



European
Commission

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



This publication is a Technical report by the Joint Research Centre (JRC), the European Commission's science and knowledge service. It aims to provide evidence-based scientific support to the European policymaking process. The scientific output expressed does not imply a policy position of the European Commission. Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use that might be made of this publication.

Contact Information

Name: Sylvain Quoilin
Address: European Commission, Westerduinweg 3, P.O. Box 2, 1755 ZG Petten, Netherlands
E-mail: sylvain.quoilin@ec.europa.eu
Tel.: +31 224 56 5305

JRC Science Hub

<https://ec.europa.eu/jrc>

JRC105452

EUR 28427 EN

PDF ISBN 978-92-79-65265-3 ISSN 1831-9424 doi:10.2760/25400

Luxembourg: Publications Office of the European Union, 2017

© European Union, 2017

The reuse of the document is authorised, provided the source is acknowledged and the original meaning or message of the texts are not distorted. The European Commission shall not be held liable for any consequences stemming from the reuse.

How to cite: Quoilin S., Hidalgo Gonzalez I., Zucker A., Modelling Future EU Power Systems Under High Shares of Renewables - The Dispa-SET 2.1 open-source model, EUR 28427 EN, doi:10.2760/25400

All images © European Union 2017, except: [Front page, Unknown author, Copenhagen - Denmark - Wind Farm], 2009. Source: [commons.wikimedia.org]

Modelling Future EU Power Systems Under High Shares of Renewables

Abstract

The increase in the share of renewable sources leads to new needs in terms of flexible resources. This evolution of the power system requires proper simulation tools. The Dispa-SET model simulates with a high level of detail the short-term operation of large-scale power systems, solving the unit commitment and optimal dispatch problems. 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.

Modelling Future EU Power Systems Under High Shares of Renewables

Sylvain QUOILIN

Ignacio HIDALGO GONZALEZ

Andreas ZUCKER

Contents

1. Introduction	4
2. Model Description	4
2.1. Variables	5
2.1.1. Sets	5
2.1.2. Parameters	5
2.1.3. Optimization Variables	5
2.2. Optimisation model	5
2.2.1. Objective function	7
2.2.2. Demand-related constraints	9
2.2.3. Power output bounds	10
2.2.4. Minimum up and down times	11
2.2.5. Storage-related constraints	13
2.2.6. Emission limits	14
2.2.7. Network-related constraints	14
2.2.8. Curtailment	14
2.2.9. Load shedding	14
2.3. Rolling Horizon	14
2.4. Power plant clustering	15
2.4.1. MILP clustering	15
2.4.2. LP clustering	16
3. Implementation and interface	17
3.1. Resolution Flow Chart	17
3.2. Dispa-SET database	19
3.3. Configuration File	19
3.4. Simulation environment	19
3.4.1. UCM_h.gms and UCM.gpr	19
3.4.2. make_gdx.gms	20
3.4.3. makeGDX.bat	20
3.4.4. writeresults.gms	21
3.4.5. Inputs.gdx	21
3.4.6. InputDispa-SET - Sets.xlsx	21
3.4.7. InputDispa-SET - Config.xlsx	21
3.4.8. InputDispa-SET - [ParameterName].xlsx	21
3.5. Post-processing	22
4. Input Data	23
4.1. Technologies	23
4.2. Fuels	23
4.3. Unit-specific or technology-specific inputs	25
4.4. Countries	25

4.5. Power plant data.....	25
4.5.1. Common fields	26
4.5.2. Storage units	26
4.5.3. CHP units	26
4.6. Renewable generation	27
4.7. Storage and hydro data	27
4.7.1. Inflows	27
4.7.2. Storage level.....	27
4.7.3. Variable capacity storage	27
4.8. Power plant outages.....	28
4.9. Interconnections.....	28
4.9.1. Net transfer capacities.....	28
4.9.2. Historical physical flows	28
4.10 Fuel Prices	29
5. Getting Started	30
5.1. Prerequisites	30
5.1.1. Using Dispa-SET with GAMS:	30
5.1.2. Using Dispa-SET with PYOMO:	30
5.2. Run the pre-processing tool.....	31
5.2.1. Check the configuration file	31
5.2.2. Pre-processing	31
5.2.3. Check the simulation environment	31
5.3. Run the optimisation	32
5.3.1. Using the GAMS user interface:	32
5.3.2. Using the GAMS command line:	32
5.3.3. From Dispa-SET, using GAMS:	32
5.3.4. From Dispa-SET, using PYOMO:	32
5.4. Postprocessing	32
6. Conclusions	35
References	36
Appendix 1: Dispa-SET api	37
7. Appendix: API Documentation.....	38
7.1. DispaTools	38
7.2. DispaCheck	41
7.3. IO - data	41
7.4. Input generation.....	42
7.5. Post Processing.....	42
7.6. DispaSolve.....	45

1. Introduction

Variability and uncertainty are familiar aspects of all power systems. However, the increase in the share of renewable sources leads to new needs in terms of flexible resources [Hidalgo González et al., 2015]. These resources can be provided by different means, including:

- Dispatchable power plants (i.e. with ramp up and ramp down capabilities)
- Storage systems, mainly in the form of pumped hydro units
- Grid interconnections between countries
- Demand side management
- Power-to-X solutions (gas, fuel, heat, ...)

The aim of the Dispa-SET model [Hidalgo González et al., 2014, Quoilin et al., 2014] is to represent with a high level of detail the short-term operation of large-scale power systems, solving the unit commitment problem. To that aim, it is considered 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 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.

The model is released as an open-source tool to increase its transparency, its reproducibility and its visibility [Pfenninger et al., 2017]. It is also cross-platform, and has been successfully tested in Windows, Linux and OSX. The model is structured in such a way that potential users can easily modify the input data to run their own simulations with limited knowledge in programming languages.

2. Model Description

The model is expressed as an optimisation problem written in both GAMS and PYOMO. Continuous variables include the individual unit dispatched power, the shed 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:

- Minimum and maximum power for each unit
- Power plant ramping limits
- Reserves up and down
- Minimum up/down times
- Load Shedding

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

Name	Description
f	Fuel types
h	Hours
i	Time step in the current optimization 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	Storage units (including hydro reservoirs)

- 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
- Yearly schedules for the levels of large water reservoirs

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

2.1 Variables

2.1.1 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

2.1.2 Parameters

The parameters correspond to the exogenous data provided to the model. They result in the constant coefficients of the optimisation model. The list of Dispa-SET parameters is provided in Table 2.

2.1.3 Optimization Variables

The optimisation (or decision) variables are the variables that need to be adjusted to minimise the objective function. They are defined in Table 3.

2.2 Optimisation model

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.

Table 2: Dispa-SET Parameters

Name	Units	Description
AvailabilityFactor(u,i)	%	Percentage of nominal capacity available
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
MaxRamp2U(u,h)	MW/h	Maximum 15-min Ramp-up capability
MaxRamp2D(u,h)	MW/h	Maximum 15-min Ramp-down capability
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 optimization 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

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.

