discharging power, but also to the storage capacity. The storage level is thus limited by:

$$StorageLevel_{s,i} \leq StorageCapacity_s \cdot AvailabilityFactor_{s,i} \tag{28}$$

The energy added to the storage unit is limited by the charging capacity. Charging is allowed only if the unit is not producing (discharging) at the same time (i.e. if the Committed variable is equal to 0).

$$\begin{gathered} StorageInput_{s,i} \leq StorageChargingCapacity_s \\ \cdot AvailabilityFactor_{s,i} \cdot (1 - Committed_{s,i}) \end{gathered} \tag{29}$$

Discharge is limited by the level of charge of the storage unit:

$$\begin{gathered} \frac{Power_{i,s}}{StorageDischargeEfficiency_s} + StorageOutflow_{s,i} \\ + Spillage_{s,i} - StorageInflow_{s,i} \\ \leq StorageLevel_{s,i} \end{gathered} \tag{30}$$

Charge is limited by the level of charge of the storage unit:

$$\begin{gathered} StorageInput_{s,i} \cdot StorageChargingEfficiency_s \\ - StorageOutflow_{s,i} - Spillage_{s,i} \\ + StorageInflow_{s,i} \\ \leq StorageCapacity_s - StorageLevel_{s,i} \end{gathered} \tag{31}$$

Besides, the energy stored in a given period is given by the energy stored in the previous period, net of charges and discharges:

$$\begin{gathered} i = 1: \\ StorageLevelInitial_s + StorageInflow_{s,i} \\ + StorageInput_{s,i} \cdot StorageChargingEfficiency_s \\ = StorageLevel_{s,i} + StorageOutflow_{s,i} + \frac{Power_{s,i}}{StorageDischargeEfficiency_s} \\ i > 1: \\ StorageLevel_{s,i-1} + StorageInflow_{s,i} \\ + StorageInput_{s,i} \cdot StorageChargingEfficiency_s \\ = StorageLevel_{s,i} + StorageOutflow_{s,i} + \frac{Power_{s,i}}{StorageDischargeEfficiency_s} \end{gathered} \tag{32}$$

Some storage units are equipped with large reservoirs, whose capacity at full load might be longer than the optimisation horizon. Therefore, a minimum level constraint is required for the last hour of the optimisation, which otherwise would systematically tend to empty the reservoir as much a possible. An exogenous minimum profile is thus provided and the following constraint is applied:

$$StorageLevel_{s,N} \geq min(StorageProfile_{s,N} \cdot AvailabilityFactor_{s,N} \cdot StorageCapacity_{s}, StorageLevel_{s,0} + \sum_{i=1}^{N} InFlows_{s,i}) \quad (33)$$

where StorageProfile is a non-dimensional minimum storage level provided as an exogenous input. The minimum is taken to avoid infeasibilities in case the provided inflows are not sufficient to comply with the imposed storage level at the end of the horizon.

4.4.6 Heat production constraints (CHP plants only)

In DispaSET power plants can be indicated as CHP satisfying one heat demand. Heat Demand (MW) can be covered either by a CHP plant or by alternative heat supply options (Heat Slack). The storage acts as an accumulation of heat storage and it can be considered either as a physical storage hot water vessel or even as the inertia of the heat network.

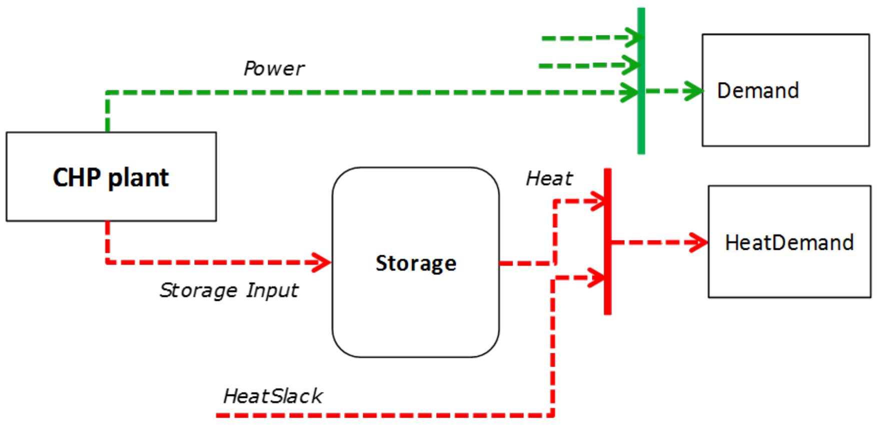


Figure 5: Conceptual scheme of the CHP energy flows including storage

End use heat demand has to be satisfied either by the heat storage or by an alternative heat supply:

$$Heat(chp, i) + HeatSlack(chp, i) = HeatDemand(chp, i) \quad (34)$$

The produced heat (StorageInput) has to be lower than the maximum heat capacity of the CHP unit ($CHPMaxHeat$). As $CHPMaxHeat$ is not required to be defined for all CHP plants this constraint is sometimes ignored.

$$StorageInput_{chp,i} \leq CHPMaxHeat_{chp} \quad (35)$$

The constraints between heat and power production differ for each plant design and are explained within the following subsections.

4.4.6.1 Steam plants with backpressure turbine

This option includes steam-turbine based power plants with a backpressure turbine. The feasible operating region is between the points AB in 6. The slope of the line is the heat to power ratio.

$$Power_{chp,i} = StorageInput_{chp,i} \cdot CHPPowerToHeat_{chp}$$

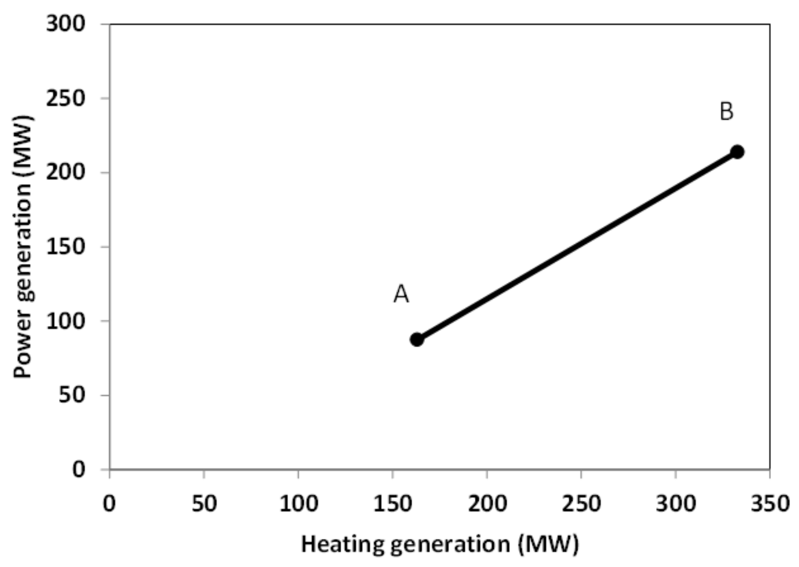


Figure 6: Feasible operating line of a backpressure CHP unit

4.4.6.2 Steam plants with extraction/condensing turbine

This options includes steam-turbine based power plants with an extraction/condensing turbine. The feasible operating region is within ABCDE in 7. The vertical dotted line BC corresponds to the minimum condensation line (as defined by *CHPMaxHeat*). The slope of the DC line is the heat to power ratio and the slope of the AB line is the inverse of the power penalty ratio.

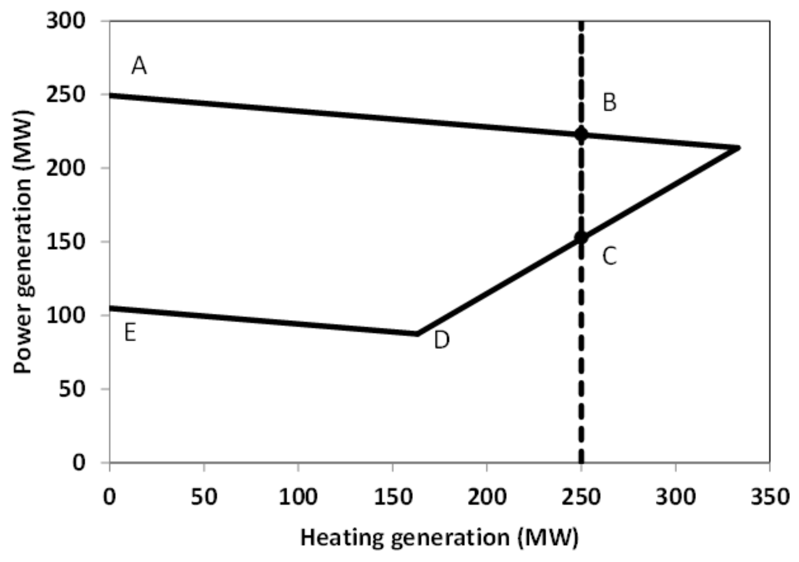


Figure 7: Feasible operating region of an extraction CHP unit

Line DC:

$$Power_{chp,i} \geq StorageInput_{chp,i} \cdot CHPPowerToHeat_{chp} LineAB : \tag{36}$$

$$Power_{chp,i} \leq PowerCapacity_{chp} - StorageInput_{chp,i} \cdot CHPPowerLossFactor_{chp}$$

Line ED:

$$Power_{chp,i} \geq PowerMustRun_{chp,i} - StorageInput_{chp,i} \cdot CHPPowerLossFactor_{chp} \tag{37}$$

4.4.6.3 Power plant coupled with any power to heat option

This option includes power plants coupled with resistance heater or heat pumps. The feasible operating region is between ABCD. The slope of the AB and CD line is the inverse of the COP or efficiency. The vertical dotted line corresponds to the heat pump (or resistance heater) thermal capacity (as defined by *CHPMaxHeat*)

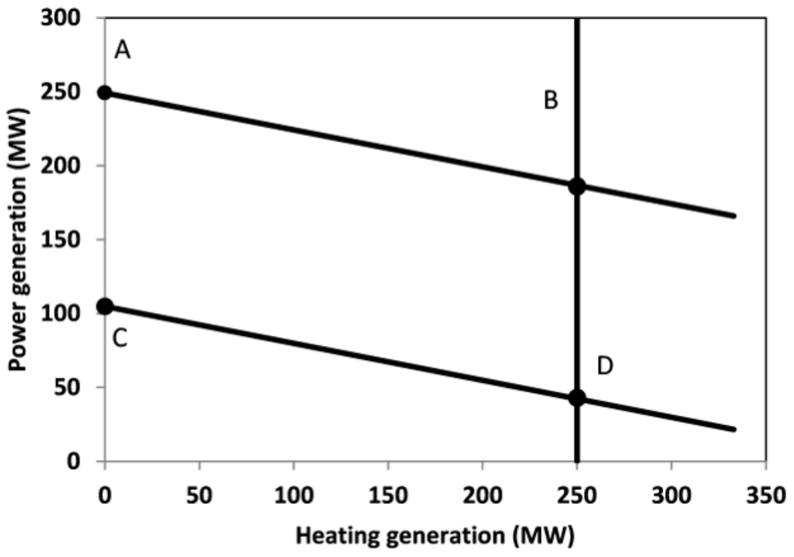


Figure 8: Feasible operating region of power to heat (heat pump or electrical boiler)

