University of Liège - School of Engineering and Computer Science

MASTER'S THESIS

Improving the simulation of variable renewable energy in integrated assessment models

Master's thesis completed in order to obtain the degree of Master of Science in Computer Science Engineering by Straet François

Supervisors:
Prof. S. Quoilin, Prof. B. Boigelot

Academic year 2022-2023

Keywords

CF	Capacity factor
DLL	Dynamically loaded library
LP	Linear programming
MILP	Mixed integer linear programming
MTS	Mid-term scheduling
P2H	Power
RES	Renewable energy sources
VRES	Variable renewable energy sources

Abstract

TODO

Acknowledgements

Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region

TODO

Todo-list

1. Put setcounter macro to match the arabic numberings to the actual page numbers
2. Check that all Keywords are actually used

Contents

Warnings

I have ?? warnings:

- `Warning` No.1
- `Warning` No.0
- `Warning` No.-1

1 Introduction

Todo

2 The Dispa-SET model

2.1 Overview

The online documentation describes the Dispa-SET [4] model as "an open-source unit commitment and optimal dispatch model focused on the balancing and flexibility problems in European grids".

More precisely, it is focused on simulating large scale power systems, with emphasis on high shares of VRES. It follows that it is used as tool for the analysis of the impacts of VRES on the power systems, thank to its ability to take into account several technical constraints of the power system.

A schematic of the Dispa-SET architecture is given in Figure **??**.

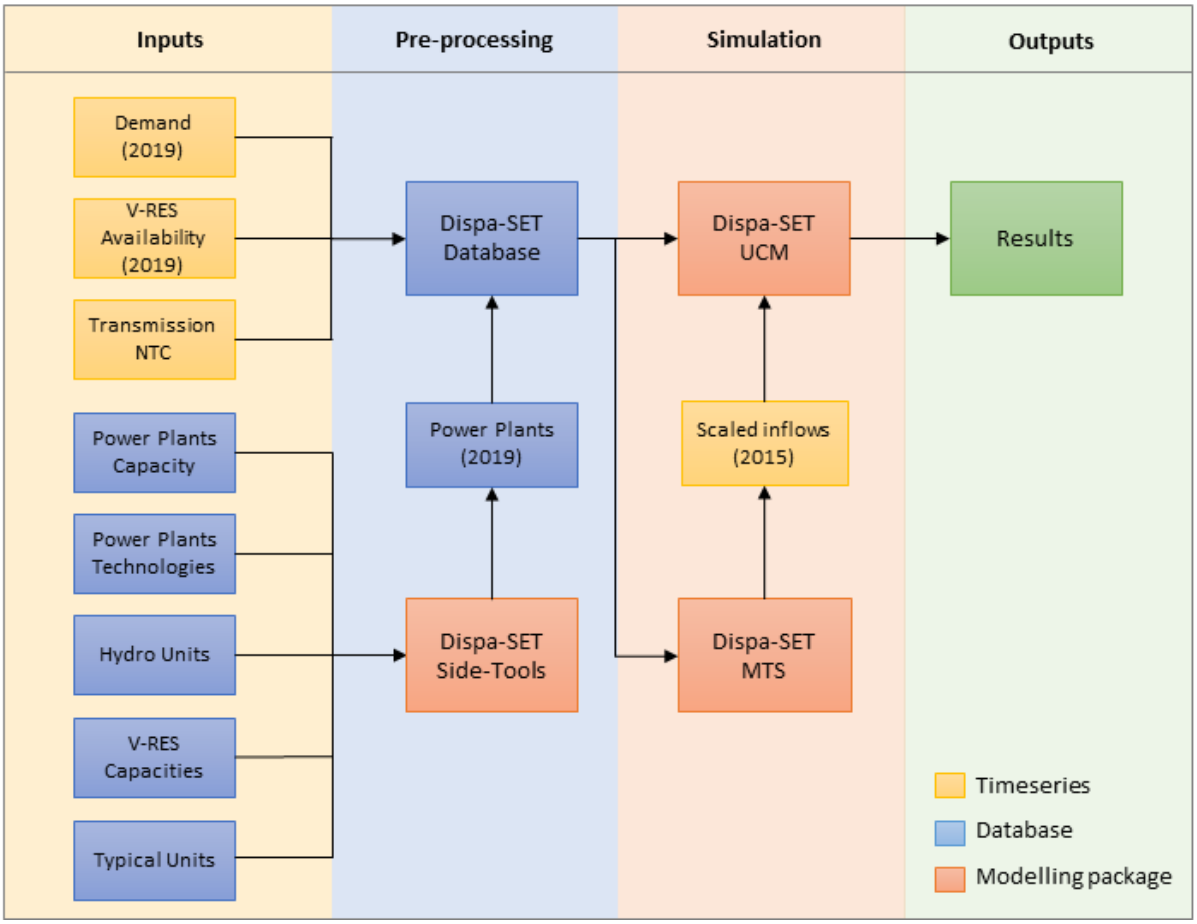


Figure 2.1: Block diagram of Dispa-SET architecture

Dispa-SET has several option regarding the formulation of the problem:

– Linear programming (LP) optimization problem

– Mixed Integer Linear Programming optimization problem.

Its interface is written in the Python programming language, and calls GAMS [1] as the main solver engine.