2.2.1 Objective function

The goal of the unit commitment problem is to minimize 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.

$$\begin{aligned}
\min \sum_{u,n,i} & \left[\begin{aligned}
& CostStartUp_{p_{u,i}} + CostShutDown_{u,i} + CostFixed_u \cdot Committed_{u,i} \\
& + CostVariable_{u,i} \cdot Power_{u,i} + CostRampUp_{p_{u,i}} + CostRampDown_{u,i} \\
& + PriceTransmission_{i,l} \cdot Flow_{i,l} + (CostLoadShedding_{i,n} \cdot ShedLoad_{i,n}) \\
& + VOLL_{Power} \cdot (LostLoadMaxPower_{i,n} + LostLoadMinPower_{i,n}) \\
& + VOLL_{Reserve} \cdot (LostLoadReserve2U_{i,n} + LostLoadReserve2D_{i,n}) \\
& + VOLL_{Ramp} \cdot (LostLoadRampUp_{p_{u,i}} + LostLoadRampDown_{u,i}) \end{aligned} \right] \quad (1)
\end{aligned}$$

The costs can be broken down as:

- Fixed costs: depending on whether the unit is on or off.
- Variable costs: stemming from the power output of the units.
- 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.
- Loss of load: power exceeding the demand or not matching it, ramping and reserve.

The variable production costs (in EUR/MWh), are determined by fuel and emission prices corrected by the 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:

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

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

The start-up and shut-down costs are positive variables, active when the commitment status between two consecutive time periods is modified:

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

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:

$$\begin{aligned}
& i = 1 : \\
& \quad CostRampUp_{u,i} \geq CostRampUp_u \cdot (Power_{u,i} - PowerInitial_u) \\
& \quad CostRampDown_{u,i} \geq CostRampDown_u \cdot (PowerInitial_u - Power_{u,i}) \\
& \quad i > 1 : \\
& \quad \quad CostRampUp_{u,i} \geq CostRampUp_u \cdot (Power_{u,i} - Power_{u,i-1}) \\
& \quad \quad CostRampDown_{u,i} \geq CostRampDown_u \cdot (Power_{u,i-1} - Power_{u,i})
\end{aligned} \tag{4}$$

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 capacity the system is not able to provide when the minimum/maximum power, reserve, or 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.

2.2.2 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.

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

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).

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

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

$$\begin{aligned}
& \text{If RampStartUpMaximum}_u \geq \text{PowerMinStable}_u \cdot 4 \\
& \text{Then FlexibilityUp}_u = \text{RampStartUpMaximum}_u \\
& \text{Else FlexibilityUp}_u = 0
\end{aligned} \tag{7}$$

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:

$$\begin{aligned}
\text{MaxRamp2U}_{u,i} \leq & (\text{PowerCapacity}_u \cdot \text{AvailabilityFactor}_{u,i} \\
& \cdot (1 - \text{OutageFactor}_{u,i}) - \text{Power}_{u,i}) \cdot 4
\end{aligned} \tag{8}$$

The same applies to the 15 min ramping down capabilities:

$$\begin{aligned}
& \text{MaxRamp2D}_{u,i} \\
\leq & \max(\text{RampDownMaximum}_u, \text{FlexibilityDown}_u) \cdot \text{Committed}_{u,i}
\end{aligned} \tag{9}$$

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:

$$\begin{aligned}
& \text{If RampShutDownMaximum}_u < \text{PowerMinStable}_u \cdot 4 : \\
& \text{MaxRamp2D}_{u,i} \leq (\text{Power}_{u,i} - \text{PowerMinStable}_u \cdot \text{Committed}_{u,i}) \cdot 4 \\
& \text{Else :} \\
& \text{MaxRamp2D}_{u,i} \leq \text{Power}_{u,i} \cdot 4
\end{aligned} \tag{10}$$

The reserve requirements are defined 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:

$$\text{Demand2U}_{n,i} = \sqrt{10 \cdot \max_h (\text{Demand}_{DA,n,h}) + 150^2} - 150 \tag{11}$$

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

$$\text{Demand2D}_{n,h} = 0.5 \cdot \text{Demand2U}_{n,h} \tag{12}$$

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

$$\begin{aligned}
& \text{Demand2U}_{n,h} \\
\leq & \sum_{u,t} (\text{MaxRamp2U}_{u,i} \cdot \text{Technology}_{u,t} \cdot \text{Reserve}_t \cdot \text{Location}_{u,n}) \\
& + \text{LostLoadReserve2UH}_{n,i}
\end{aligned} \tag{13}$$

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

2.2.3 Power output bounds

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

$$\begin{aligned}
& \text{PowerMustRun}_{u,i} \cdot \text{Committed}_{u,i} \\
& \leq \text{Power}_{u,i}
\end{aligned} \tag{14}$$

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

$$\begin{aligned}
& \text{Power}_{u,i} \\
\leq & \text{PowerCapacity}_u \cdot \text{AvailabilityFactor}_{u,i} \\
& \cdot (1 - \text{OutageFactor}_{u,i}) \cdot \text{Committed}_{u,i}
\end{aligned} \tag{15}$$

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:

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

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:

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

2.2.4 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. In order to avoid excessive ageing of the generators, or because of their physical characteristics, 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 these constraints linearly, it is necessary to keep track of the number of hours the unit must be online at the beginning of the simulation:

$$TimeUpLeftInitial_u = \min \{N, (TimeUpMinimum_u - TimeUpInitial_u) \cdot CommittedInitial_u\} \tag{19}$$

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:

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

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.

$$TimeUpLeftJustStarted_{u,i} = \min \{N - i + 1, TimeUpMinimum_u\} \quad (21)$$

The equation imposing the unit to remain committed is written:

$$\begin{aligned} & i = 1 : \\ & \sum_{ii=i}^{i+TimeUpLeftJustStarted_{u,i}-1} Committed_{u,ii} \geq \\ & TimeUpLeftJustStarted_{u,i} \cdot (Committed_{u,i} - CommittedInitial_u) \\ & i > 1 : \\ & \sum_{ii=i}^{i+TimeUpLeftJustStarted_{u,i}-1} Committed_{u,ii} \geq \\ & TimeUpLeftJustStarted_{u,i} \cdot (Committed_{u,i} - Committed_{u,i-1}) \end{aligned} \quad (22)$$

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

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

Related to the initial status of the unit:

$$TimeDownLeft_u \sum_{i=1} Committed_{u,i} = 0 \quad (24)$$

The $TimeDownLeftJustStopped$ parameter is computed by:

$$TimeDownLeftJustStopped_{u,i} = \min \{N - i + 1, TimeDownMinimum_u\} \quad (25)$$

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

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

This formulation avoids the use of additional binary variables to describe the start-up and shut-down of each unit.