Line AB:

$$Power_{chp,i} \leq PowerCapacity_{chp} - StorageInput_{chp,i} \cdot CHPPowerLossFactor_{chp} \tag{38}$$

Line CD:

$$Power_{chp,i} \geq PowerMustRun_{chp,i} - StorageInput_{chp,i} \cdot CHPPowerLossFactor_{chp} \tag{39}$$

4.4.6.4 Heat Storage

Heat storage is modeled in a similar way as electric storage. All variables are in terms of MWh. The energy balance is written:

$$\begin{aligned} i = 1 : \\ StorageInitial_{chp} + StorageInput_{chp,i} = \\ StorageLevel_{chp,i} + Heat_{chp,i} + StorageSelfDischarge_{chp} \cdot StorageLevel_{chp,i}/24 \\ i > 1 : \\ +StorageLevel_{chp,i-1} + StorageInput_{chp,i} = \\ StorageLevel_{chp,i} + Heat_{chp,i} + StorageSelfDischarge_{chp} \cdot StorageLevel_{chp,i}/24 \end{aligned} \tag{40}$$

Storage level must be above a minimum and below storage capacity:

$$StorageMinimum_{chp} \leq StorageLevel_{chp,i} \leq StorageCapacity_{chp} \tag{41}$$

4.4.7 Emission limits

The operating schedule also needs to take into account any cap on the emissions (not only CO2) from the generation units existing in each node:

$$\begin{aligned} \sum_{u} (Power_{u,i} \cdot EmisionRate_{u,p} \cdot Location_{u,n}) \\ \leq EmisionMaximum_{n,p} \end{aligned} \tag{42}$$

It is important to note that the emission cap is applied to each optimisation horizon: if a rolling horizon of one day is adopted for the simulation, the cap will be applied to all days instead of the whole year.

4.4.8 Network-related constraints

The power flow between nodes is limited by the capacities of the transmission lines:

$$\begin{aligned} FlowMinimum_{l,i} &\leq Flow_{l,i} \\ Flow_{l,i} &\leq FlowMaximum_{l,i} \end{aligned} \tag{43}$$

In this model, a simple Net Transfer Capacity (NTC) between countries approach is followed. No DC power flow or Locational Marginal Pricing (LMP) model is implemented.

4.4.9 Curtailment

If curtailment of intermittent generation sources is allowed in one node by setting the corresponding parameters to 1, the amount of curtailed power is bounded by the output of the renewable (tr) units present in that node:

$$\begin{aligned} &CurtailedPower_{n,i} \\ &\leq \sum_{u,tr} \left(Power_{u,i} \cdot Technology_{u,tr} \cdot Location_{u,n} \right) \cdot Curtailment_{n} \end{aligned} \tag{44}$$

4.4.10 Load shedding

If load shedding is allowed in a node, the amount of shed load is limited by the shedding capacity contracted on that particular node (e.g. through interruptible industrial contracts)

$$ShedLoad_{n,i} \leq LoadShedding_{n} \tag{45}$$

5. Optimisation

5.1 Rolling Horizon

The mathematical problem described in the previous sections could in principle be solved for a whole year split into time steps of one hour, but with all likelihood the problem would become extremely demanding in computational terms when attempting to solve the model with a realistically sized dataset. Therefore, the problem is split into smaller optimisation problems that are run recursively throughout the year.

Figure 9 shows an example of such approach, in which the optimisation horizon is one day, with a look-ahead (or overlap) period of one day. The initial values of the optimisation for day j are the final values of the optimisation of the previous day. The look-ahead period is modelled to avoid issues related to the end of the optimisation period such as emptying the hydro reservoirs, or starting low-cost but non-flexible power plants. In this case, the optimisation is performed over 48 hours, but only the first 24 hours are conserved.

Although the previous example corresponds to an optimisation horizon and an overlap of one day, these two values can be adjusted by the user in the Dispa-SET configuration file. As a rule of thumb, the optimisation horizon plus the overlap period should be as least twice the maximum duration of the time-dependent constraints (e.g. the minimum up and down times). In terms of computational efficiency, small power systems can be simulated with longer optimisation horizons, while larger systems benefit from a reduced time horizon, the minimum being one day.

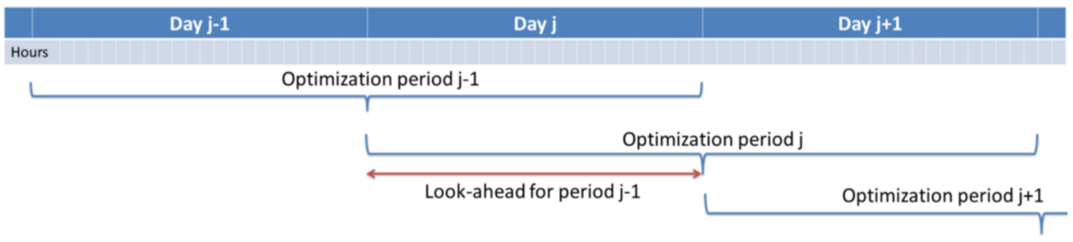


Figure 9: Principle of the rolling horizon optimisation with an horizon and a look-ahead period of one day

5.2 Power plant clustering

For computational efficiency reasons, it is useful to cluster some of the original units into larger units. This reduces the number of continuous and binary variables and can, under some conditions, be performed without significant loss of simulation accuracy.

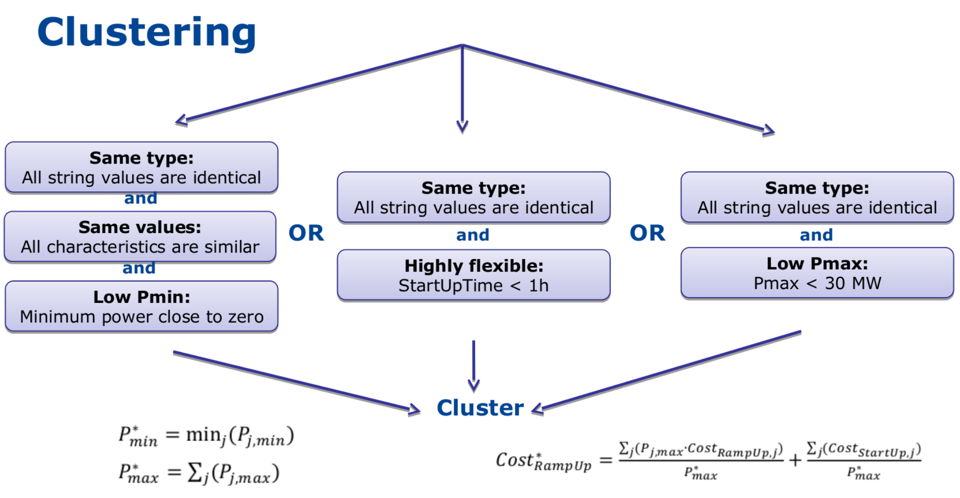
The clustering occurs at the beginning of the pre-processing phase (i.e. the units in the Dispa-SET database do not need to be clustered).

In Dispa-SET, different clustering options are available. Different clusters of power plants can be automatically generated from the same input data. They are described in the two next sections.

5.2.1 MILP clustering

In this formulation, the units that are either very small or very flexible are aggregated into larger units. Some of these units (e.g. the aeroderivative gas turbines) indeed present a low capacity or a high flexibility: their output power does not exceed a few MW and/or they can reach full power in less than 15 minutes (i.e. less than the simulation time step). For these units, a unit commitment model with a time step of 1 hour is unnecessary and computationally inefficient. They are therefore merged into one single, highly flexible unit with averaged characteristics.

The condition for the clustering of two units is a combination of sub-conditions regarding their type, maximum power, flexibility and technical similarities. They are summarised in Figure 10 (NB: the thresholds are for indicative purpose only, they can be user-defined).



Clustering
Same type: All string values are identical
and
Same values: All characteristics are similar
and
Low Pmin: Minimum power close to zero
OR
Same type: All string values are identical
and
Highly flexible: StartUpTime < 1h
OR
Same type: All string values are identical
and
Low Pmax: Pmax < 30 MW
Cluster
$P^*_{min} = \min_j(P_{j,min})$
$P^*_{max} = \Sigma_j(P_{j,max})$
$Cost^*_{RampUp} = \frac{\Sigma_j(P_{j,max} \cdot Cost_{RampUp,j})}{P^*_{max}} + \frac{\Sigma_j(Cost_{StartUp,j})}{P^*_{max}}$

Figure 10: Combination of the conditions for the clustering of power plants

When several units are clustered, the minimum and maximum capacities of the new aggregated unit (indicated by ’*’) are given by:

$$P^*_{min} = min(P_{j,min}) \tag{46}$$

$$P^*_{max} = \sum_j (P_{j,min}) \tag{47}$$

The last equation is also applied for the storage capacity or for the storage charging power.

The unit marginal (or variable cost) is given by weighting over the individuals units' capacities:

$$Cost^*_{Variable} = \frac{\sum_j (P_{j,max} \cdot Cost_{Variable,j})}{P^*_{max}} \tag{48}$$

The start-up/shut-down costs are transformed into ramping costs (the following equation is for start-up, but the same applies to shut-down):

$$Cost^*_{RampUp} = \frac{\sum_j (P_{j,max} \cdot Cost_{RampUp,j})}{P^*_{max}} + \frac{\sum_j (Cost_{StartUp,j})}{P^*_{max}} \tag{49}$$

Other characteristics, such as the plant efficiency, the minimum up/down times or the CO2 emissions are computed as a weighted average:

$$Efficiency^* = \frac{\sum_j (P_{j,max} \cdot Efficiency_j)}{P^*_{max}} \tag{50}$$

It should be noted that only very similar units should be aggregated (i.e. their quantitative characteristics should be similar), in order to avoid errors.