2.2 Objective function

The Dispa-SET model aims at minimizing the overall operating costs of the grid, that is, its objective function. These costs typically include transportation, power and heating costs required to split efficiently the demand between the available generation units.

The total system costs is splitted as follows:

- *Fixed cost*: fixed amount, charged if the unit is on.
- *Variable costs*: amount that is a function of the power output the units are operating at.
- *Start-up* and *Shutdown costs*: amount charged on start and on shutdown of a unit.
- *Ramp-up* and *Ramp-down costs*: costs due to the fact *Needs verification*
- *Shed load costs*: costs due to necessary load sheddings.
- *Loss of load costs*: due to generated either exceeding the demand, or not matching it .
- *Transmission costs*: due to the use and wear of the transmission network.
- *Spillage costs*: due to spillage in storage units.

Adding these, Equation **??** is obtained, where u refers to the index on each units, and i is the time index.

2.3 Supply and demand balance

At all time and in each zone, the fundamental constraint that has to be met is the supply-demand balance in terms of energy production (supply) and consumption (demand), in the day-ahead market.

The supply sources are:

- The power outputs from each units
- The power outputs from storage units discharging
- The (eventual) net income from importation from neighbouring zones
- The (eventual) shed load

Whereas the demand originates from:

- The load in that zone
- The (eventual) net exportations to neighbouring zones
- The power consumed by charging storage units
- The power consumed by P2H units

The Equation **??** expresses this target equality.

$$
\begin{aligned}
Min_{TotalSystemCost} = \\
&\sum_{u,i}(CostStartUp_{i,u} + CostShutDown_{i,u}) + \\
&\sum_{u,i}(CostRampUp_{i,u} + CostRampDown_{i,u}) + \\
&\sum_{u,i} CostFixed_u \cdot Comitted_{i,u} \cdot TimeStep + \\
&\sum_{u,i} CostVariable_{i,u} \cdot Power_{i,u} \cdot TimeStep + \\
&\sum_{hu,i} CostVariable_{i,u} \cdot Heat_{i,u} \cdot TimeStep + \\
&\sum_{l,i} PriceTransmission_{i,l} \cdot Flow_{i,l} \cdot TimeStep + \\
&\sum_{n,i} CostLoadShedding_{i,n} \cdot ShedLoad_{i,n} \cdot TimeStep + \\
&\sum_{n\_th,i} CostHeatSlack_{n\_th,i} \cdot HeatSlack_{n\_th,i} \cdot TimeStep + \\
&\sum_{n\_h2,i} CostH2Slack_{n\_h2,i} \cdot H2Slack_{n\_h2,i} \cdot TimeStep + \\
&\sum_{chp,i} CostVariable_{chp,i} \cdot CHPPowerLossFactor_{chp,i} \cdot Heat_{chp,i} \cdot TimeStep + \\
&\sum_{i,n} VOLL_{Power}(LL_{MaxPower,i,n} + LL_{MinPower,i,n}) \cdot TimeStep + \\
&\sum_{i,n} 0.8 \cdot VOLL_{reserve}(LL_{2U,i,n} + LL_{2D,i,n} + LL_{3D,i,n}) \cdot TimeStep + \\
&\sum_{u,i} 0.7 \cdot VOLL_{Ramp}(LL_{RampUP,u,i} + LL_{RampDown,u,i}) \cdot TimeStep + \\
&\sum_{s,i} CostOfSpillage \cdot Spillage_{s,i}
\end{aligned}
\tag{2.1}
$$

Equation 2.1: Objective function of the Dispa-SET model

$$\sum_{u}(Power_{u,i} \cdot Location_{u,n}) + \sum_{l}(Flow_{u,i} \cdot LineNode_{l,n})$$
$$= Demand_{DA,n,i} + Demand_{Flex,n,i} + \sum_{s}(StorageInput_{s,i} \cdot Location_{s,n}) +$$
$$\sum_{p2h}(PowerConsumption_{p2h,i} \cdot Location_{p2h,i}) - ShedLoad_{n,i} - LL_{MaxPower_{n,i}} + LL_{MinPower_{n,i}} \tag{2.2}$$

Equation 2.2: Supply-demand balance in Dispa-SET

2.4 Rolling horizon

I'm not sure I understood all

2.5 Mid-term scheduling (MTS)

Witout further constraints, the optimization will most often leave all the storage facilities empty at the end of the simulation horizon (typically a few days). This comes from the fact that the variable operational cost of discharging these storage units is really small in comparison to running another unit, thus charging extra fixed and variable costs.

To take this into account, the minimum storage level at the last optimization step is given as an exogenous input to the model. This input can be computed from the so-called Mid-term hydro-thermal scheduling (MTS), that is a long-term scheduling specification.

Talk about MTS option

2.6 Dispa-SET components and representation

Dispa-SET provides us with several predefined configurations, each of these defining the zones and their units of interest, and linking to the relevant data (e.g. times series provided as `csv` files).

In this work, the european setting is used.

2.6.1 Zones

Most of the EU contries are represented, for completeness they are reported on Table **??**.

2.6.2 Technologies

Table **??** lists all the technologies taken into account by Dispa-SET, alongside with their main properties:

- VRES: does the technology belongs to VRES?
- Storage: can it store energy?
- Flexibility: ease of control of the unit's power output.