2.2.5 Storage-related constraints

Generation units with energy storage capabilities (mostly large hydro reservoirs and pumped hydro storage units) must meet additional restrictions related to the amount of energy stored. 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). 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:

$$StorageMinimum_s \leq StorageLevel_{s,i} \quad (27)$$

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:

$$StorageLevel_{s,i} \leq StorageCapacity_s \cdot AvailabilityFactor_{s,i} \quad (28)$$

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).

$$StorageInput_{s,i} \leq StorageChargingCapacity_s \cdot AvailabilityFactor_{s,i} \cdot (1 - Committed_{s,i}) \quad (29)$$

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

$$\begin{aligned} \frac{Power_{i,s}}{StorageDischargeEfficiency_s} + StorageOutflow_{s,i} \\ + Spillage_{s,i} - StorageInflow_{s,i} \\ \leq StorageLevel_{s,i} \end{aligned} \quad (30)$$

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

$$\begin{aligned} StorageInput_{s,i} \cdot StorageChargingEfficiency_s \\ - StorageOutflow_{s,i} - Spillage_{s,i} \\ + StorageInflow_{s,i} \\ \leq StorageCapacity_s - StorageLevel_{s,i} \end{aligned} \quad (31)$$

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

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

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 as possible. An exogenous minimum profile is thus provided and the following constraint is applied:

$$\begin{aligned} \text{StorageLevel}_{s,N} &\geq \min(\text{StorageProfile}_{s,N} \\ &\cdot \text{AvailabilityFactor}_{s,N} \cdot \text{StorageCapacity}_s, \\ &\text{StorageLevel}_{s,0} + \sum_{i=1}^N \text{InFlows}_{s,i}) \end{aligned} \quad (33)$$

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.

2.2.6 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:

$$\begin{aligned} \sum_u (\text{Power}_{u,i} \cdot \text{EmissionRate}_{u,p} \cdot \text{Location}_{u,n}) \\ \leq \text{EmissionMaximum}_{n,p} \end{aligned} \quad (34)$$

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.

2.2.7 Network-related constraints

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

$$\begin{aligned} \text{FlowMinimum}_{l,i} &\leq \text{Flow}_{l,i} \\ \text{Flow}_{l,i} &\leq \text{FlowMaximum}_{l,i} \end{aligned} \quad (35)$$

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.

2.2.8 Curtailment

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

$$\begin{aligned} \text{CurtailedPower}_{n,i} \\ \leq \sum_{u,tr} (\text{Power}_{u,i} \cdot \text{Technology}_{u,tr} \cdot \text{Location}_{u,n}) \cdot \text{Curtailment}_n \end{aligned} \quad (36)$$

2.2.9 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)

$$\text{ShedLoad}_{n,i} \leq \text{LoadShedding}_n \quad (37)$$

2.3 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 optimization problems that are run recursively throughout the year.

Figure 1 shows an example of such approach, in which the optimization horizon is one day, with a look-ahead (or overlap) period of one day. The initial values of the optimization for day j are the final values of the optimization of the previous day. The look-ahead period is modelled to avoid issues related to the end of the optimization period such as emptying the hydro reservoirs, or starting low-cost but non-flexible power plants. In this case, the optimization is performed over 48 hours, but only the first 24 hours are conserved.

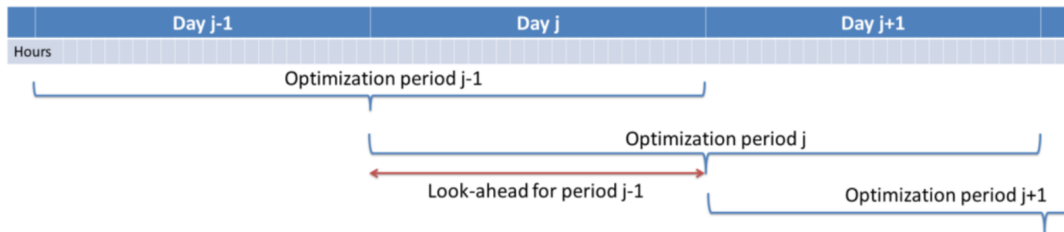


Figure 1: Principle of the rolling horizon optimisation with an horizon and a look-ahead period of one day

Although the previous example corresponds to an optimization 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 optimization horizon plus the overlap period should be at 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 optimization horizons, while larger systems benefit from a reduced time horizon, the minimum being one day.

2.4 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, in 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 and can be automatically generated from the same input data. They are described in the two next sections.

2.4.1 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 turbojets) 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 summarized in Figure 2 (NB: the thresholds are for indicative purpose only, they can be user-defined).

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

$$P_{min}^* = \min(P_{j,min}) \quad (38)$$

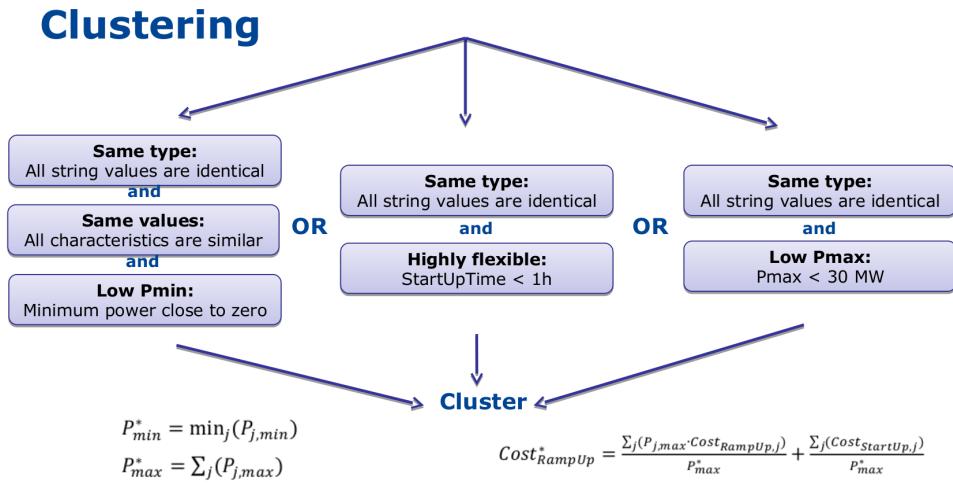


Figure 2: Combination of the conditions for the clustering of power plants

$$P_{max}^* = \sum_j(P_{j,min}) \quad (39)$$

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:

$$Cost_{Variable}^* = \frac{\sum_j(P_{j,max} \cdot Cost_{Variable,j})}{P_{max}^*} \quad (40)$$

The start-up/shut-down costs are transformed into ramping costs (example with ramp-up):

$$Cost_{RampUp}^* = \frac{\sum_j(P_{j,max} \cdot Cost_{RampUp,j})}{P_{max}^*} + \frac{\sum_j(Cost_{StartUp,j})}{P_{max}^*} \quad (41)$$

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

$$Efficiency^* = \frac{\sum_j(P_{j,max} \cdot Efficiency_j)}{P_{max}^*} \quad (42)$$

It should be noted that only very similar units are aggregated (i.e. their quantitative characteristics should be similar), which avoids errors due to excessive aggregation.

2.4.2 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.

3. Implementation and interface

The typical step-by-step procedure to parametrize 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. Open the GAMS simulation files (project: UCM.gpr and model: UCM_h.gms) and run the model.
5. Read and display the simulation results.

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

3.1 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 3.