5.2.2 LP clustering

Dispa-SET provides the possibility to generate the optimisation model as an LP problem (i.e. without the binary variables). In that case, the following constraints are removed since they can only be expressed in an MILP formulation:

- Minimum up and down times
- Start-up costs
- Minimum stable load

Since the start-up of individual units is not considered anymore, it is not useful to disaggregate them in the optimisation. All units of a similar technology, fuel and zone can be aggregated into a single unit using the equations proposed in the previous section.

6. Implementation and interface

The typical step-by-step procedure to parametrise and run a Dispa-SET simulation is the following:

1. Fill the Dispa-SET database with properly formatted data (time series, power plant data, etc.)
2. Configure the simulation parameters (rolling horizon, data slicing) in the configuration file.
3. Generate the simulation environment which comprises the inputs of the optimisation
4. Run the optimisation
5. Read and display the simulation results.

This section provides a detailed description of these steps and the corresponding data entities.

6.1 Resolution Flow Chart

The whole resolution process for a dispa-SET run is defined from the processing and formatting of the raw data to the generation of aggregated results, plots, and statistics. A flow chart of the consecutive data entities and processing steps is provided in Figure 11 for the case of the GAMS solver.

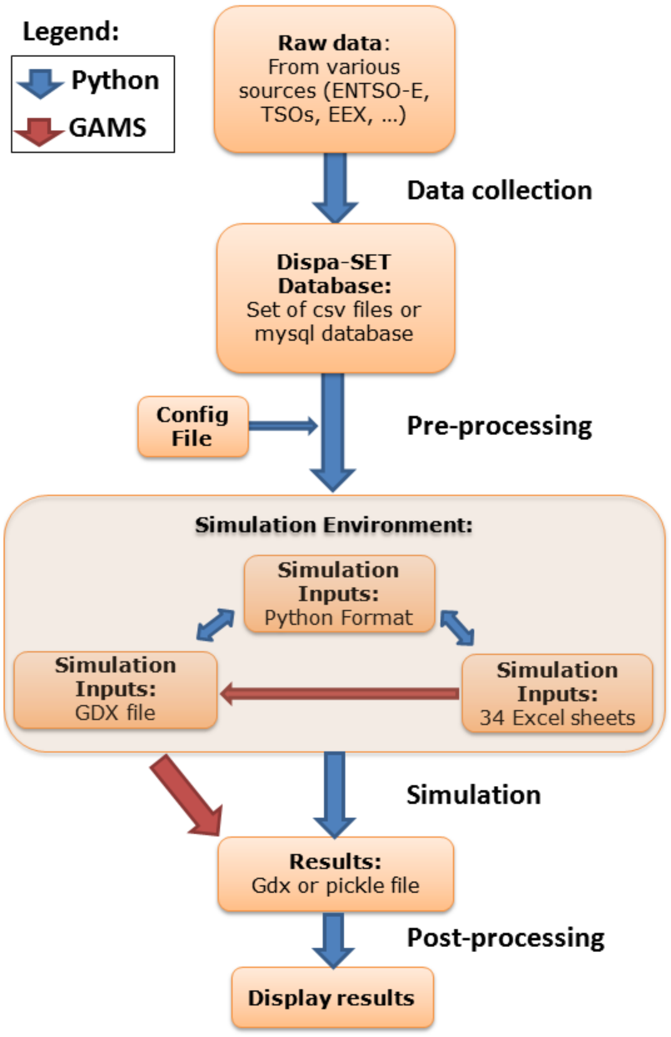

Legend:
Python
GAMS
Raw data: From various sources (ENTSO-E, TSOs, EEX, ...)
Data collection
Dispa-SET Database: Set of csv files or mysql database
Config File
Pre-processing
Simulation Environment:
Simulation Inputs: Python Format
Simulation Inputs: GDX file
Simulation Inputs: 34 Excel sheets
Simulation
Results: Gdx or pickle file
Post-processing
Display results

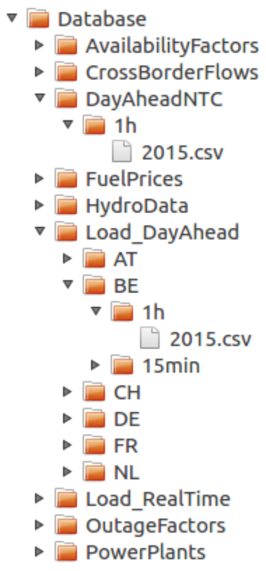
Figure 11: Flow-chart of the Dispa-SET resolution process, from the raw data to the analysis of the results

Each box in the flow chart corresponds to one data entity. The links between these data entities correspond to scripts written in Python or in GAMS. The different steps perform various tasks, which can be summarized by:

1. **Data collection:**
    - Read csv sheets, assemble data
    - Convert to the right format (timestep, units, etc).
    - Define proper time index (duplicates not allowed)
    - Connect to database
    - Check if data present & write data
    - Write metadata
2. **Pre-processing:**
    - Read the config file
    - Slice the data to the required time range
    - Deal with missing data
    - Check data for consistency (min up/down times, start-up times, etc.)
    - Calculate variable costs for each unit
    - Cluster units
    - Define scenario according to user inputs (curtailment, participation to reserve, amount of VRE, amount of storage, etc.)
    - Define initial state (basic merit-order dispatch)
    - Write the simulation environment to a user-defined folder
3. **Simulation environment and interoperability:**
    - Generate Self-consistent folder with all required files to run the simulation:
        - Excel files (xlsx file)
        - GDX file (GAMS input file)
        - Input files in pickle format (a serialized memory representation of all objects)
        - Gams model files
    - Python scripts to translate the data between one format to the other
    - Possibility to modify the inputs manually and re-generate a GDX file from the Excel files
4. **Simulation:**
    - The GAMS simulation file is run from the simulation environment folder
    - Alternatively the model is run with the PYOMO solver
    - All results and inputs are saved within the simulation environment i.e. user defined directory where all the files needed to run the simulation are saved
5. **Post-processing:**
    - Read the simulation results saved in the simulation environment
    - Aggregate the power generation and storage curves
    - Compute yearly statistics
    - Generate plots

6.2 Dispa-SET database

Dispa-SET provides the option for storing all data either in a MySQL database or as a collection of csv files. The publicly available version only comes with the latter, where the input data is stored as csv files in a directory structure. The path to the required data is then provided by the user in the configuration file.



```
▼ Database
    ▶ AvailabilityFactors
    ▶ CrossBorderFlows
    ▼ DayAheadNTC
        ▼ 1h
              2015.csv
    ▶ FuelPrices
    ▶ HydroData
    ▼ Load_DayAhead
        ▶ AT
        ▼ BE
            ▼ 1h
                  2015.csv
            ▶ 15min
        ▶ CH
        ▶ DE
        ▶ FR
        ▶ NL
    ▶ Load_RealTime
    ▶ OutageFactors
    ▶ PowerPlants
```

Figure 12: Partially unfolded view of the database structure

Figure 12 shows a partially unfolded view of the database structure. In that example, data is provided for the day-ahead net transfer capacities for all lines in the EU, for the year 2015 and with a one hour time resolution. Time series are also provided for the day-ahead load forecast for Belgium in 2015 with one hour time resolution.

6.3 Configuration File

The .xlsx config file is read at the beginning of the pre-processing phase. It provides general inputs for the simulation as well as links to the relevant data files in the database (Figure 13).

6.4 Simulation environment

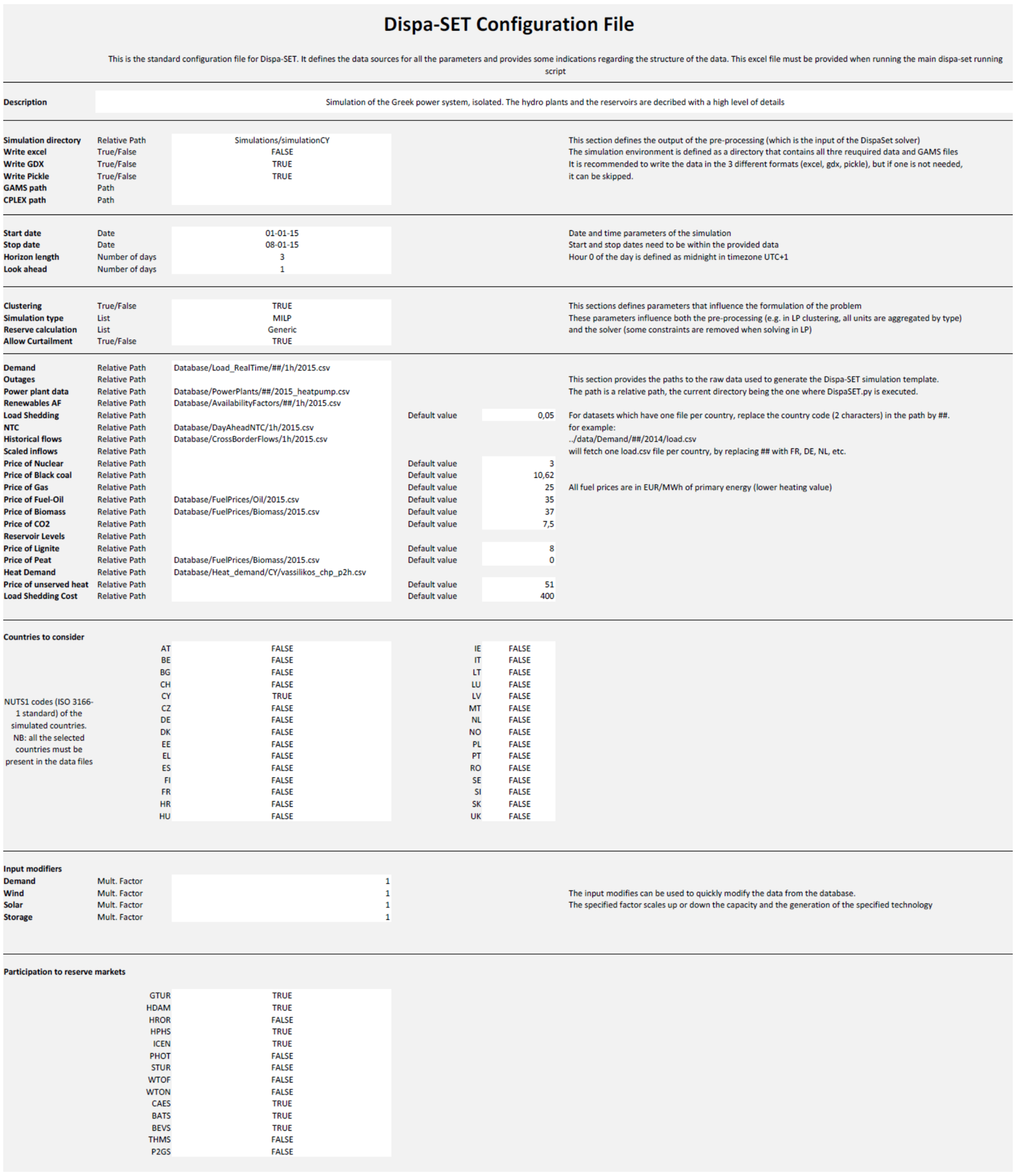
This section describes the different simulation files, templates and scripts required to run the Dispa-SET optimisation model. For each simulation, these files are included into a single directory corresponding to a self-sufficient simulation environment. This simulation environment directory is generated by the pre-processing scripts (Figure 11).