Code	Country	Code	Country
AT	Austria	IE	Ireland
BE	Belgium	IT	Italy
BG	Bulgaria	LT	Lithuania
CH	Switzerland	LV	Latvia
CZ	Czech Republic	NL	Netherlands
DE	Germany	NO	Norway
DK	Denmark	PL	Poland
EE	Estonia	PT	Portugal
EL	Greece	RO	Romania
ES	Spain	SE	Sweden
FI	Finland	SI	Slovenia
FR	France	SK	Slovakia
HR	Croatia	UK	United Kingdom
HU	Hungary		

Table 2.1: Countries present in Dispa-SET EU, and their ISO Alpha 2 country codes. These are all the EU contry except for Cyprus and Malta and Luxembourg, plus Norway, Switzerland and the UK.

Due to the intermittency of their resources, and because one cannot dispatch them, VRES are considered inflexible.

Although, hydroelectric units disposing of a reservoir are have some room for flexibility, given their ability to manage their storage level.

Steam turbines, because of their dependency on the fuel used, e.g. nuclear energy would be less flexible than natural gas.

Heating and combined heat and power units are not covered, as only the electricity is of interest in this scope.

2.6.3 Fuels

Table **??** summarizes the fuel types in Dispa-SET.

Identifier	Description	VRES	Storage	Flexibility
COMC	Combined cycle	No	No	High
GTUR	Gas turbine	No	No	High
ICEN	Internal combustion engine	No	No	High
STUR	Steam turbine	No	No	Medium
HDAM	Conventional hydro dam	No	Yes	Medium
HROR	Hydro run-of-river	Yes	No	Low
HPHS	Pumped hydro storage	No	Yes	Medium
WTOF	Offshore wind turbine	Yes	No	Low
WTON	Onshore wind turbine	Yes	No	Low
PHOT	Solar photovoltaic	Yes	No	Low
BATS	Stationary batteries	No	Yes	High

Table 2.2: Technologies present in Dispa-SET

It is important to highlight that technologies may not always be powered by the same fuel, for instance, the steam turbines can use most of them.

Each unit must specify its technology and fuel. Depending on the optimization problem formulation, units featuring the same (technology-fuel) pair will be grouped together and thereafter be treated as one single unit.

Fuel	Description
BIO	Biofuels
GAS	Gas
HRD	Coal
LIG	Lignite
NUC	Nuclear energy
OIL	Petroleum
PEA	Peat Moss
GEO	Geothermal steam
SUN	Solar energy
WAT	Hydro energy
WIN	Wind energy
WST	Energy from waste
OTH	Other fuels and energy carriers

Table 2.3: Fuel types in Dispa-SET

A major consideration for the optimization problem is the fuel prices, that are summarized in Table **??**.

A key feature is the relationship between the price of coal and the price of gas: depending on which one is the cheaper, the optimal behaviour change dramatically. Obviously, the cheapest one will always be preferred over the other when choice arise.

	Price
Nuclear	3
Black coal	20
Gas	45
Fuel-Oil	65
Biomass	10.08
Lignite	7.23
Peat	9.36

Table 2.4: Fuel prices considered[€/MWh]

2.6.4 Other prices

Some other price values are relevant, such as the price of the load shedding per MWh. These are presented in Table **??**.

What	Price
CO2	25
Unserved Heat	84.21
Load Shedding Cost	1000
Transmission	0
Unserved H2	75
Curtailment Cost	20

Table 2.5: Other relevant prices [€/MWh]

Field	Description	Type
Unit	Unit name	string
PowerCapacity	Maximum power output	value in MW
Nunits	Number of initial units clustered	integer
Zones	The unit's zone	string
Fuel	The fuel used	string
Efficiency	The unit's efficiency	real in [0,1]
MinEfficiency	Efficiency at minimum load	real in [0,1]
MinUpTime	Minimum up time	value in hours
MinDownTime	Minimum down time	value in hours
RampUpRate	Ramp up rate	value in minute^{-1}
RampDownRate	Ramp down rate	value in minute^{-1}
RampingCost	Cost of raming up or down	value in €hour
$\text{StartUpCost}_p u$	Start up cost per clustered unit	value in €
$\text{NoLoadCost}_p u$	Cost of having no load on a unit	value in €/hour
PartLoadMin	Ratio of the minimum nominal capacity	real value
StartUpTime	Time to start up the plant	value in hour
CO2Intensity	Amount of CO_2 emitted per MW	value in €/MW

Table 2.6: The table fields used to describe a optionnally aggregated power plant unit

2.6.5 Power plants

As a dispatch model, Dispa-SET oviously has to model the units it dispatches, namley the power plants that are present in each of the modelled zones.

For performance reasons, some of the units initially described are merged into clustered units at the pre-processing step. Thus, the amount of variable in the simulation is reduced, while the accuracy is not significantly impacted [4].

Dispa-SET disposes of utilities to do so, but also needs to craft the new, aggregated units properties table. These are defined by the set of feilds that are shown on Table **??**.

2.6.6 Notes on the other inputs

– The electricity demand is a time series from year 2019, per zone. It is assumed to be independent of the price.