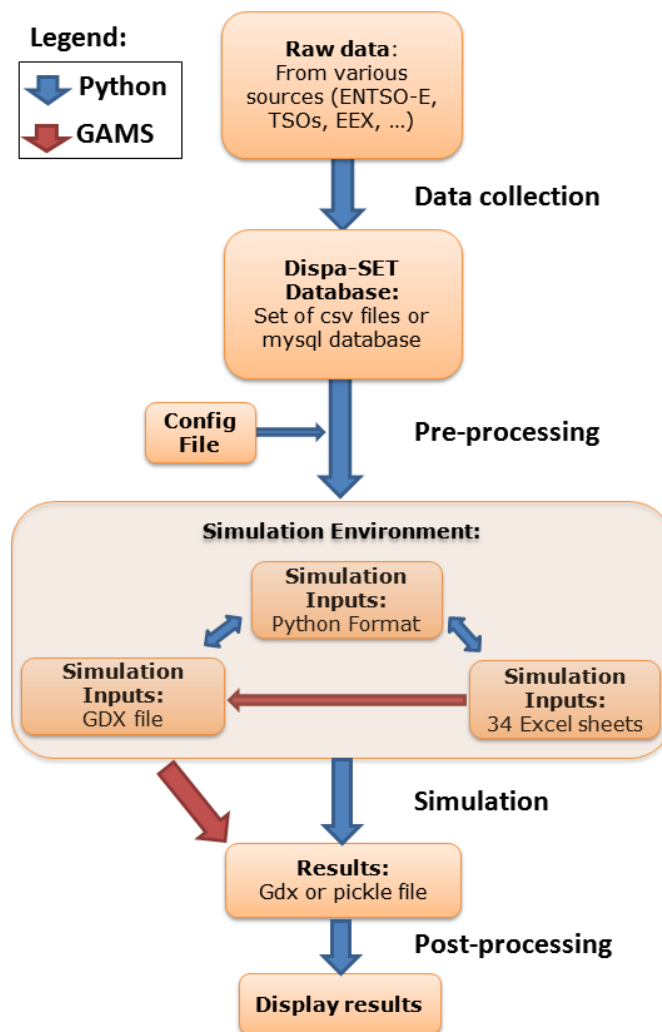


Figure 3: Flow-chart of the Dispa-SET resolution process, from the raw data to the analysis of the results

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 cost 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:

• **Self-consistent folder with all required files to run the simulation:**

- Excel files
- GDX file (GAMS input file)
- Input files in pickle format
- 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

5. Post-processing:

- Read the simulation results saved in the simulation environment
- Aggregate the power generation and storage curves
- Compute yearly statistics
- Generate plots

3.2 Dispa-SET database

Although two versions of the database are available (mysql and csv), the public version of Dispa-SET only comes with the latter. The Dispa-SET input data is stored as csv files in a directory structure. A link to the required data is then provided by the user in the configuration file.

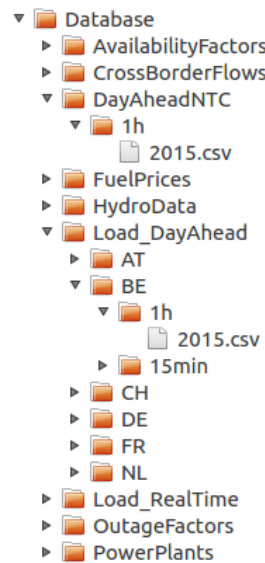


Figure 4: Partially unfolded view of the database structure

Figure 4 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.

3.3 Configuration File

The excel 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 5).

3.4 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 3).

3.4.1 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	Standard simulation for 6 countries, with the MILP formulation			
Simulation director	Relative Path	Simulations\simulation_test	This section defines the output of the pre-processing (which is the input of the DispaSet solver). The simulation environment is defined as a directory that contains all the required data and GAMS files. It is recommended to write the data in the 3 different formats (excel, gdx, pickle), but if one is not needed, it can be skipped.	
Write excel	True/False	FALSE		
Write GDX	True/False	TRUE		
Write Pickle	True/False	TRUE		
GAMS path	Path			
Start date	Date	1/1/2015	Date and time parameters of the simulation. Start and stop dates need to be within the provided data. Hour 0 of the day is defined as midnight in timezone UTC+1	
Stop date	Date	12/31/2015		
Horizon length	Number of days	3		
Look ahead	Number of days	1		
Clustering	True/False	TRUE	This section defines parameters that influence the formulation of the problem. These parameters influence both the pre-processing (e.g. in LP clustering, all units are aggregated by type) and the solver (some constraints are removed when solving in LP)	
Simulation type	List	MILP		
Reserve calculation	List	Generic		
Allow Curtailment	True/False	TRUE		
Demand	Relative Path	Database\Load_RealTime\##	This section provides the paths to the raw data used to generate the Dispa-SET simulation template. The path is a relative path, the current directory being the one where DispaSET.py is executed. For datasets which have one file per country, replace the country code (2 characters) in the path by ##. For example: <code>..data\Demand\##\2014\load.csv</code> will fetch one load.csv file per country, by replacing ## with FR, DE, NL, etc. All fuel prices are in EUR/MWh of primary energy (lower heating value)	
Outages	Relative Path			
Power plant data	Relative Path	Database\PowerPlants\##\Zi		
Renewables AF	Relative Path	Database\Availability\Factors		
Load Shedding	Relative Path			Default value: 0.05
NTC	Relative Path	Database\CrossAhead\NTC\FH		
Historical flows	Relative Path	Database\CrossBorderFlows		
Scaled inflows	Relative Path	Database\HydroData\Scaled		
Price of Nuclear	Relative Path			Default value: 3
Price of Black coal	Relative Path	Database\FuelPrices\Coal\Zi		Default value: 11
Price of Gas	Relative Path	Database\FuelPrices\Gas\Zi		Default value: 20
Price of Fuel-Oil	Relative Path	Database\FuelPrices\Oil\Zi		Default value: 35
Price of Biomass	Relative Path	Database\FuelPrices\Biomass		Default value: 37
Price of CDZ	Relative Path			Default value: 7
Reservoir Levels	Relative Path	Database\HydroData\Reserv		
Countries to consider				
	AT	TRUE	IE	FALSE
	BE	TRUE	IT	FALSE
	BG	FALSE	LT	FALSE
	CH	TRUE	LU	FALSE
	CY	FALSE	LV	FALSE
	CZ	FALSE	MT	FALSE
	DE	TRUE	NL	TRUE
	DK	FALSE	NO	FALSE
	EE	FALSE	PL	FALSE
	EL	FALSE	PT	FALSE
	ES	FALSE	RO	FALSE
	FI	FALSE	SE	FALSE
	FR	TRUE	SI	FALSE
	HR	FALSE	SK	FALSE
	HU	FALSE	UK	FALSE
Input modifiers				
Demand	Multi. Factor	1	The input modifiers can be used to quickly modify the data from the database. The specified factor scales up or down the capacity and the generation of the specified technology	
Wind	Multi. Factor	1		
Solar	Multi. Factor	1		
Storage	Multi. Factor	1		
Participation to reserve markets				
	COMC	TRUE		
	GTUR	TRUE		
	HDAM	TRUE		
	HROR	FALSE		
	HPSH	TRUE		
	ICEN	TRUE		
	PHOT	FALSE		
	STUR	FALSE		
	VTOF	FALSE		
	VTON	FALSE		
	CAES	TRUE		
	BATS	TRUE		
	BEVS	TRUE		
	THMS	FALSE		
	P2GS	FALSE		

Figure 5: View of the Dispa-SET configuration file

3.4.2 make_gdx.gms

GAMS file that reads the different template excel files and generates the Inputs.gdx file. This file should be opened in GAMS.