6.4.1 UCM_h.gms and UCM.gpr

UCM_h.gms is the main GAMS model described in Chapter 1. A copy of this file is included in each simulation environment, allowing keeping track of the exact version of the model used for the simulation. The model must be run in GAMS and requires a proper input file (Inputs.gdx).

Requires:	Inputs.gdx	Input file for the simulation.
Generates:	Results.gdx	Simulation results in gdx format
	Results.xlsx	Simulation results in xlsx format.

UCM.gpr is the GAMS project file which should be opened before UCM_h.gms.



Dispa-SET Configuration File

This is the standard configuration file for Dispa-SET. It defines the data sources for all the parameters and provides some indications regarding the structure of the data. This excel file must be provided when running the main dispa-set running script

Description Simulation of the Greek power system, isolated. The hydro plants and the reservoirs are decribed with a high level of details

Simulation directory	Relative Path	Simulations/simulationCY	This section defines the output of the pre-processing (which is the input of the DispaSet solver)
Write excel	True/False	FALSE	The simulation environment is defined as a directory that contains all thre reuquired data and GAMS files
Write GDX	True/False	TRUE	It is recommended to write the data in the 3 different formats (excel, gdx, pickle), but if one is not needed,
Write Pickle	True/False	TRUE	it can be skipped.
GAMS path	Path		
CPLEX path	Path		

Start date	Date	01-01-15	Date and time parameters of the simulation
Stop date	Date	08-01-15	Start and stop dates need to be within the provided data
Horizon length	Number of days	3	Hour 0 of the day is defined as midnight in timezone UTC+1
Look ahead	Number of days	1	

Clustering	True/False	TRUE	This sections defines parameters that influence the formulation of the problem
Simulation type	List	MILP	These parameters influence both the pre-processing (e.g. in LP clustering, all units are aggregated by type)
Reserve calculation	List	Generic	and the solver (some constraints are removed when solving in LP)
Allow Curtailment	True/False	TRUE	

Demand	Relative Path	Database/Load_RealTime/##/1h/2015.csv			
Outages	Relative Path				This section provides the paths to the raw data used to generate the Dispa-SET simulation template.
Power plant data	Relative Path	Database/PowerPlants/##/2015_heatpump.csv			The path is a relative path, the current directory being the one where DispaSET.py is executed.
Renewables AF	Relative Path	Database/AvailabilityFactors/##/1h/2015.csv			
Load Shedding	Relative Path		Default value	0,05	For datasets which have one file per country, replace the country code (2 characters) in the path by ##.
NTC	Relative Path	Database/DayAheadNTC/1h/2015.csv			for example:
Historical flows	Relative Path	Database/CrossBorderFlows/1h/2015.csv			../data/Demand/##/2014/load.csv
Scaled inflows	Relative Path				will fetch one load.csv file per country, by replacing ## with FR, DE, NL, etc.
Price of Nuclear	Relative Path		Default value	3	
Price of Black coal	Relative Path		Default value	10,62	
Price of Gas	Relative Path		Default value	25	All fuel prices are in EUR/MWh of primary energy (lower heating value)
Price of Fuel-Oil	Relative Path	Database/FuelPrices/Oil/2015.csv	Default value	35	
Price of Biomass	Relative Path	Database/FuelPrices/Biomass/2015.csv	Default value	37	
Price of CO2	Relative Path		Default value	7,5	
Reservoir Levels	Relative Path				
Price of Lignite	Relative Path		Default value	8	
Price of Peat	Relative Path	Database/FuelPrices/Biomass/2015.csv	Default value	0	
Heat Demand	Relative Path	Database/Heat_demand/CY/vassilikos_chp_p2h.csv			
Price of unserved heat	Relative Path		Default value	51	
Load Shedding Cost	Relative Path		Default value	400	

Countries to consider

NUTS1 codes (ISO 3166-1 standard) of the simulated countries. NB: all the selected countries must be present in the data files

AT	FALSE	IE	FALSE
BE	FALSE	IT	FALSE
BG	FALSE	LT	FALSE
CH	FALSE	LU	FALSE
CY	TRUE	LV	FALSE
CZ	FALSE	MT	FALSE
DE	FALSE	NL	FALSE
DK	FALSE	NO	FALSE
EE	FALSE	PL	FALSE
EL	FALSE	PT	FALSE
ES	FALSE	RO	FALSE
FI	FALSE	SE	FALSE
FR	FALSE	SI	FALSE
HR	FALSE	SK	FALSE
HU	FALSE	UK	FALSE

Input modifiers

Demand	Mult. Factor	1	
Wind	Mult. Factor	1	The input modifies can be used to quickly modify the data from the database.
Solar	Mult. Factor	1	The specified factor scales up or down the capacity and the generation of the specified technology
Storage	Mult. Factor	1	

Participation to reserve markets

GTUR	TRUE
HDAM	TRUE
HROR	FALSE
HPHS	TRUE
ICEN	TRUE
PHOT	FALSE
STUR	FALSE
WTOF	FALSE
WTON	FALSE
CAES	TRUE
BATS	TRUE
BEVS	TRUE
THMS	FALSE
P2GS	FALSE

Figure 13: View of the Dispa-SET configuration file

6.4.2 make_gdx.gms

GAMS script that reads the different template excel files and generates the Inputs.gdx file.

Requires:	InputDispa-SET - xxx.xlsx	Dispa-SET template files
Generates:	Inputs.gdx	Input file for the simulation

6.4.3 makeGDX.bat (only relevant for Windows and GAMS)

Batch script that generates the input file from the template without requiring opening GAMS. The first time it is executed, the path of the GAMS folder must be provided.

Requires:	InputDispa-SET - xxx.xlsx	Dispa-SET template files
	make_gdx.gms	GAMS file to generate Inputs.gdx
Generates:	Inputs.gdx	Input file for the simulation

6.4.4 writeresults.gms (only relevant for Windows and GAMS)

GAMS script to generate the excel Results.xlsx file from the Results.gdx generated by GAMS (in case the write_excel function was deactivated in GAMS.

Requires:	Results.gdx	Simulation results in gdx format
Generates:	Results.xlsx	Simulation results in xlsx format

6.4.5 Inputs.gdx (only relevant for the GAMS version)

All the inputs of the model must be stored in the Inputs.gdx file since it is the only file read by the main GAMS model. This file is generated from the Dispa-SET template.

Requires:	InputDispa-SET - xxx.xlsx	Dispa-SET template files
Generates:		

6.4.6 InputDispa-SET - Sets.xlsx

Single .xlsx file that contains all the sets used in the model in a column format.

6.4.7 InputDispa-SET - Config.xlsx

Single .xlsx file that contains simulation metadata in the form of a Table (see Fig. 13). This metadata allows setting the rolling horizon parameter and slicing the input data to simulate a subset only.

Table 4: Description of the ”Config” parameter

FirstDay	2012	10	1	First day of the simulation in the template data
LastDay	2013	9	30	Last day of the simulation in the template data
RollingHorizon Length	0	0	3	Length of the rolling horizons
RollingHorizon LookAhead	0	0	1	Overlap period of the rolling horizon

6.4.8 InputDispa-SET - [ParameterName].xlsx

Series of 42 .xlsx files, each corresponding to a parameter of the Dispa-SET model (see Table 2). The files are formatted according to a pre-defined template described hereunder.

The name of each input files is "Input Dispa-SET - [Parameter name].xlsx". These files provide a human readable copy of the data read by the optimisation model.

The structure of all input files follows the following rules:

1. There is one file per model parameter
2. Each file contains only one sheet
3. The first row is left blank for non-time series data (i.e. data starts at cell A2)
4. For time series data, the rows are organized as follows:
    (a) The first row is left blank
    (b) Rows 2 to 5 contains the year, month, day and hour of each data
    (c) Row 6 contains the time index of the data, which will be used in Dispa-SET
    (d) The time series data therefore starts at cell A6
5. If one of the input sets of the data is u (the unit name), it is always defined as the first column of the data (column A)

6. If one of the input sets of the data is h (the time index), it is always defined as the only horizontal input in row 6

In the particular case of the file "Input Dispa-SET - Sets.xlsx" (cfr. section 6.4.6), all the required sets are written in columns with the set name in row 2.

6.5 Post-processing

Post-processing module is provided in the file 'postprocessing.py' which contains a series of functions to read the simulation inputs and results, to plot them, and to derive statistics.

The following statistics are computed:

- The total energy generated by each fuel, in each country.
- The total energy curtailed
- the total load shedding
- The overall country balance of the interconnection flows
- The total hours of congestion in each interconnection line
- The total amount of lost load, indicating (if not null) that the unit commitment problem was infeasible for some hours
- The number of start-ups of power plants for each fuel

The following plots can be generated (cfr. section 3.3 for a description of the post-processing functions):

- A dispatch plot (by fuel type) for each country
- A commitment status (ON/OFF) plot for all the unit in a given country
- The level (or state of charge) of all the storage units in a given country
- The overall power generation by fuel type for all countries (bar plot)

An example usage of these functions is provided in the "Read_Results.ipynb" notebook or in the "Read_Results.py" script.

7. Input Data

In this section, "Input Data" refers to the data stored in the Dispa-SET database. The format of this data is pre-defined and imposed, in such a way that it can be read by the pre-processing tool.

The user should be aware of the following requirements:

- All the time series should be provided in UTC timezone with their timestamps (e.g. '2013-02-20 02:00:00').
- Although the optimisation model is designed to run with any technology or fuel name, the pre-processing and the post-processing tools of Dispa-SET use some hard-coded values. The Dispa-SET database should also comply with this convention (described in the next sections). Any non-recognised technology or fuel will be discarded in the pre-processing.

7.1 Technologies

The Dispa-SET input distinguishes between the technologies defined in Table 5. The VRES column indicates the variable renewable technologies (set "tr" in the optimisation) and the Storage column indicates the technologies which can accumulate energy.