– The net transfer capacities (NTC) between the different zones are given as inputs as hourly times series over a year. Then the maximum is picked and it is assumed that it remains

constant over the year. *Needs verification*

– The availability factors (AF) for renewable energy sources, defined as the ratio of the nominal power that is possible to output hourly. It is given as an hourly time series (adimentional).

  This variable energy generation is either curtailed or sent to the grid.

  Non-renewable technologies have their AF set to 1.

3 Data generation

3.1 Overview

This section aims at describing the process that lead to the creation of the dataset, required in order to train the neural network model.

First, an input space is defined, on which will properly select points to form the dataset features. Then, computationally expensive simulation will be run on these points to obtain the desired predicted features for this dataset.

This dataset will afterwards be used to train the surrogate model on [11].

3.2 Preparatory work

As stated earlier, this setting only considers the european power system in Dispa-SET, then to create and validate our surrogate model. Each simulation is run over a period of 2019.

3.2.1 Unit groupings

In this context, the precise technology and fuel types of each plant is not relevant, as they won't influence the input features of our dataset. Hence, the units are grouped into five categories: flexible units, slow units, storage units, PV units and wind units.

IRENA [10] describes flexible units as "units that can ramp up and down quickly, have a low minimum operating level and fast start-up and shutdown times", what criterion will be used to separate regular units into the slow and flexible units. This criterion is presented in Table **??**.

Units	Fuel	Condition
$Flex_{units}$	GAS, HRD, OIL, BIO, LIG, PEA, NUC, GEO	$PartLoadMin < 0.5$ and $TimeUpMin < 5$ and $RampUpRate > 0.01$
$Slow_{units}$		$PartLoadMin \geq 0.5$ or $TimeUpMin \geq 5$ or $RampUpRate \leq 0.01$

Table 3.1: Flexible and slow units classification criterion

Refer to Tables **??** and **??** for their names.

One also has to consider the limit to the number of hydroelectric units that are possible to build given a geographical area. Given that EU is already almost at saturation, stationary batteries are considered, among other energy storage technology (e.g. compressed air, electric vehicles' battery grid).

As for the other groups, they can be simply described with technology-fuel pairs as follows:

- $Storage_{units}$ with (OTH, BATS)
- PV_{units} with (SUN, PHOT)
- $Wind_{units}$ with either (WIN, WTON) or (WIN, WTOF), the latter not being considered in this work

3.2.2 Parameters estimates

The availability factors of PHOT and WTON are also required, as well as the peak load. These values are computed from the reference simulation (2019), and are shown in Table **??**.

needs check

Variable	Value	Units
AF_{PV}	0.1313	[·]
AF_{WTON}	0.2604	[·]
$PeakLoad$	440929	MW

Table 3.2: Values of availability factors and peak load

3.3 Design space

3.3.1 Shape

The first necessary step in order to select our data points for our dataset, is to define the space in which we will sample them. In our case, this space will be the product of 6 ranges, that is a 6 dimensional hypercube.

One may argue that some areas of this hypercube, typically around the vertices, will be extremely unlikely to happen in a real setting. More precisely, as this cube will be the input space of the surrogate model that will be connected to another model, it may be suitable to prune the areas of the cube that will never be reached. Indeed, if we know that some areas will never be queried, there is no use covering them.

Furthermore, assuming we would obtain the exact space of possible queries, this space is not likely to be close to some common shape (hypercube, hyperball or combination). Given that most of the design of experiments techniques assume these kinds of space, a mapping would be needed to benefit from the better sampling strategies. Such a mapping would be pretty complex to develop.

More importantly, the cost of being more general than strictly required is small, mainly consisting of a slightly larger surrogate model (in this specific case, a larger neural network), and a larger dataset.

For these reasons, an hypercube will be used.

3.3.2 Variables

The six adimensional variables, corresponding to a dimension, are described below, with their given range.

The notation $PowerCap_x$ refers to the maximum power output of all the units in x. See Table **??** for the values of the AF and peak load value.

1. **CapacityRatio** [·]

   Ratio of the maximum production over the maximum demand.

$$CapacityRatio = \frac{PowerCap_{flexunits} + PowerCap_{slowunits} + PowerCap_{storageunits}}{PeakLoad} \tag{3.1}$$

2. **ShareFlexibility** [·]

   Share of the units that are flexible.

$$Share_{flex} = \frac{PowerCap_{flexunits}}{PowerCap_{flexunits} + PowerCap_{slowunits}} \tag{3.2}$$

3. **ShareStorage** [·]

   Ratio of the maximum power output of all storage units over the maximum demand.

$$Share_{storage} = \frac{PowerCap_{storageunits}}{PeakLoad} \tag{3.3}$$

4. **ShareWind** [·]

   Ratio of the maximum power output of all wind units over the maximum demand.

$$Share_{wind} = \frac{PowerCap_{windunits}}{PeakLoad} \cdot AF_{WTON} \tag{3.4}$$

5. **SharePV** [·]

   Ratio of the maximum power output of all PV units over the maximum demand.

$$Share_{PV} = \frac{PowerCap_{PVunits}}{PeakLoad} \cdot AF_{PV} \tag{3.5}$$

6. **rNTC** [·]

   Net transfer capacity ratio. This variable is a measure of the grid effect on the network, as the zones are able to transmit power between them.