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

3.4.3 makeGDX.bat

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

3.4.4 writeresults.gms

GAMS file 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

3.4.5 Inputs.gdx

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:

3.4.6 InputDispa-SET - Sets.xlsx

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

3.4.7 InputDispa-SET - Config.xlsx

Single excel file that contains simulation metadata in the form of a Table. 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

3.4.8 InputDispa-SET - [ParameterName].xlsx

Series of 42 excel 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 the input files are "Input Dispa-SET - [Parameter name].xlsx". These files contain the data to be read by the model, after conversion into a GDX file.

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 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 data therefore starts at 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 3.4.6), all the required sets are written in columns with the set name in row 2.

3.5 Post-processing

Post-processing is implemented in the form of a series of functions to read the simulation inputs and results, to plot them, and to derive statistics.

The following values 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:

- 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.

4. 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.

Two important preliminary comments should be formulated:

- All the time series should be registered with their timestamps (e.g. '2013-02-20 02:00:00') relative to the UTC time zone.
- 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-recognized technology or fuel will be discarded in the pre-processing.

4.1 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
HPSH	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

4.2 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	Bagasse, Biodiesel, Gas From Biomass, Gasification, Biomass, Briquettes, Cattle Residues, Rice Hulls Or Padi Husk, Straw, Wood Gas (From Wood Gasification), Wood Waste Liquids Excl Blk Liq (Incl Red Liquor, Sludge, Wood, Spent Sulfite Liquor And Oth Liquids, Wood And Wood Waste
GAS	Blast Furnace Gas, Boiler Natural Gas, Butane, Coal Bed Methane, Coke Oven Gas, Flare Gas, Gas (Generic), Methane, Mine Gas, Natural Gas, Propane, Refinery Gas, Sour Gas, Synthetic Natural Gas, Top Gas, Voc Gas & Vapor, Waste Gas, Wellhead Gas
GEO	Geothermal steam
HRD	Anthracite, Other Anthracite, Bituminous Coal, Coker By-Product, Coal Gas (From Coal Gasification), Coke, Coal (Generic), Coal-Oil Mixture, Other Coal, Coal And Pet Coke Mi, Coal Tar Oil, Anthracite Coal Waste, Coal-Water Mixture, Gob, Hard Coal / Anthracite, Imported Coal, Other Solids, Soft Coal, Anthracite Silt, Steam Coal, Subbituminous, Pelletized Synthetic Fuel From Coal, Bituminous Coal Waste)
HYD	Hydrogen
LIG	Lignite black, Lignite brown, lignite
NUC	U, Pu
OIL	Crude Oil, Distillate Oil, Diesel Fuel, No. 1 Fuel Oil, No. 2 Fuel Oil, No. 3 Fuel Oil, No. 4 Fuel Oil, No. 5 Fuel Oil, No. 6 Fuel Oil, Furnace Fuel, Gas Oil, Gasoline, Heavy Oil Mixture, Jet Fuel, Kerosene, Light Fuel Oil, Liquefied Propane Gas, Methanol, Naphtha, Gas From Fuel Oil Gasification, Fuel Oil, Other Liquid, Orimulsion, Petroleum Coke, Petroleum Coke Synthetic Gas, Black Liquor, Residual Oils, Re-Refined Motor Oil, Oil Shale, Tar, Topped Crude Oil, Waste Oil
PEA	Peat Moss
SUN	Solar energy
WAT	Hydro energy
WIN	Wind energy
WST	Digester Gas (Sewage Sludge Gas), Gas From Refuse Gasification, Hazardous Waste, Industrial Waste, Landfill Gas, Poultry Litter, Manure, Medical Waste, Refused Derived Fuel, Refuse, Waste Paper And Waste Plastic, Refinery Waste, Tires, Agricultural Waste, Waste Coal, Waste Water Sludge, 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

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

4.3 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 lookups 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.

4.4 Countries

Although the nodes names can be freely user-defined in the database, for the Dispa-SET EU 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	Lithuania
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. northern Ireland is not considered and is instead included within the 'IE' node.

4.5 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.

4.5.1 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).

4.5.2 Storage units

Some parameters (listed in Table 10) must only 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.

4.5.3 CHP units

Some parameters (listed in Table 11) must only 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/other
Power-to-heat ratio	CHPPowerToHeat	

NB: CHP units are not (yet) supported in version 2.1.0 of Dispa-SET

4.6 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.

4.7 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).

4.7.1 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 turbinning 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.

4.7.2 Storage level

Because emptying the storage has a zero marginal cost, a non-constrained optimization 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 optimization.

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.

4.7.3 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.

4.8 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 $(1 - \text{OutageFactor})$.

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.

4.9 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.

4.9.1 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).

4.9.2 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.

4.10 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.

5. Getting Started

This short tutorial describes the main steps to get a practical example of Dispa-SET running. It covers both the installation of the required software and libraries, and the use of the Dispa-SET tools and internal functions. This section is provided as an example, the detailed description of all Dispa-SET functions is provided in the Appendix.

5.1 Prerequisites

Install Python 2.7, with full scientific stack. The Anaconda 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
- matplotlib
- pickle

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

```
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 Files\Common Files\MicrosoftVisual C++ for Python\9.0)

5.1.1 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.

The GAMS API for python has been pre-compiled in the "Externals" folder and is usable with most operating systems (both 32 and 64 bits). 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\win32\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. If it is not successful, the user is prompted for the proper installation path.

5.1.2 Using Dispa-SET with PYOMO:

Dispa-SET can also be run using PYOMO, which avoids the necessity of a GAMS license. The following steps should be followed:

- Install pyomo

```
pip install pyomo
```

- Install a solver and add it to the PATH environment variable (e.g. if cplex is installed, the "cplex" command should be callable from any command prompt).

5.2 Run the pre-processing tool

5.2.1 Check the configuration file

Dispa-SET runs are defined in dedicated excel configuration files stored in the "ConfigFiles" folder. The configuration file "ConfigTest.xlsx" has been setup for testing purposes. It generates a 10-days optimisation using data relative to the Dutch power system.

5.2.2 Pre-processing

To run the pre-processing tool with this configuration file, several options are available:

- From the command line, specify the configuration file to be used as an argument. Within the "Dispa-SET" folder run:

```
python Dispa-SET.py -b ConfigFiles/ConfigTest.xlsx
```

- From a python IDE (e.g. Spyder), specifying the configuration file. Configure the "Dispa-SET.py" file to be run with the command line option "-b ConfigFiles/ConfigTest.xlsx". Figure 6 shows the "Run Settings" dialog properly configured for Spyder.

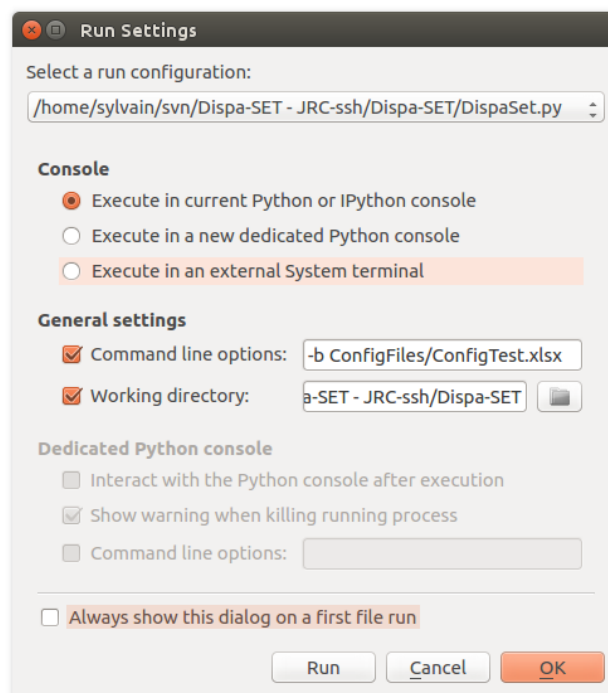


Figure 6: Example of Run Settings for the Spyder IDE

- Without specification of the configuration file. In that case the pre-processing tool prompts the user for the path to the configuration file, which should be provided as follows:

```
ConfigFiles/ConfigTest.xlsx
```

5.2.3 Check the simulation environment

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).

5.3 Run the optimisation

The optimisation can be run directly from the simulation environment folder generated by the pre-processing tool. Different options are available for that purpose:

5.3.1 Using the GAMS user interface:

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

1. Open the UCM.gpr project file and then the UCM.h.gmx model file.
2. Run the model in GAMS.

The result file is written in the.gdx format and stored in the Simulation folder, together with all input files.

5.3.2 Using the GAMS command line:

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

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

```
gams UCM_h.gms
```

5.3.3 From Dispa-SET, 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 Dispa-SET.py -br ConfigFiles/ConfigTest.xlsx
```

This generates the simulation environment, runs the optimisation, and stores the results in the same folder.

5.3.4 From Dispa-SET, using PYOMO:

The PYOMO interface of Dispa-SET can be called using the dedicated API:

- Import the Dispa-SET PYOMO solver

```
from DispaSET.DispaSolve import DispaSolve
```

- Load the inputs from a pre-processed simulation environment folder (NB: the pickle output must be activated in the pre-processing configuration file)

```
import pickle
SimData = pickle.load(open('Simulations/simulation_test/Inputs.p', 'rb'))
```

- Run the optimisation and store the results in the "result" variable.

```
results = DispaSolve(SimData['sets'], SimData['parameters'])
```

5.4 Postprocessing

Various functions and tools are provided within the PostProcessing.py file to load, analyse and plot the simulation results. The use of these functions is illustrated in the "Read_results_notebook.ipynb" Notebook, which can be run by changing the path variable to the simulation folder. 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 7). 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 8). Both the dispatch plot and the commitment plot can be called using the *CountryPlots* function.

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

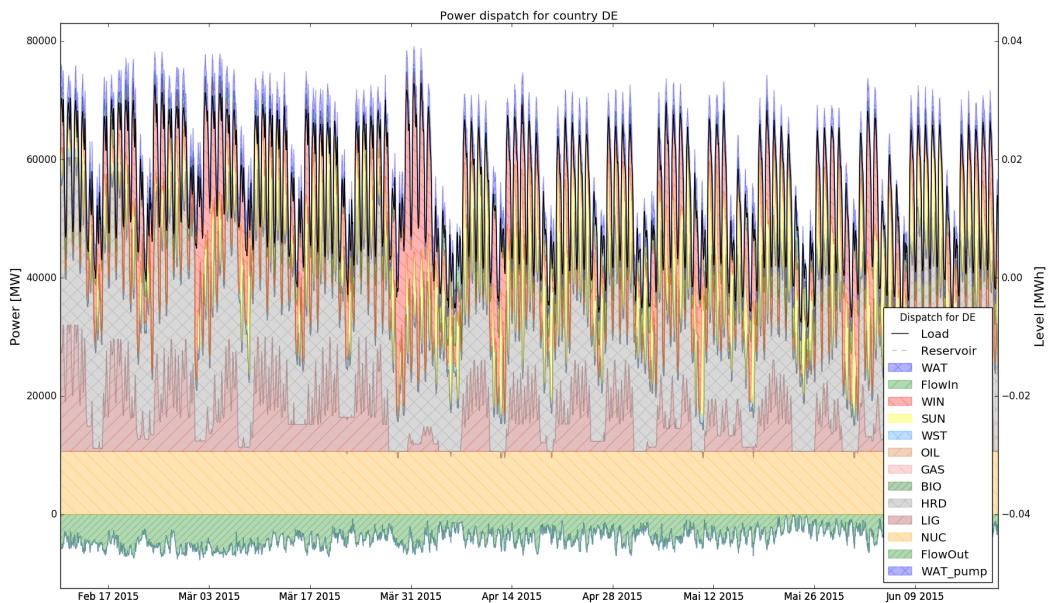


Figure 7: Example result: Power Dispatch for Germany, disaggregated by fuel type

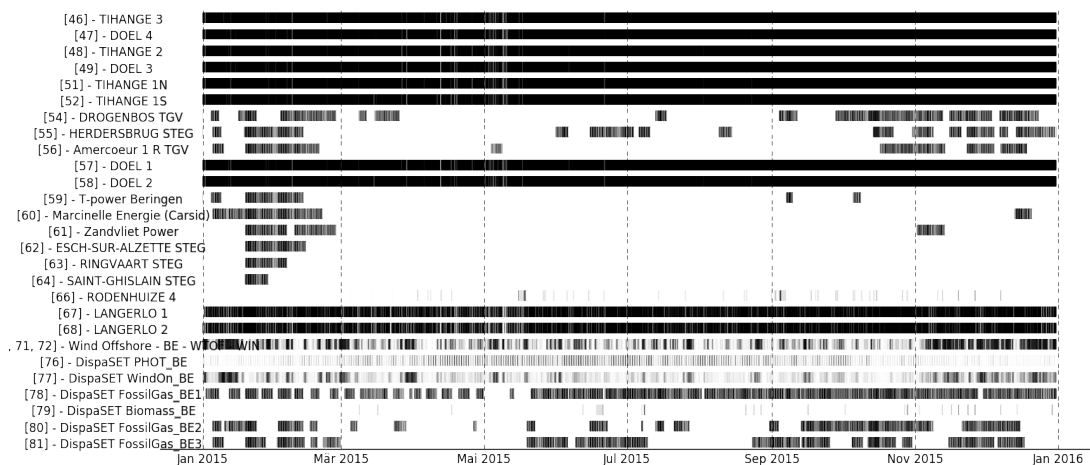


Figure 8: Example result: Commitment and power level status of each unit in Belgium

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 9):

```
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 9: Example output of the *ResultAnalysis* function

The yearly energy balance per fuel or per technology is also useful to compare the energy mix in each zone. This can be plotted using the *EnergyBarPlot* function, the output being shown in Figure 10.