Table 5: Dispa-SET technologies

Technology	Description	VRES	Storage
COMC	Combined cycle	N	N
GTUR	Gas turbine	N	N
HDAM	Conventional hydro dam	N	Y
HROR	Hydro run-of-river	Y	N
HPHS	Pumped hydro storage	N	Y
ICEN	Internal combustion engine	N	N
PHOT	Solar photovoltaic	Y	N
STUR	Steam turbine	N	N
WTOF	Offshore wind turbine	Y	N
WTON	Onshore wind turbine	Y	N
CAES	Compressed air energy storage	N	Y
BATS	Stationary batteries	N	Y
BEVS	Battery-powered electric vehicles	N	Y
THMS	Thermal storage	N	Y
P2GS	Power-to-gas storage	N	Y

7.2 Fuels

Dispa-SET only considers a limited number of fuel types. They are summarised in Table 6, together with some examples.

Table 6: Dispa-SET fuels

Fuel	Examples
BIO	Biomass
GAS	Natural gas
GEO	Geothermal steam
HRD	Hard coal
HYD	Hydrogen
LIG	Lignite
NUC	Nuclear fuel
OIL	Oil products
PEA	Peat
SUN	Solar energy
WAT	Hydropower
WIN	Wind energy
WST	Waste

Different fuels may be used to power a given technology, e.g. steam turbines may be fired with almost any fuel type. In Dispa-SET, each unit must be defined with the pair of values (technology, fuel). Table 7 is derived from a commercial power plant database and indicates the number of occurrences of each combination. It appears clearly that, even through some combinations are irrelevant, both characteristics are needed to define a power plant type.

Table 7: Number of unit for each combination of fuel and technology in EU28

f/t	COMC	GTUR	HDAM	HPHS	HROR	ICEN	PHOT	STUR	WTOF	WTON	Tot
BIO		2				10		79			91
GAS	485	188				28		97			798
GEO								10			10
HRD	4							389			393
HYD		1						1			2
LIG								249			249
NUC								138			138
OIL	7	94				27		146			274
PEA								17			17
SUN							20	7			27
UNK		2				1		1			4
WAT			33	23	21			1			78
WIN									9	27	36
WST		3				7		46			56
Tot	496	290	33	23	21	73	20	1181	9	27	2173

7.3 Unit-specific or technology-specific inputs

Some parameters, such as the availability factor, the outage factor or the inflows may be defined at the unit level or at the technology level. For that reason, the pre-processing tool first looks up the unit name in the database to assign the unit a value, and then lookups the technology if no unit-specific information has been found.

7.4 Demand

Electricity demand (MWh) is given per zone/country and the first row of each column should reflect that name.

Heat demand timeseries is needed where CHP plants are used. In the current formulation, each CHP plant is covering a heat load. In other words, one power plant is connected to a single district heating network. Therefore, in the heat demand input file, the first column has to be a time index and the following columns the heat demand in MW. The first row should contain the exact name of the power plant that will cover this demand.

7.5 Countries

Although the nodes names can be freely user-defined in the database, for the Dispa-SET model, the ISO 3166-1 standard has been adopted to describe each country at the NUTS-1 level. The list of countries is defined in Table 8.

Table 8: NUTS-1 zones defined in Dispa-SET

Code	Country
AT	Austria
BE	Belgium
BG	Bulgaria
CH	Switzerland
CY	Cyprus
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
EL	Greece
ES	Spain
FI	Finland
FR	France
GB	Great Britain
HR	Croatia
HU	Hungary
IE	Ireland
IT	Italy
LT	Lituania
LU	Luxembourg
LV	Latvia
MT	Malta
NL	Netherlands
NO	Norway
PL	Poland
PT	Portugal
RO	Romania
SE	Sweden
SI	Slovenia
SK	Slovakia

It should be noted that ‘UK’ (United Kingdom) has been replaced by ‘GB’ (Great Britain) in this list, i.e. excluding Northern Ireland which is included within the ‘IE’ node.

7.6 Power plant data

The power plant database may contain as many fields as desired, e.g. to ensure that the input data can be traced back, or to provide the id of this plant in another database. However, some fields are required by Dispa-SET and must therefore be defined in the database.

7.6.1 Common fields

The common fields that are required for all units are listed in Table 9:

Table 9: Common fields for all units

Description	Field name	Units
Unit name	Unit	
Commissioning year	Year	
Technology	Technology	
Primary fuel	Fuel	
Zone	Zone	
Capacity	PowerCapacity	MW
Efficiency	Efficiency	%
Efficiency at minimum load	MinEfficiency	%
CO2 intensity	CO2Intensity	TCO2/MWh
Minimum load	PartLoadMin	%
Ramp up rate	RampUpRate	%/min
Ramp down rate	RampDownRate	%/min)
Start-up time	StartUPTime	h
Minimum up time	MinUpTime	h
Minimum down time	MinDownTime	h
No load cost	NoLoadCost	EUR/h
Start-up cost	StartUpCost	EUR
Ramping cost	RampingCost	EUR/MW
Presence of CHP	CHP	y/n

NB: the fields indicated with % as unit must be entered in a non-dimensional way (i.e. 90% should be written 0.9).

7.6.2 Storage units

Some parameters (listed in Table 10) need only to be defined for the units equipped with storage. They can be left blank for all other units.

Table 10: Specific fields for storage units

Description	Field name	Units
Storage capacity	STOCapacity	MWh
Self-discharge rate	STOSelfDischarge	%/h
Maximum charging power	STOMaxChargingPower	MW
Charging efficiency	STOChargingEfficiency	%

In the case of a storage unit, the discharge efficiency should be assigned to the common field "Efficiency". Similarly, the common field "PowerCapacity" is the nominal power in discharge mode.

7.6.3 CHP units

Some parameters (listed in Table 11) need only to be defined for the units equipped with CHP. They can be left blank for all other units.

Table 11: Specific fields for CHP units

Description	Field name	Units
CHP Type	CHPType	extraction/back-pressure/p2h
Power-to-heat ratio	CHPPowerToHeat	
Power Loss factor	CHPPowerLossFactor	
Maximum heat production	CHPMaxHeat	MW(th)
Capacity of heat Storage	STOCapacity	MWh(th)
% of storage heat losses per timestep	STOSelfDischarge	%

In the current version of DispaSet three type of combined heat and power units are supported:

- Extraction/condensing units
- Backpressure units
- Power to heat

For each of the above configurations the following fields must be filled:

Table 12: Mandatory fields per type of CHP unit (X: mandatory, o:optional)

Description	Extraction	Backpressure	Power to heat
CHPType	X	X	X
CHPPowerToHeat	X	X	
CHPPowerLossFactor	X		X
CHPMaxHeat	o	o	X
STOCapacity	o	o	o
STOSelfDischarge	o	o	o

7.7 Renewable generation

Variable renewable generation is defined as power generation from renewable sources that cannot be stored: its is either fed to the grid or curtailed. The technologies falling under this definition are the ones described in the subset "tr" in the model definition.

The time-dependent generation for these technologies must be provided as an exogenous time series in the form of an "availability factor". The latter is defined as the proportion of the nominal power capacity that can be generated at each hour.

In the database, the time series are provided as column vectors with the technology name as header. After the pre-processing, an availability factor is attributed to each unit according to their technology. Non-renewable technologies are assigned an availability factor of 1.

7.8 Storage and hydro data

Storage units are an extension of the regular units, including additional constraints and parameters. In the power plant table, four additional parameters are required: storage capacity (in MWh), self-discharge (in %/h), discharge power (in MW) and discharge efficiency (in %).

Some other parameters must be introduced in the form of time series in the "HydroData" section of the Dispa-SET database. There are described hereunder.

It should be noted that the nomenclature adopted for the modelling of storage units refers to the characteristics of hydro units with water reservoirs. However, these parameters (e.g. inflows, level) can easily be transposed to the case of alternative storage units such as batteries or compressed air energy storage (CAES).

7.8.1 Inflows

The Inflows are defined as the contribution of exogenous sources to the level (or state of charge) or the reservoir. They are expressed in MWh of potential energy. If the inflows are provided as m^3/h, they must be converted.

The input to Dispa-Set is defined as "ScaledInflows". It is the normalized values of the inflow with respect to the nominal power of the storage unit (in discharge mode). As an example, if the inflow value at a certain time is 100 MWh/h and if the turbining capacity of the hydro plant is 200 MW, the scaled inflow value must be defined as 0.5.

Scaled inflows should be provided in the form of time series with the unit name or the technology as columns header.

7.8.2 Storage level

Because emptying the storage has a zero marginal cost, a non-constrained optimisation tends to leave the storage completely empty at the end of the optimisation horizon. For that reason, a minimum storage level is imposed at the last hour of each horizon. In Dispa-SET, a typical optimisation horizon is a few days. The model is therefore not capable of optimising the storage level e.g. for seasonal variations. The minimum storage level at the last hour is therefore an exogenous input. It can be selected from a historical level or obtained from a long-term hydro scheduling optimisation.

The level input in the Dispa-SET database is normalized with respect to the storage capacity: its minimum value is zero and its maximum is one.

7.8.3 Variable capacity storage

In special cases, it might be necessary to simulate a storage unit whose capacity varies in time. A typical example is the simulation of the storage capacity provided by electric vehicles: depending on the time of the day, the connected battery capacity varies.

This special case can be simulated using the "AvailabilityFactor" input. In the case of a storage unit, it reduces the available capacity by a factor varying from 0 to 1.

7.9 Power plant outages

In the current version, Dispa-SET does not distinguish planned outages from unplanned outages. They are characterized for each unit by the "OutageFactor" parameter. This parameter varies from 0 (no outage) to 1 (full outage). The available unit power is thus given by its nominal capacity multiplied by (1-OutageFactor).

The outages are provided in the dedicated section of the Database for each unit. They consist of a time series with the unit name as columns header.

7.10 Interconnections

Two case should be distinguished when considering interconnections:

- Interconnections occurring between the simulated zones
- Interconnections occurring between the simulated zones and the Rest of the World (RoW)

These two cases are addresses by two different datasets described hereunder.

7.10.1 Net transfer capacities

Dispa-SET endogenously models the internal exchanges between countries (or zones) using a commercial net transfer capacity (NTC). It does not consider (yet) DC power flows or more complex grid simulations.

Since the NTC values might vary in time, they must be supplied as time series, whose header include the origin country, the string ' -> ' and the destination country. As an example, the NTC from Belgium to France must be provided with the header 'BE -> FR'.

Because NTCs are not necessarily symmetrical, they must be provided in both directions (i.e. 'BE -> FR' and 'FR -> BE'). Non-provided NTCs are considered to be zero (i.e. no interconnection).