   The data we are provided contains hourly logs of the power transmitted between each pair of zones. The following describes how to compute the rNTC value given these.

   First, we compute the average net transfer capacity (NTC) for each zone z to any other zone x over each of the N_h hours in the input data, via Equation **??**.

$$NTC_{z \to x} = \frac{1}{N_h} \sum_h NTC_{z \to x,h} \tag{3.6}$$

   Then Equation **??** is used to compute the zonal NTC, that is the ratio of the sum of all NTCs from this zone to any other zones, over the peak load for that zone.

$$NTC_z = \frac{\sum_x NTC_{z \to x}}{PeakLoad_z} \tag{3.7}$$

   A zonal NTC of 1 for zone z thus means that z would be able, at any time, to fulfill the integrity of its demand by importing electricity from connected zones.

   The final rNTC value is a weighed sum of the zonal NTCs. The weight for a zone z is computed as the ratio of its peak load over the sum of each peak loads. This is expressed by Equation **??**.

$$rNTC = \sum_z \frac{PeakLoad_z}{\sum_x PeakLoad_x} NTC_z \tag{3.8}$$

3.3.3 Reference values and ranges

Given the 2019 input data, a reference simulation is run and one obtains the values presented in table **??**.

Variable	Value	Lower bound	Upper bound
CapacityRatio	1.658	0.5	1.8
ShareFlexibility	0.418	0.01	0.99
ShareStorage	0.497	0	0.5
ShareWind	0.106	0	0.5
SharePV	0.035	0.2	0.5
rNTC	0.282	0	0.7

Table 3.3: Values of the different variable for the reference simulation in 2019.

SharePV c'est un peu optimiste ou il y a un zéro en trop ?

3.4 Design of experiments

A strategy to choose sampling points from some design space is called a design of experiment (DoE). It aims at producing a set of samples that represent as best as possible the entire design space, a property that is required to obtain a well balanced dataset.

The main methods to acheive such sampling are [6] illustrated in the following.

1. The "naïve" sampling: take samples at regular intervals on the design space. Note that one may not choose the same intervals for different dimensions. It is depicted on Figure **??**.
2. The Monte-Carlo sapling: pick samples at random all over the design space. It is depicted on Figure **??**.
3. The Latin-hypercube sampling [14], maximizing a criterion that is either [2]:
   (a) centering samples in sampling intervals
   (b) maximizing the minimum distance between two samples
   (c) maximizing the minimum distance between two samples, but place sample in a random location in its interval
   (d) minimizing the maximum correlation between two samples

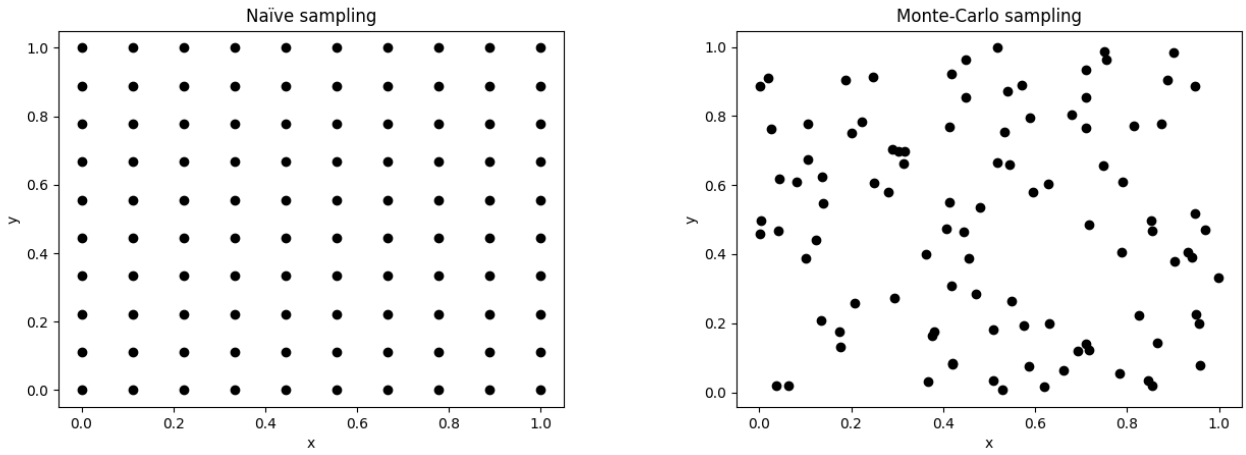
   These are depicted on Figures **??**, **??**, **??** and **??**.

From these 2-dimensional illustration it is clear that the latin hypercube sampling performs best, the naive sampling featuring too much regularities that is not wanted, as they may introduce some bias, and Monte-Carlo sampling tends to make more clusters of samples, that would be inefficient (indeed, making two times the same simulation is useless).

3.5 Generation of the dataset

With the input samples now obtained, the next step is now to compute the simulations on each of these points.

But this task is not trivial: we don't have the data that corresponds to these exact configuration. It is thus needed to craft some new simulation settings given a reference, that is the year 2019.