```

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

```

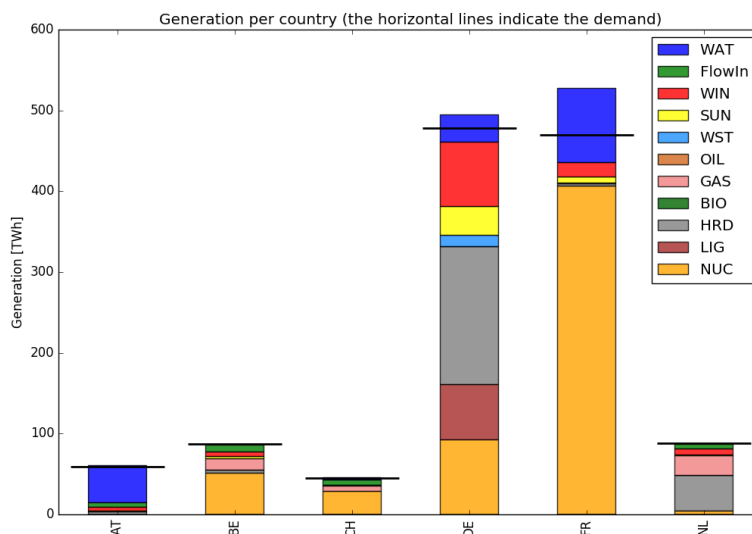


Figure 10: Example result: Energy balance per simulated country

6. Conclusions

This document has described the formulation and implementation of Dispa-SET 2.1, a model developed by the DG Joint Research Centre of the European Commission. The model is an open-source unit commitment and optimal dispatch simulation tool for power systems. It aims at representing with a high level of detail the short-term operation of large-scale power systems. It is particularly well designed to evaluate higher penetration rates of variable energy sources (e.g. wind or solar) in future EU power systems by computing effects such as flexibility requirement, lines congestion, cycling of power plants, usage of storage capacity, etc. These indicators are required to properly address different research topics relevant for supporting European energy policy making.

The tool is data-intensive: it requires high resolution time series for the demand profiles, the renewable generation, the power plant outages and the fuel prices. It also requires a database of power plants including their technical characteristics. Starting from this detailed input dataset, optimisation problems of variable complexity can be generated. The most detailed formulation is MILP, in which each unit commitment status is represented by a binary variable. The low-detail formulation is LP, in which power plants are aggregated in clusters of similar technologies and fuels. This adjustable model complexity is an interesting feature since it allows performing lower-accuracy simulations when computational efficiency is an issue (e.g. in Monte-Carlo simulations).

The model is released as an open-source tool¹ 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:

- A better representation of reserve needs, distinguishing between different types of reserves (secondary and tertiary).
- The addition of a capacity planning module.
- The linkage with long-term energy planning models such as the JRC-EU-TIMES [Simoes et al., 2013], by soft-linking [Deane et al., 2012], or by direct integration of flexibility constraints [Quoilin et al., 2015].
- The addition of new constraints (e.g. hydropower and water requirements for cooling).
- The addition of stochastic features.

¹The source code of the public version of Dispa-SET is available at: <https://github.com/squoilin/Dispa-SET>

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.
- [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>.
- [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.
- [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>.
- [Poncellet et al., 2016] Poncellet, 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., 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.
- [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.

7. Appendix: API Documentation

7.1 DispaTools

This files gathers different functions used in the Dispa-SET pre-processing and post-processing tools

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

`DispaTools.InputDir` (*msg=None*)

Function that requires the user to input a directory

`DispaTools.InputFile` (*msg=None*)

Function that requires the user to input a file path

`DispaTools.MergeSeries` (*plants, data, mapping, method='WeightedAverage'*)

Function that merges the times series corresponding to the merged units (e.g. out-ages, 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')

Return merged Pandas dataframe with the merged time series when necessary

`DispaTools.ParamDefinition` (*sets_in, sets, value=0*)

Function to define a Dispa-SET 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

`DispaTools.append_to_dict` (*k, source, destination*)

Function to add a record to a dictionary (e.g. a new power plant) from another dictionary

`DispaTools.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)
```

`DispaTools.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 Dispa-SET format)
- **method** – Select clustering method ('Standard'/'LP'/None)

- **Nslices** – Number of slices used to fingerprint each power plant characteristics 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

`DispaTools.ds_to_df(inputs)`

Function that converts the Dispa-SET data format into a dictionary of dataframes

Parameters

- **list_vars** – List containing the Dispa-SET variables
- **format** – Format version of the Dispa-SET variables
- **timeindex** – Time index to be applied to the dataframes. The size of the output is adapted accordingly.

Returns dictionary of dataframes

`DispaTools.find_nearest(array, value)`

Function to find the index of the nearest value in a vector

`DispaTools.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 and Linux. More searching rules and patterns should be added in the future

`DispaTools.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

`DispaTools.get_sets(instance, varname)`

Get sets that belong to a pyomo variable or parameter

Parameters

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

Returns A list with the sets that belong to this Parameter

`DispaTools.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 a special case in Dispa-SET

`DispaTools.interconnections(Simulation_list, list_all_countries, NTC_inter, Historical_flows)`

Function that checks for the possible interconnections of the countries included in the simulation. If the interconnection is defined 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 is defined 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
- **list_all_countries** – List of all countries
- **NTC** – Day-ahead net transfer capacities (pd dataframe)
- **Historical_flows** – Historical flows (pd dataframe)

`DispaTools.invert_dic_df(dic)`

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

Returns dictionary of dataframes, with swapped headers

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

Function that loads a csv sheet into a dataframe and saves a temporary pickle version of it. If the pickle is newer than the sheet, the later is not loaded again to increase the computational efficiency.

Parameters

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

`DispaTools.load_set(input, string, set_name)`

Load a particular set from the Dispa-SET input data structure (Obsolete for v2.1.0 onwards)

`DispaTools.load_var(input, string)`

Load a particular variable from the Dispa-SET input data structure v2.0 (Obsolete from v2.1.0 onwards)

`DispaTools.load_xl_to_pd(path_excel, file_excel, sheet_excel, path_pandas, file_pandas, header=0)`

Function that loads an xls sheet into a panda variable and saves it in a separate path. If the saved variable is newer than the sheet, do no load the sheet again.

`DispaTools.mylogspace(low, high, N)`

Ad Hoc logspace function in which low and high are the first and last values of the space

`DispaTools.pyomo_format(sets, param)`

Function that flattens the multidimensional Dispa-SET input data into the pyomo format: a dictionary with a tuple and the parameter value. The tuple contains the strings of the corresponding set values

`DispaTools.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)

`DispaTools.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 Shrunked string or list of strings

`DispaTools.tuple_format(array)`

Function that flattens a n-dimensional matrix and returns a dictionary with the values and their coordinates in a tuple

7.2 DispaCheck

This files gathers different functions used in the Dispa-SET to check the input data

`DispaCheck.check_df(df, StartDate=None, StopDate=None, name=^)`

Function that checks the time series provided as inputs

`DispaCheck.check_simulation_environment(SimulationPath, type='pickle', first-line=7)`

Function to test the validity of Dispa-SET inputs :param SimulationPath: Path to the simulation folder :param type: choose between: "list", "excel", "pickle" :param first-line: Number of the first line in the data (only if type=='excel')

`DispaCheck.check_units(config, plants)`

Function that checks the power plant characteristics

7.3 IO - data

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)