7.10.2 Historical physical flows

In Dispa-SET, the flows between internal zones and the rest of the world cannot be modelled endogenously. They must be provided as exogenous inputs. These inputs are referred to as "Historical physical flows", although they can also be user-defined.

In the input table of historical flows, the headers are similar to those of the NTCs (ie. 'XX -> YY'). All flows occurring between an internal zone of the simulation and an outside zone are considered as external flows and summed up. As an example, the historical flows 'FR -> XX', 'FR -> YY' and 'FR -> ZZ' will be aggregated in to a single interconnection flow 'FR -> RoW' if XX, YY and ZZ are not simulated zones.

These aggregated historical flows are then imposed to the solver as exogenous inputs.

In Dispa-SET, the flows are defined as positive variables. For each zone, there will thus be a maximum of two vectors defining its exchanges with the rest of the world (e.g. 'FR -> RoW' and 'RoW -> FR').

As for the NTCs, undefined historical flows are considered to be zero, i.e. not providing any historical flows is equivalent to consider the system as isolated.

7.11 Fuel Prices

Fuel prices vary both geographically and in time. They must therefore be provided as a time series for each simulated zone. One table is provided per fuel type, with as column header the zone to which it applies. If no header is provided, the fuel price is applied to all the simulated zones.

8. Conclusions

This document describes the formulation and implementation of Dispa-SET 2.2, a model developed by the DG Joint Research Centre of the European Commission, in close collaboration with University of Líege.

The main new features include an improved user interface, many bug fixes and the inclusion of a novel heat module. This new module allows simulating CHP power plants, district heating, heat pumps, or electrical boiler. In addition, the possibility to include thermal storage allows quantifying the potential flexibility in operation of these systems.

The model is released as an open-source tool[2] and is provided with an open dataset for testing purposes. It can therefore be freely re-used or modified to fit the needs of a particular case study.

The development of Dispa-SET is an ongoing process. New features will be added in the future, including for example:

- The addition of new constraints (e.g. hydropower and water requirements for cooling).
- An innovative clustered formulation for accelerating the solution time of large scale problems (e.g Europe-wide models)
- A better representation of reserve needs, distinguishing between different types of reserves (secondary and tertiary).
- The addition of a capacity planning module.
- A better representation of grid constraints through the implementation of a DC power flow formulation instead of the NTC approach currently in use.

[2]The source code of the public version of Dispa-SET is available at: `http://www.dispaset.eu`

References

[Arroyo and Conejo, 2000] Arroyo, J. M. and Conejo, A. J., 'Optimal response of a thermal unit to an electricity spot market', *IEEE Transactions on power systems*, Vol. 15, No 3, 2000, pp. 1098–1104.

[Beltramo et al., 2017] Beltramo, A., Julea, A., Refa, N., Drossinos, Y., Thiel, C. and Quoilin, S., 'Using electric vehicles as flexible resource in power systems: A case study in the Netherlands', In '2017 14th International Conference on the European Energy Market (EEM)', pp. 1–6. .

[Carrión and Arroyo, 2006] Carrión, M. and Arroyo, J. M., 'A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem', *IEEE Transactions on power systems*, Vol. 21, No 3, 2006, pp. 1371–1378. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1664974.

[Deane et al., 2012] Deane, J. P., Chiodi, A., Gargiulo, M. and Ó Gallachóir, B. P., 'Soft-linking of a power systems model to an energy systems model', *Energy*, Vol. 42, No 1, Jun. 2012, pp. 303–312. ISSN 0360-5442. . URL http://www.sciencedirect.com/science/article/pii/S0360544212002551.

[Fernandez B. C. et al., 2017] Fernandez B. C., R., CARERI, F., KAVVADIAS, K., Hidalgo Gonzalez, I., Zucker, A. and PETEVES, E., 'Systematic mapping of power system models: Expert survey', JRC Technical Report, EU Commission, 2017. DOI: 10.2760/422399.

[Hidalgo González et al., 2014] Hidalgo González, I., Quoilin, S. and Zucker, A., 'Dispa-SET 2.0: unit commitment and power dispatch model', Tech. rep., Publications Office of the European Union, 2014.

[Hidalgo González et al., 2015] Hidalgo González, I., Ruez Castello, P., Scobbi, A., NIJS, W., Quoilin, S., Zucker, A. and Thiel, C., 'Addressing flexibility in energy system models', Tech. rep., Publications Office of the European Union, 2015.

[Morales-España et al., 2013] Morales-España, G., Latorre, J. M. and Ramos, A., 'Tight and compact MILP formulation for the thermal unit commitment problem', *IEEE Transactions on Power Systems*, Vol. 28, No 4, 2013, pp. 4897–4908. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6485014.

[Pavičević et al., 2017] Pavičević, M., Tomić, I., Quoilin, S., Zucker, A. and Pukšec, T., 'Applying the Dispa-SET model on the Western Balkans power systems', In 'Proceedings of the 2017 SDEWES Conference', .

[Pfenninger et al., 2017] Pfenninger, S., DeCarolis, J., Hirth, L., Quoilin, S. and Staffell, I., 'The importance of open data and software: Is energy research lagging behind?', *Energy Policy*, Vol. 101, Feb. 2017, pp. 211–215. ISSN 0301-4215. . URL http://www.sciencedirect.com/science/article/pii/S0301421516306516.

[Pina et al., 2013] Pina, A., Silva, C. A. and Ferrão, P., 'High-resolution modeling framework for planning electricity systems with high penetration of renewables', *Applied Energy*, Vol. 112, Dec. 2013, pp. 215–223. ISSN 03062619. . URL http://linkinghub.elsevier.com/retrieve/pii/S030626191300487X.

[Poncelet et al., 2016] Poncelet, K., Delarue, E., Six, D., Duerinck, J. and D'haeseleer, W., 'Impact of the level of temporal and operational detail in energy-system planning models', *Applied Energy*, Vol. 162, Jan. 2016, pp. 631–643. ISSN 0306-2619. . URL http://www.sciencedirect.com/science/article/pii/S0306261915013276.

[Quoilin et al., 2014] Quoilin, S., Gonzalez Vazquez, I., Zucker, A. and Thiel, C., 'Available technical flexibility for balancing variable renewable energy sources: case study in Belgium', In 'Proceedings of the 9th Conference on Sustainable Development of Energy, Water and Environment Systems', URL http://orbi.ulg.be/handle/2268/172402.

[Quoilin et al., 2017a] Quoilin, S., Hidalgo Gonzalez, I. and Zucker, A., 'Modelling Future EU Power Systems Under High Shares of Renewables: The Dispa-SET 2.1 open-source model', JRC Technical Report, EU Commission, 2017a.

[Quoilin et al., 2015] Quoilin, S., Nijs, W., Gonzalez, I. H., Zucker, A. and Thiel, C., ‘Evaluation of simplified flexibility evaluation tools using a unit commitment model’, In ‘2015 12th International Conference on the European Energy Market (EEM)’, pp. 1–5. .

[Quoilin et al., 2017b] Quoilin, S., Nijs, W. and Zucker, A., ‘Evaluating flexibility and adequacy in future EU power systems: model coupling and long-term forecasting’, In ‘Proceedings of the 2017 ECOS Conference’, San Diego.

[Sánchez Pérez, 2017] Sánchez Pérez, A., *Modelling Hydropower in detail to assess its contribution to flexibility services in the European power system*. Master Thesis, University of Utrecht, Netherlands, 2017.

[Simoes et al., 2013] Simoes, S., Nijs, W., Ruiz, P., Sgobbi, A. R., D Bolat, P., Thiel, C. and Peteves, S., ‘The JRC-EU-TIMES model–Assessing the long-term role of the SET Plan Energy technologies’, Tech. rep., Publications Office of the European Union, 2013.

[Tomić et al., 2017] Tomić, I., Pavičević, M., Quoilin, S., Zucker, A., Krajačić, G., Pukšec, T. and Duić, N., ‘Applying the Dispa-SET model on the seven countries from the South East Europe’, In ‘8th Energy Planning and Modeling of Energy Systems-Meeting Belgrade’, .

9. Appendix: API Documentation

9.1 Subpackages

9.1.1 DispaSET.misc package

9.1.1.1 Submodules

9.1.1.2 DispaSET.misc.colorstreamhandler module

`DispaSET.misc.colorstreamhandler.`**`ColorStreamHandler`**
alias of `_AnsiColorStreamHandler`

9.1.1.3 DispaSET.misc.gdx_handler module

Collection of functions to write Dispa-SET input data to a gdx file and/or to a simulation directory with one excel file per parameter.

Example: read gdx file:

```
data = GdxToList(gams_dir,'Results.gdx',varname='all',verbose=True)
```

write it to a dictionary of dataframes:

```
dataframes = GdxToDataframe(data,fixindex=True,verbose=True)
```

@author: Sylvain Quoilin (sylvain.quoilin@ec.europa.eu)

`DispaSET.misc.gdx_handler.`**`gdx_to_dataframe`**`(`*data*, *fixindex=False*, *verbose=False*`)`
This function structures the raw data extracted from a gdx file (using the function GdxToList) and outputs it as a dictionary of pandas dataframes (or series)

Parameters

- **`data`** – Dictionary with all the collected values (within lists), from GdxToList function
- **`fixindex`** – This flag allows converting string index into integers and sort the data

Returns dictionary of dataframes

`DispaSET.misc.gdx_handler.`**`gdx_to_list`**`(`*gams_dir*, *filename*, *varname='all'*, *verbose=False*`)`
This function loads the gdx with the results of the simulation All results are stored in an unordered list

Parameters

- **`gams_dir`** – Gams working directory
- **`filename`** – Path to the gdx file to be read
- **`varname`** – In case online one variable is needed, specify it name (otherwise specify "all")

Returns Dictionary with all the collected values (within lists)

`DispaSET.misc.gdx_handler.`**`get_gams_path`**`()`
Function that attempts to search for the GAMS installation path (required to write the GDX or run gams)

It returns the path if it has been found, or an empty string otherwise.