(a) Naïve strategy

(b) Monte-Carlo strategy

Figure 3.1: Basic sampling strategies

3.5.1 Adjusting function

To do so, Dispa-SET disposes of utility functions that do just that, adjusting the capacities, i.e., the power ouputs of the units, in function of some parameters.

These adjusting functions have been specifically developped for this topic by Vidal in her master thesis [13].

– `adjust_flexibility` modifies installed capacities to reach the desired $Share_{flex}$.

To do so, it first computes the target capacity, by multiplying the total capacity by the desired $Share_{flex}$. It then add or subtracts the missing or exceeding flexible unit power capacity to each zone, weighting by their total capacity, following Equation **??**.

$$capacity_{z,new} = capacityz, old + \frac{capacity_{z,new}}{\sum_x capacity_{x,old}}(target - actual) \tag{3.9}$$

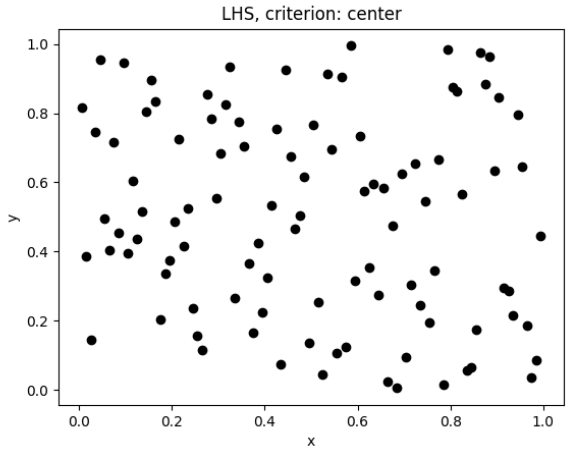
– `adjust_capacity` applies a linear scaling to the power output of some given set of units, in particular, it will be called multiples times to adjust the storage, PV and wind capacities.

Scaling factors applied are summarized in Table **??**.

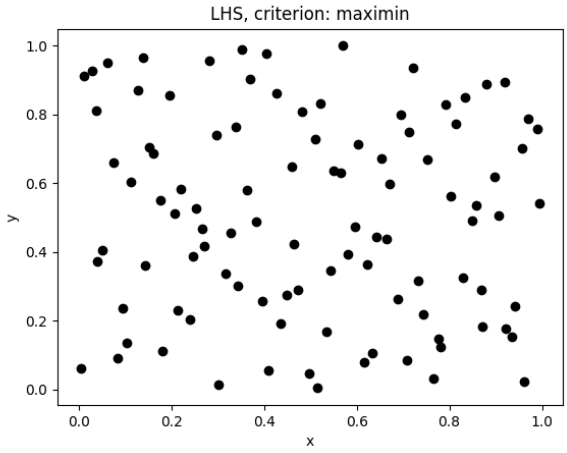
Units	Scaling factor
$Storage_{units}$	$Share_{storage}$
$Wind_{units}$	$\frac{CapacityRatio \cdot Share_{wind}}{AF_{wton}}$
PV_{units}	$\frac{CapacityRatio \cdot Share_{PV}}{AF_{PV}}$

Table 3.4: Scaling factors applied to different units

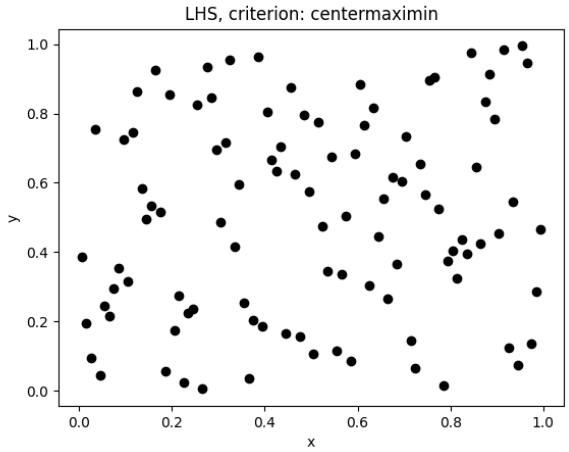
– `adjust_rntc` applies a linear scaling to each zonal NTC time series.



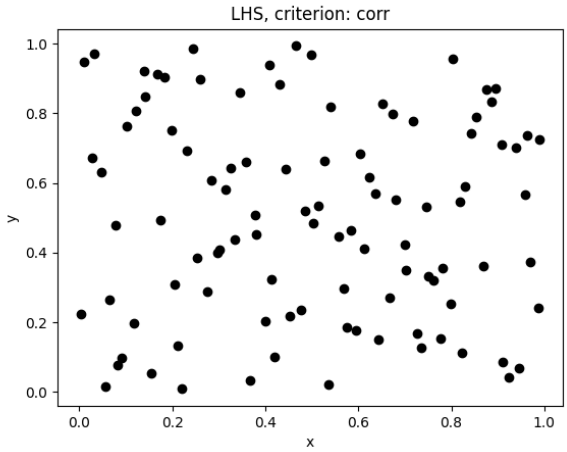
(a) LHS with center criterion



(b) LHS with max min distance criterion



(c) LHS with max min distance and center criterion



(d) LHS with correlation criterion

Figure 3.2: Latin hypercube sampling strategies with every criterion

3.5.2 Extracted outputs

As every simulation runs outputs a lot more than only the curtailment and lost load values, some other outputs variables will be extracted from the simulations, while not directly useful for the time being.