`Dispa-SET_io_data.GdxToDataframe(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 GdxTo-List function
- **fixindex** – This flag allows converting string index into integers and sort the data

Returns dictionary of dataframes

`Dispa-SET_io_data.GdxToList(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)

`Dispa-SET_io_data.GetGdx(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

`Dispa-SET_io_data.LoadConfig(ConfigFile)`

Function that loads the Dispa-SET excel config file and returns a dictionary with the values

Parameters `ConfigFile` – String with (relative) path to the Dispa-SET excel configuration file

`Dispa-SET_io_data.insert_symbols(gdxHandle, sets, parameters)`

Function that writes all sets and parameters to the `gdxHandle`

Parameters

- `sets` – dictionary with all the sets
- `parameters` – dictionary with all the parameters

`Dispa-SET_io_data.write_toexcel(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 Dispa-SET variables

Returns Binary variable (True)

`Dispa-SET_io_data.write_variables(gams_dir, gdx_out, list_vars)`

This function performs the following: (1) Use the `gdxcc` library to create a `gdxHandle` instance (2) Check that the `gams` path is well defined (3) 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

7.4 Input generation

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

@author: S. Quoilin

`GenerateInputs.BuildSimulation(config)`

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

Parameters `config` – dictionary with all the configuration fields loaded from the excel file. Output of the `'LoadConfig'` function.

7.5 Post Processing

Set of functions useful to analyse to Dispa-SET output data.

@author: Sylvain Quoilin, JRC

PostProcessing.**AggregateByFuel** (*PowerOutput, Inputs, SpecifyFuels=None*)

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

Parameters

- **PowerOutput** – Dataframe of power generation with units as columns and time as index
- **Inputs** – Dispa-SET inputs version 2.1.0
- **SpecifyFuels** – If not all fuels should be considered, list containing the relevant ones

Returns PowerByFuel Dataframe with power generation by fuel

PostProcessing.**CountryPlots** (*inputs, results, c, rng=[]*)

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

Parameters

- **inputs** – Dispa-SET inputs
- **results** – Dispa-SET results
- **c** – Considered country (e.g. 'BE')

PostProcessing.**EnergyBarPlot** (*datain, results, PPindicators*)

Plots the generation for each country, disaggregated by fuel type

Parameters

- **datain** – Inputs values (in the dataframe format: output of the function `ds_to_df`)
- **results** – dictionary with the outputs of the model (output of the function `GetResults`)
- **PPindicators** – Power plant statistics (output of the function `PerPowerPlantIndicators`)

PostProcessing.**FilterByCountry** (*PowerOutput, inputs, c*)

This function filters the Dispa-SET Output Power dataframe by country

Parameters

- **PowerOutput** – Dataframe of power generation with units as columns and time as index
- **Inputs** – Dispa-SET inputs version 2.1.0
- **c** – Selected country (e.g. 'BE')

Returns Power Dataframe with power generation by fuel

PostProcessing.**GetDemand** (*Inputs, c*)

Get the demand curve of a specific country

Parameters

- **Inputs** – Dispa-SET inputs
- **c** – Country to consider (e.g. 'BE')

PostProcessing.**GetPlotData** (*inputs, results, c*)

Function that reads the results dataframe of a Dispa-SET simulation and extract the dispatch data specific 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

PostProcessing.**GetResults** (*path='.', cache=False, TempPath='.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
- **TempPath** – Temporary path to store the cache file

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

PostProcessing.**PerPowerPlantIndicators** (*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** – Dispa-SET inputs
- **results** – Dispa-SET results

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

PostProcessing.**ResultAnalysis** (*inputs, results*)

Reads the Dispa-SET results and provides useful general information to stdout

Parameters

- **inputs** – Dispa-SET inputs
- **results** – Dispa-SET results

PostProcessing.**UnitFuel** (*Inputs, shrink=True*)

Function that associates its fuel to each unit from the Dispa-SET inputs

Parameters

- **inputs** – Dispa-SET inputs (version 2.1.0)
- **shrink** – If True, the unit name is reduced to 64, as in GAMS

Returns fuels Dictionary with the location of each unit

PostProcessing.**UnitLocation** (*Inputs, shrink=True*)

Function that associates its location to each unit from the Dispa-SET inputs

Parameters **inputs** – Dispa-SET inputs (version 2.1.0)

Returns loc Dictionary with the location of each unit

PostProcessing.**colormap** (*f*)

Function that associates a color to each dispa-SET fuel for plotting

The complete list can be obtained with:

```
matplotlib.colors.cnames
```

PostProcessing.**dispatch_plot** (*demand, plotdata, level=None, rng=[]*)

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 GetPlot-Data

- **level** – Optional pandas series with an aggregated reservoir level for the considered zone.
- **rng** – Indexes of the values to be plotted. If undefined, the first week is plotted

PostProcessing.**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

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

Create multiaxis rug plot from a 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 are colored based on their value.

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

Returns: plot

Function written by K. Kavvadias

7.6 DispaSolve

This worksheet contains the two main functions to solve the Dispa-SET optimization problem using PYOMO.

@author: 'Sylvain Quoilin'

DispaSolve.**DispOptim** (*sets, parameters, LPFormulation=False*)

This is the main optimization function of Dispa-SET. Two operation are performed: 1. Translation of the Dispa-SET data format into the pyomo format 2. Definition of the Pyomo optimization model as a ConcreteModel

Parameters

- **sets** – Dictionary containing the sets (defined as a list of strings or integers)
- **parameters** – Dictionary containing the parameters of the optimization problem (in the Dispa-SET 2.1.0 format)

Returns The Pyomo optimization model instance

DispaSolve.**DispaSolve** (*sets, parameters, LPFormulation=False*)

The DispaSolve function defines the rolling horizon optimization and saves each result variable in a pandas dataframe. The definition of the rolling horizon must be included into the Dispa-SET Config parameter'

Parameters

- **sets** – Dictionary containing the sets (defined as a list of strings or integers)
- **parameters** – Dictionary containing the parameters of the optimization problem (in the Dispa-SET 2.1.0 format)

Returns Dictionary of pandas dataframes with the optimization variables

***Europe Direct is a service to help you find answers
to your questions about the European Union.***

Freephone number (*):

00 800 6 7 8 9 10 11

(* The information given is free, as are most calls (though some operators, phone boxes or hotels may charge you).

More information on the European Union is available on the internet (<http://europa.eu>).

HOW TO OBTAIN EU PUBLICATIONS

Free publications:

- one copy:
via EU Bookshop (<http://bookshop.europa.eu>);
- more than one copy or posters/maps:
from the European Union's representations (http://ec.europa.eu/represent_en.htm);
from the delegations in non-EU countries (http://eeas.europa.eu/delegations/index_en.htm);
by contacting the Europe Direct service (http://europa.eu/europedirect/index_en.htm) or
calling 00 800 6 7 8 9 10 11 (freephone number from anywhere in the EU) (*).

(* The information given is free, as are most calls (though some operators, phone boxes or hotels may charge you).

Priced publications:

- via EU Bookshop (<http://bookshop.europa.eu>).

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.



EU Science Hub

ec.europa.eu/jrc



@EU_ScienceHub



EU Science Hub - Joint Research Centre



Joint Research Centre



EU Science Hub