Currently works for Windows, Linux and OSX. More searching rules and patterns should be added in the future

DispaSET.misc.gdx_handler.**get_gdx**(*gams_dir*, *resultfile*)
Short wrapper of the two gdx reading functions (GdxToDataframe and GdxToList)

Parameters

- **gams_dir** – Gams working directory
- **resultfile** – Path to the gdx file to be read

Returns dictionary of dataframes

DispaSET.misc.gdx_handler.**import_local_lib**(*lib*)
Try to import the GAMS api and gdxcc to write gdx files

DispaSET.misc.gdx_handler.**package_exists**(*package*)

DispaSET.misc.gdx_handler.**write_variables**(*gams_dir*, *gdx_out*, *list_vars*)
This function performs the following: * Use the gdxcc library to create a gdxHandle instance * Check that the gams path is well defined * Call the "insert_symbols" function to write all sets and parameters to gdxHandle

Parameters

- **gams_dir** – (Relative) path to the gams directory
- **gdx_out** – (Relative) path to the gdx file to be written
- **list_vars** – List with the sets and parameters to be written

9.1.1.4 DispaSET.misc.str_handler module

DispaSET.misc.str_handler.**clean_strings**(*x*, *exclude_digits=False*, *exclude_punctuation=False*)
Function to convert strange unicode and remove characters punctuation

Parameters x – any string or list of strings

Usage:

```
df['DisplayName'].apply(clean_strings)
```

DispaSET.misc.str_handler.**shrink_to_64**(*x*, *N=64*)
Function that reduces the length of the keys to be written to 64 (max admissible length for GAMS)

Parameters

- **x** – String or list of strings
- **N** – Integer with the maximum string length (if different from 64)

Returns Shrinked string or list of strings

9.1.1.5 Module contents

9.1.2 DispaSET.postprocessing package

9.1.2.1 Submodules

9.1.2.2 DispaSET.postprocessing.postprocessing module

Set of functions useful to analyse to DispaSET output data.

@author: Sylvain Quoilin, JRC

DispaSET.postprocessing.postprocessing.**GAMSstatus**(*statustype*, *num*)
Function that returns the model status or the solve status from gams

Parameters

- **statustype** – String with the type of status to retrieve ("solver" or "model")
- **num** – Indicated termination condition (Integer)

Returns String with the status

`DispaSET.postprocessing.postprocessing.`**`aggregate_by_fuel`**(*PowerOutput*, *Inputs*, *SpecifyFuels=None*)

This function sorts the power generation curves of the different units by technology

Parameters

- **PowerOutput** – Dataframe of power generationwith units as columns and time as index
- **Inputs** – Dispaset inputs version 2.1.1
- **SpecifyFuels** – If not all fuels should be considered, list containing the relevant ones

Returns PowerByFuel Dataframe with power generation by fuel

`DispaSET.postprocessing.postprocessing.`**`ds_to_df`**(*inputs*)

Function that converts the dispaset data format into a dictionary of dataframes

Parameters inputs – input file

Returns dictionary of dataframes

`DispaSET.postprocessing.postprocessing.`**`filter_by_country`**(*PowerOutput*, *inputs*, *c*)

This function filters the dispaset Output Power dataframe by country

Parameters

- **PowerOutput** – Dataframe of power generationwith units as columns and time as index
- **Inputs** – Dispaset inputs version 2.1.1
- **c** – Selected country (e.g. "BE")

Returns Power Dataframe with power generation by country

`DispaSET.postprocessing.postprocessing.`**`get_imports`**(*flows*, *c*)

Function that computes the balance of the imports/exports of a given zone

Parameters

- **flows** – Pandas dataframe with the timeseries of the exchanges
- **c** – Country (zone) to consider

Returns NetImports Scalar with the net balance over the whole time period

`DispaSET.postprocessing.postprocessing.`**`get_indicators_powerplant`**(*inputs*, *results*)

Function that analyses the dispa-set results at the power plant level Computes the number of startups, the capacity factor, etc

Parameters

- **inputs** – DispaSET inputs
- **results** – DispaSET results

Returns out Dataframe with the main power plants characteristics and the computed indicators

`DispaSET.postprocessing.postprocessing.`**`get_load_data`**(*inputs*, *c*)

Get the load curve, the residual load curve, and the net residual load curve of a specific country

Parameters

- **inputs** – DispaSET inputs (output of the get_sim_results function)
- **c** – Country to consider (e.g. ”BE”)

Return out Dataframe with the following columns: Load: Load curve of the specified country ResidualLoad: Load minus the production of variable renewable sources NetResidualLoad: Residual netted from the interconnections with neightbouring countries

`DispaSET.postprocessing.postprocessing.`**`get_plot_data`**(*inputs*, *results*, *c*)

Function that reads the results dataframe of a DispaSET simulation and extract the dispatch data spedific to one country

Parameters

- **results** – Pandas dataframe with the results (output of the GdxToDataframe function)
- **c** – Country to be considered (e.g. ”BE”)

Returns plotdata Dataframe with the dispatch data storage and outflows are negative

`DispaSET.postprocessing.postprocessing.`**`get_result_analysis`**(*inputs*, *results*)

Reads the DispaSET results and provides useful general information to stdout

Parameters

- **inputs** – DispaSET inputs
- **results** – DispaSET results

`DispaSET.postprocessing.postprocessing.`**`get_sim_results`**(*path='.'*, *cache=False*, *temp_path='.pickle'*)

This function reads the simulation environment folder once it has been solved and loads the input variables together with the results.

Parameters

- **path** – Relative path to the simulation environment folder (current path by default)
- **cache** – If true, caches the simulation results in a pickle file for faster loading the next time
- **temp_path** – Temporary path to store the cache file

Returns inputs,results Two dictionaries with all the input and outputs

`DispaSET.postprocessing.postprocessing.`**`plot_country`**(*inputs*, *results*, *c=''*, *rng=None*)

Generates plots from the dispa-SET results for one specific country

Parameters

- **inputs** – DispaSET inputs
- **results** – DispaSET results
- **c** – Considered country (e.g. ”BE”)

`DispaSET.postprocessing.postprocessing.`**`plot_country_capacities`**(*inputs*, *plot=True*)

Plots the installed capacity for each country, disaggregated by fuel type

:param inputs : Dictionnary with the inputs of the model (output of the function GetResults)

DispaSET.postprocessing.postprocessing.**plot_dispatch**(*demand*, *plotdata*, *level=None*, *curtailment=None*, *rng=None*)

Function that plots the dispatch data and the reservoir level as a cumulative sum

Parameters

- **demand** – Pandas Series with the demand curve
- **plotdata** – Pandas Dataframe with the data to be plotted. Negative columns should be at the beginning. Output of the function GetPlotData
- **level** – Optional pandas series with an aggregated reservoir level for the considered zone.
- **curtailment** – Optional pandas series with the value of the curtailment
- **rng** – Indexes of the values to be plotted. If undefined, the first week is plotted

DispaSET.postprocessing.postprocessing.**plot_dispatch_safe**(*demand*, *plotdata*, *level=None*, *curtailment=None*, *rng=None*)

Function that plots the dispatch data and the reservoir level as a cumulative sum. In this case, the Pandas index is not used since it can cause a bug in matplotlib

Parameters

- **demand** – Pandas Series with the demand curve
- **plotdata** – Pandas Dataframe with the data to be plotted. Negative columns should be at the beginning. Output of the function GetPlotData
- **level** – Optional pandas series with an aggregated reservoir level for the considered zone.
- **curtailment** – Optional pandas series with the value of the curtailment
- **rng** – Indexes of the values to be plotted. If undefined, the first week is plotted

DispaSET.postprocessing.postprocessing.**plot_energy_country_fuel**(*inputs*, *results*, *PPindicators*)

Plots the generation for each country, disaggregated by fuel type

Parameters

- **results** – Dictionnary with the outputs of the model (output of the function GetResults)
- **PPindicators** – Por powerplant statistics (output of the function get_indicators_powerpla

DispaSET.postprocessing.postprocessing.**plot_rug**(*df_series*, *on_off=False*, *cmap='Greys'*)

Create multiaxis rug plot from pandas Dataframe

Parameters: df_series: 2D pandas with timed index

on_off:

- If True all points that are above 0 will be plotted as one color.

- If False all values will be colored based on their value.

cmap: palette name (from colorbrewer, matplotlib etc.)

Returns: plot

Function written by K. Kavvadias

9.1.2.3 Module contents

9.1.3 DispaSET.preprocessing package

9.1.3.1 Submodules

9.1.3.2 DispaSET.preprocessing.data_check module

This files gathers different functions used in the DispaSET to check the input data

__author__ = ”Sylvain Quoilin (sylvain.quoilin@ec.europa.eu)”

`DispaSET.preprocessing.data_check.`**`check_AvailabilityFactors`**(*plants*, *AF*)
Function that checks the validity of the provided availability factors and warns if a default value of 100% is used.

`DispaSET.preprocessing.data_check.`**`check_MinMaxFlows`**(*df_min*, *df_max*)
Function that checks that there is no incompatibility between the minimum and maximum flows

`DispaSET.preprocessing.data_check.`**`check_chp`**(*config*, *plants*)
Function that checks the CHP plant characteristics

`DispaSET.preprocessing.data_check.`**`check_clustering`**(*plants*, *plants_merged*)
Function that checks that the installed capacities are still equal after the clustering process

Parameters

- **`plants`** – Non-clustered list of units
- **`plants_merged`** – clustered list of units

`DispaSET.preprocessing.data_check.`**`check_df`**(*df*, *StartDate=None*, *StopDate=None*, *name=”*)
Function that check the time series provided as inputs

`DispaSET.preprocessing.data_check.`**`check_heat_demand`**(*plants*, *data*)
Function that checks the validity of the heat demand profiles

Parameters **`plants`** – List of CHP plants

`DispaSET.preprocessing.data_check.`**`check_simulation_environment`**(*SimulationPath*, *store_type='pickle'*, *firstline=7*)
Function to test the validity of disapset inputs :param SimulationPath: Path to the simulation folder :param store_type: choose between: ”list”, ”excel”, ”pickle” :param firstline: Number of the first line in the data (only if type==”excel”)

`DispaSET.preprocessing.data_check.`**`check_sto`**(*config*, *plants*, *raw_data=True*)
Function that checks the storage plant characteristics

`DispaSET.preprocessing.data_check.`**`check_units`**(*config*, *plants*)
Function that checks the power plant characteristics

`DispaSET.preprocessing.data_check.`**`isStorage`**(*tech*)
Function that returns true the technology is a storage technology

`DispaSET.preprocessing.data_check.`**`isVRE`**(*tech*)
Function that returns true the technology is a variable renewable energy technology

9.1.3.3 DispaSET.preprocessing.data_handler module

DispaSET.preprocessing.data_handler.**NodeBasedTable**(*path*, *idx*, *countries*, *tablename=''*, *default=None*)

This function loads the tabular data stored in csv files relative to each zone (a.k.a node, country) of the simulation.