Of course, these may reveal themselves useful for future work.

All the outputs extracted from the simulations are displayed in Table **??**.

3.6 Implementation

One simple, yet important notice: running all of these simulation is not feasible on a basic hardware. This arises the need for the cluster use, and thus of submitting these as jobs on the cluster.

As the NIC5 cluster provided by CÉCI is obviously shared, one needs to manage the submitted jobs appropriately. In our case, we simply have the same program to be run a bunch of times, that are jobs independent of each other.

Parameter	Unit	Parameter	Unit
Cost	€/MWh	Shedding	TWh
Congestion	h	LostLoad	TWh
PeakLoad	MW	CF gas	[·]
MaxCurtailment	MW	CF nuc	[·]
MaxLoadShedding	MW	CF wat	[·]
Demand	TWh	CF win	[·]
NetImports	TWh	CF sun	[·]
Curtailment	TWh		

Table 3.5: Values extracted from each simulations

3.6.1 Steps

For a complete experiment to be completed, these steps have to be followed:

1. Generating the reference simulation, to extract the data that will be manipulated by the adjusting function
2. For each sample, do:
   (a) Call the adjusting function and create the simulation directory, with all the simulation input data
   (b) Call GAMS in this simulation directory
   (c) Fetch GAMS outputs in this directory

3.6.2 Scripts and code

The steps presented above almost map to a script or function written. The flow of the dataset generation is as described here.

1. The SLURM script `main.sh` is submitted on the cluster. It fetches relevant data in the `config.py` script.
2. It submits the generation of the reference simulation as another job on the cluster and waits for its completion.
3. It calls `sampling.py` with argument `--sample-only`, that will create the `samples.csv` file containing all the samples.
4. It prepares the file `dataset.csv` by writing its header line.
5. It finally runs the bash script `launch-job-series.sh`, that will submit some fixed number of sample jobs, as an array, through the SLURM script `launch-simulation-jobs.sh`.
6. Each sample job runs:
   (a) The simulation directory is prepared by calling the `sampling.py` script with argument `--prepare-one` and the index of this simulation (it reads the corresponding line in `samples.csv`).

(b) GAMS is called on the simulation directory.

(c) The simulation results are read with `read_results.py --single`, then outputted to `dataset.csv`.

(d) The simulation directory is cleaned.

3.6.3 Technical aspects

During the creation of this set of scripts, some technical details require specific attention:

– The GAMS solver spawns threads for efficiency, the number of threads must not exceed the number of CPUs allocated by SLURM for that job, otherwise it will disturb the whole node on which it is running, including unrelated jobs.

– The total amount of each prepared simulation directory is too large to fit on the allocated disk memory on the cluster.

– The Dispa-SET ajdusting function do not write adjusted data to a directory if this directory already exists before the function is called

Should I write more about the "history" of the scripts ?

3.6.4 Dataset fields

For completeness, all the fields in the created dataset in `dataset.csv` are shown on Table **??**.

Parameter	Unit	Parameter	Unit
Cost	€/MWh	CF nuc	[·]
Congestion	h	CF wat	[·]
PeakLoad	MW	CF win	[·]
MaxCurtailment	MW	CF sun	[·]
MaxLoadShedding	MW	Capacity ratio	[·]
Demand	TWh	Share flex	[·]
NetImports	TWh	Share sto	[·]
Curtailment	TWh	Share Wind	[·]
ENS	TWh	Share PV	[·]
CF gas	[·]	rNTC	[·]

Table 3.6: Dataset fields

4 Surrogate model training

4.1 Overview

This section presents the training process that lead to the description and definition of the target surrogate model, given a ready to use dataset, and the desired characteristics.

This is a straightforward example of a regression problems, that is to predict some output features from the input features, given a number of training example, that is the dataset. This is a typical machine learning problem, and will be tackled as such.

4.2 Machine learning methods

In the following, some machine learning algorithms for regression problems [3] are assessed.

4.2.1 K nearest neighbors

In this setting, the K-nearest neighbors (k-NN) methods would be one of the easiest to implement[1]. Indeed, these simply require a single parameter value k and output the average value of the output features of the k nearest neighbors, computing a distance on the input features.

This will come with the drawback of not being able to learn quick variations in the outputs. What is considered too problematic, as these critical points would need to be properly modelled for the reliability of the surrogate model.

4.2.2 Decision trees

Decision trees, and extremely randomized trees [8] another category of easy to implement[2] machine learning methods, as they also come with a limited, fixed number design parameters.

But these also come with the drawback of having a piecewise constant output, and as stated above, it is required to be able to represent significantly fast variations in the output. Although it is possible to mimic with many successive steps, being peicewise constant will now introduce non linearities in the outputs, that is, unwanted steps in the prediction.

4.2.3 Kernel-based methods

One may consider to apply a kernel method with one of the other method mentioned above. While this may indeed improve the quality of the resulting model, and also solve the issue that was representing fast changes in the output, there is a main, almost trivial issue with them. Obviously, one need a kernel, but what kernel?

As there are no ready-to-use kernel for this specific case, and that finding such a kernel would be a hard task, kernel methods are abandonned.