Parameters

- **path** – Path to the data to be loaded
- **idx** – Pandas datetime index to be used for the output
- **countries** – List with the country codes to be considered
- **fallback** – List with the order of data source.
- **tablename** – String with the name of the table being processed
- **default** – Default value to be applied if no data is found

Returns Dataframe with the time series for each unit

DispaSET.preprocessing.data_handler.**UnitBasedTable**(*plants, path, idx, countries, fallbacks=['Unit'], tablename='', default=None, RestrictWarning=None*)

This function loads the tabular data stored in csv files and assigns the proper values to each unit of the plants dataframe. If the unit-specific value is not found in the data, the script can fallback on more generic data (e.g. fuel-based, technology-based, zone-based) or to the default value. The order in which the data should be loaded is specified in the fallback list. For example, ["Unit","Technology"] means that the script will first try to find a perfect match for the unit name in the data table. If not found, a column with the unit technology as header is search. If not found, the default value is assigned.

Parameters

- **plants** – Dataframe with the units for which data is required
- **path** – Path to the data to be loaded
- **idx** – Pandas datetime index to be used for the output
- **countries** – List with the country codes to be considered
- **fallback** – List with the order of data source.
- **tablename** – String with the name of the table being processed
- **default** – Default value to be applied if no data is found
- **RestrictWarning** – Only display the warnings if the unit belongs to the list of technologies provided in this parameter

Returns Dataframe with the time series for each unit

DispaSET.preprocessing.data_handler.**define_parameter**(*sets_in*, *sets*, *value=0*)

Function to define a DispaSET parameter and fill it with a constant value

Parameters

- **sets_in** – List with the labels of the sets corresponding to the parameter
- **sets** – dictionary containing the definition of all the sets (must comprise those referenced in sets_in)
- **value** – Default value to attribute to the parameter

`DispaSET.preprocessing.data_handler.`**`invert_dic_df`**(*dic*, *tablename=''*)

Function that takes as input a dictionary of dataframes, and inverts the key of the dictionary with the columns headers of the dataframes

Parameters

- **`dic`** – dictionary of dataframes, with the same columns headers and the same index
- **`tablename`** – string with the name of the table being processed (for the error msg)

Returns dictionary of dataframes, with swapped headers

`DispaSET.preprocessing.data_handler.`**`load_config_excel`**(*ConfigFile*)

Function that loads the DispaSET excel config file and returns a dictionary with the values

Parameters **`ConfigFile`** – String with (relative) path to the DispaSET excel configuration file

`DispaSET.preprocessing.data_handler.`**`load_config_yaml`**(*filename*)

Loads YAML file to dictionary

`DispaSET.preprocessing.data_handler.`**`load_csv`**(*filename*, *TempPath='.pickle'*, *header=0*, *skiprows=[]*, *skip_footer=0*, *index_col=None*, *parse_dates=False*)

Function that loads an xls sheet into a dataframe and saves a temporary pickle version of it. If the pickle is newer than the sheet, do no load the sheet again.

Parameters

- **`file_excel`** – path to the excel file
- **`TempPath`** – path to store the temporary data files

`DispaSET.preprocessing.data_handler.`**`merge_series`**(*plants*, *data*, *mapping*, *method='WeightedAverage'*, *tablename=''*)

Function that merges the times series corresponding to the merged units (e.g. outages, inflows, etc.)

Parameters

- **`plants`** – Pandas dataframe with the information relative to the original units
- **`data`** – Pandas dataframe with the time series and the original unit names as column header
- **`mapping`** – Mapping between the merged units and the original units. Output of the clustering function
- **`method`** – Select the merging method ("WeightedAverage"/"Sum")
- **`tablename`** – Name of the table being processed (e.g. "Outages"), used in the warnings

Return merged Pandas dataframe with the merged time series when necessary

`DispaSET.preprocessing.data_handler.`**`write_to_excel`**(*xls_out*, *list_vars*)

Function that reads all the variables (in list_vars) and inserts them one by one to excel

Parameters

- **`xls_out`** – The path of the folder where the excel files are to be written
- **`list_vars`** – List containing the dispaset variables

Returns Binary variable (True)

9.1.3.4 DispaSET.preprocessing.preprocessing module

This is the main file of the DispaSET pre-processing tool. It comprises a single function that generated the DispaSET simulation environment.

@author: S. Quoilin

`DispaSET.preprocessing.preprocessing.`**`adjust_capacity`**(*inputs*, *tech_fuel*, *scaling=1*, *value=None*, *singleunit=False*, *write_gdx=False*, *dest_path=''*)

Function used to modify the installed capacities in the Dispa-SET generated input data The function update the Inputs.p file in the simulation directory at each call

Parameters

- **inputs** – Input data dictionnary OR path to the simulation directory containing Inputs.p
- **tech_fuel** – tuple with the technology and fuel type for which the capacity should be modified
- **scaling** – Scaling factor to be applied to the installed capacity
- **value** – Absolute value of the desired capacity (! Applied only if scaling != 1 !)
- **singleunit** – Set to true if the technology should remain lumped in a single unit
- **write_gdx** – boolean defining if Inputs.gdx should be also overwritten with the new data
- **dest_path** – Simulation environment path to write the new input data. If unspecified, no data is written!

Returns New SimData dictionnary

`DispaSET.preprocessing.preprocessing.`**`adjust_storage`**(*inputs*, *tech_fuel*, *scaling=1*, *value=None*, *write_gdx=False*, *dest_path=''*)

Function used to modify the storage capacities in the Dispa-SET generated input data The function update the Inputs.p file in the simulation directory at each call

Parameters

- **inputs** – Input data dictionnary OR path to the simulation directory containing Inputs.p
- **tech_fuel** – tuple with the technology and fuel type for which the capacity should be modified
- **scaling** – Scaling factor to be applied to the installed capacity
- **value** – Absolute value of the desired capacity (! Applied only if scaling != 1 !)
- **write_gdx** – boolean defining if Inputs.gdx should be also overwritten with the new data
- **dest_path** – Simulation environment path to write the new input data. If unspecified, no data is written!

Returns New SimData dictionnary

`DispaSET.preprocessing.preprocessing.`**`build_simulation`**(*config*, *plot_load=False*)

This function reads the DispaSET config, loads the specified data, processes it when needed, and formats it in the proper DispaSET format. The output of the function is a directory with all inputs and simulation files required to run a DispaSET simulation

Parameters

- **config** – Dictionary with all the configuration fields loaded from the excel file. Output of the ”LoadConfig” function.
- **plot_load** – Boolean used to display a plot of the demand curves in the different zones

`DispaSET.preprocessing.preprocessing.`**`get_git_revision_tag`**`()`

Get version of DispaSET used for this run. tag + commit hash

9.1.3.5 DispaSET.preprocessing.utils module

This file gathers different functions used in the DispaSET pre-processing tools

@author: Sylvain Quoilin (sylvain.quoilin@ec.europa.eu)

`DispaSET.preprocessing.utils.`**`clustering`**`(`*plants*, *method='Standard'*, *Nslices=20*, *PartLoadMax=0.1*, *Pmax=30*`)`

Merge excessively disaggregated power Units.

Parameters

- **plants** – Pandas dataframe with each power plant and their characteristics (following the DispaSET format)
- **method** – Select clustering method (”Standard”/âĂŹLPâĂŹ/None)
- **Nslices** – Number of slices used to fingerprint each power plant characteristics. slices in the power plant data to categorize them (fewer slices involves that the plants will be aggregated more easily)
- **PartLoadMax** – Maximum part-load capability for the unit to be clustered
- **Pmax** – Maximum power for the unit to be clustered

Returns A list with the merged plants and the mapping between the original and merged units

`DispaSET.preprocessing.utils.`**`incidence_matrix`**`(`*sets*, *set_used*, *parameters*, *param_used*`)`

This function generates the incidence matrix of the lines within the nodes A particular case is considered for the node ”Rest Of the World”, which is no explicitely defined in DispaSET

`DispaSET.preprocessing.utils.`**`interconnections`**`(`*Simulation_list*, *NTC_inter*, *Historical_flows*`)`

Function that checks for the possible interconnections of the countries included in the simulation. If the interconnections occurs between two of the countries defined by the user to perform the simulation with, it extracts the NTC between those two countries. If the interconnection occurs between one of the countries selected by the user and one country outside the simulation, it extracts the physical flows; it does so for each pair (country inside-country outside) and sums them together creating the interconnection of this country with the RoW.

Parameters

- **Simulation_list** – List of simulated countries
- **NTC** – Day-ahead net transfer capacities (pd dataframe)
- **Historical_flows** – Historical flows (pd dataframe)

9.1.3.6 Module contents

9.1.4 DispaSET.pyomo package

9.1.4.1 Submodules

9.1.4.2 DispaSET.pyomo.model module

9.1.4.3 DispaSET.pyomo.utils module

`DispaSET.pyomo.utils.`**`get_set_members`**`(`*instance*, *sets*`)`
Get set members relative to a list of sets

Parameters

- **`instance`** – Pyomo Instance
- **`sets`** – List of strings with the set names

Returns A list with the set members

`DispaSET.pyomo.utils.`**`get_sets`**`(`*instance*, *varname*`)`
Get sets that belong to a pyomo Variable or Param

Parameters

- **`instance`** – Pyomo Instance
- **`varname`** – Name of the Pyomo Variable (string)

Returns A list with the sets that belong to this Param

`DispaSET.pyomo.utils.`**`pyomo_format`**`(`*sets*, *param*`)`
Function that flattens the multidimensional dispaset input data into the pyomo format: a dictionnary with a tuple and the parameter value. The tuple contains the strings of the corresponding set values

`DispaSET.pyomo.utils.`**`pyomo_to_pandas`**`(`*instance*, *varname*`)`
Function converting a pyomo variable or parameter into a pandas dataframe. The variable must have one or two dimensions and the sets must be provided as a list of lists

Parameters

- **`instance`** – Pyomo Instance
- **`varname`** – Name of the Pyomo Variable (string)

9.1.4.4 Module contents

9.2 Submodules

9.3 DispaSET.solve module

This worksheet contains the two main functions to solve the DispaSET optimisation problem using PYOMO or GAMS.

@author: ”Sylvain Quoilin”

`DispaSET.solve.`**`is_sim_folder_ok`**`(`*sim_folder*`)`

`DispaSET.solve.`**`solve_GAMS`**`(`*sim_folder*, *gams_folder=None*, *work_dir=None*, *output_lst=False*`)`

`DispaSET.solve.`**`solve_pyomo`**`(`*sim_folder*`)`

KJ-NA-29085-EN-N

JRC Mission

As the science and knowledge service of the European Commission, the Joint Research Centre's mission is to support EU policies with independent evidence throughout the whole policy cycle.

doi:10.2760/860626

ISBN 978-92-79-77866-7