4.2.4 Artificial neural network

Artificial neural networks (ANN) are used in this work to build our surrogate model. These kind of methods provide a large flexibility, due to their entirely customizable architecture, as well

[1] Using for example scikit-learn[9] module in python

[2] Again, using scikit-learn

as a large learning capacity. This makes them able to modelize with good accuracy some complex, non-linear functions.

In most recent applications of these ANNs, a lot of different strategies are used to process the data efficiently. For example, convolutional layers convey a lot of meaning in the context of image processing, or transformers are well suited to process sequences[5].

In this case, the inputs boils down to the 6 variables values, listed in Table table:reference-values. There are no patterns in this data, because even if their actual values were correlated in some way, the simulations dataset we have as an input at this stages has its input features drawn from a latin hypercube sampling, meaning they have a fixed, very low correlation. This correlation originates from the fact that the sampling aims at optimizing the design space coverage, not from a meaningful, exploitable source.

Thus, a simple multi-layer perceptron (MLP) architecture is chosen, and the next requirement is to describe the characteristics of that MLP, that are:

- The number of layers
- The numbers of neurons in each layers
- The activation function at each layer

These hyper-parameters values will be properly set in the following.

4.2.5 Overfitting and underfitting

An inherent problem to machine learning methods is the overfitting, or its opposite, underfitting. These terms refer to the cases where the training is respectively too specific to the training data, and not enough specific.

Overfitting is thus a symptom of too much learning, leading to learning some noise or some particularities of the dataset, while underfitting means that there is not enough learning, so that the model is not able to represent all the cases, even the one that are well represented in the available data.

These will have to be assessed during training to ensure the validity of the model.

4.2.6 Bias

An other source of imprecision in machine learning is the bias. This relates to the fact that there exist some noise in the data, that cannot be filtered out, or imprecisions in the assumptions, that inevitably conducts to noise in the output.

However, in this setting, there is very little one could do to reduce its significance. First, the data points have been drawn from a latin-hypercube sampling strategy, that precisely aims at spreading the samples equitably all over the input space. Then, the output features were computed from a Dispa-SET run on this sample.

Thus, the main source of bias one may have an influence in is the bias originating from incorrectnesses during the model training, due to a poor model design.

The other plausible source of bias is the simulation made in Dispa-SET. Of course, Dispa-SET is also itself a model, thus relying on some assumptions and subject to its own modelling of the reality. And as such, it may introduce a bias in its computations, that will necessarily be learned by the surrogate model. But there is no way to assess this bias, and obviouly Dispa-SET itself focuses on making that bias as negligible as possible.

This consideration is of interest, as Dispa-SET has multiple formulations, namely LP and MILP, that then have different bias with respect to reality.

4.3

References

[1] GAMS Development Corporation. *General Algebraic Modeling System (GAMS) Release 24.5.6*. 2015. URL: https://www.gams.com/.

[2] pyDOE documentation. *Randomized designs*. Accessed in May 2023. URL: https://pythonhosted.org/pyDOE/randomized.html#latin-hypercube.

[3] P. Geurts and L. Wehenkel. "Introduction to Machine Learning". ELEN062-1 class. 2021.

[4] K. Kavvadias et al. "Integrated modelling of future EU power and heat systems: The DispaSET v2.2 open-source model". In: *JRC Technical Report, EU Commission* (2018).

[5] G. Louppe. "Deep learning". INFO 8010 class. 2022.

[6] mit.edu. *Optimal Latin Hypercube Technique*. Accessed in May 2023. URL: https://abaqus-docs.mit.edu/2017/English/IhrComponentMap/ihr-c-Reference-OptimalLatin.htm.

[7] Tom O'Malley et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.

[8] D. Ernst P. Geurts and L. Wehenkel. "Extremely randomized trees". In: *Machine learning* (2006). URL: https://doi.org/10.1007/s10994-006-6226-1.

[9] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[10] *Power System Flexibility for the Energy Transition, Part 1: Overview for policy makers*. Abu Dhabi: International Renewable Energy Agency, 2018. URL: https://www.irena.org/publications/2018/Nov/Power-system-flexibility-for-the-energy-transition.

[11] Mandar N. Thombre, Heinz A. Preisig, and Misganaw B. Addis. "Developing Surrogate Models via Computer Based Experiments". In: *12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering*. Ed. by Krist V. Gernaey, Jakob K. Huusom, and Rafiqul Gani. Vol. 37. Computer Aided Chemical Engineering. Elsevier, 2015, pp. 641–646. DOI: https://doi.org/10.1016/B978-0-444-63578-5.50102-X. URL: https://www.sciencedirect.com/science/article/pii/B978044463578550102X.

[12] Inc. Ventana Systems. *Vensim software*. Accessed in May 2023. URL: https://vensim.com/vensim-software/.

[13] Carla Vidal Montesinos. *Integrating short-term dispatch constraints in a system dynamics energy planning model*. 2022. URL: http://hdl.handle.net/2268.2/16566.

[14] wikipedia. *Latin hypercube sampling*. Accessed in May 2023. URL: https://en.wikipedia.org/wiki/Latin_hypercube_sampling.

[15] wikipedia. *Vensim*. Accessed in May 2023. URL: https://en.wikipedia.org/wiki/Vensim.