University of Liège - School of Engineering and Computer Science

MASTER'S THESIS

Improving the simulation of variable renewable energy in the MEDEAS integrated assessment model

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

Academic year 2022-2023

Acknowledgements

I would like to thank professor Quoilin first, for his mentoring during this thesis and his advices. Then Jade, for her collaboration on the linkings between the created model and MEDEAS.

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. I am also grateful to the cluster administrator, M. Collignon, for his patience and valuable advices.

The use of the popular tool ChatGPT [1] is also to be mentioned, although primarily taken advantage of for proof-reading and suggestions.

Thanks should also go to my mother, and everyone to which I talked about my thesis, what sometimes led me to get a new perspective on the work as I had to explain it in a new, concise and understandable way.

Abstract

This master thesis explores the application of a surrogate model to achieve the integration of electricity network constraints into integrated assessment models, that attempt to predict the evolution of the main features of the society, including economic, demographic, climatic and energy factors.

Tools were developped to run Dispa-SET, a dispatch model for the European electricity network, to build a dataset of simulations on adequate sample points. This dataset is then used to train the surrogate model employing appropriate machine learning methods. Finally, this model is integrated into the MEDEAS IAM, connecting their variables in a meaningful way, and the resulting model is run for different scenarios.

Results show that integrating the model in MEDEAS leads to a lower prediction of the production of variable renewable energy sources compared to the initial MEDEAS output.

Keywords: surrogate model, Dispa-SET, MEDEAS, machine learning, VRES

Contents

Keywords

AF	Availability factor
ANN	Artificial neural network
BAU	Business as ususal
CF	Capacity factor
CÉCI	Consortium des Équipements de Calcul Intensif
DLL	Dynamically loaded library
ENS	Energy not served
EROI	Energy return on investment
EU	European union
GAMS	General algebraic modeling language
IAM	Integrated assessment model
LP	Linear programming
LHS	Latin hypercube sampling
MILP	Mixed integer linear programming
MLT	Mid-level transition
MLP	Multi-layer perceptron
MTS	Mid-term scheduling
NTC	Net transfer capacity
OLT	Optimal level transition
P2H	Power to heat
PHS	Pumped hydro-storage
PV	Photovoltaic
RES	Renewable energy sources
SLURM	Simple utility for Resource Management
VRES	Variable renewable energy sources

1 Introduction

Our societies are fueled by energy. Historically, fossil fuels offered a relatively easy to access, store and use energy for decades, bringing a solution to this energy issue.

The rising awareness of the climate change, mostly caused by the release of large amounts of greenhouse gases conducts to a paradigm shift. To meet climatic goals, strict limits on emissions have been established, consequently limiting the prevously endless source of energy that were fossil fuels over the long term [2]. This directly impacts the solutions depending on their combustion, releasing massive amounts of carbon dioxide.

The solutions to compensate for the lacking energy generation, that from now on should not derive from fossil fuels, are renewable energy sources. These refer to every energy generation technique originating from a renewable source, such as the sun, the wind and the rivers. However, the production of these units is not systematically sustainable. For example, the photovoltaic cells necessary for the exploitation of the incoming solar energy are pretty difficult to recycle, making them rely on specific materials that are not obtainable renewably. Still, their use on a complete lifetime, and increasing capabilities in recycling, justifies the investment in their construction.

In this context, a meaningful increase in the electricity produced from such energy sources is to be expected, particularly from the most prominent ones:

- the sun, through photovoltaic panels,
- the wind, through on-shore and off-shore wind turbines,
- rivers, through hydroelectric dams,
- biomass, through adequate units, and
- geothermal energy, through geothermal power stations.

The third one, due to its dependency on the geographic context, will however not expand foerever, as there are not illimited spots to build such dams. In this work, the biomass and geothermal energy sources, less common for the time being, are not considered.

One thus falls back to photovoltaic and wind energy, but both have a major, trivial drawback: they rely on the sun and the wind, respectively. And this is a significant concern, because the amount of energy that can be generated by exploiting these is variable, hence their designation as variable renewable energy sources, or VRES.

That variability does not necessarily involve poor predictability, for example, there is on average more photovoltaic production potential during summer. On a daily scale as well, with the day night cycle. Weather forecasts can be taken advantage of in order to predict wind turbines' production.

The idea of residual load, the difference between the actual load and the amount of energy that can be provided from renewable sources. This residual load fluctuates, therefore needing conventional electricity units to be dispatched in real time. But since these units have start-up and shut-down costs, minimizing these remains attractive. Hence, flattening the residual load curve is desired, what governs the dispatch the units available on the network.

1.1 Flexibility assessment in energy system models

As explained before, higher shares of VRES in the electricity production mix create the additional challenge, that is the handling of the partlially predictable variability of wind and sun energy.

This handling requires a larger flexibility of the electrical system, that is, a better ability to adapt to both anticipated and unforeseen changes in the demand and supply [3].

Existing mechanisms to improve the flexibility of a power system include:

- Use of the regular dispatchable energy production plants to compensate for the energy deficit that may arise from VRES. Plant characteristics play a role to address short term drops in production, as some start up time might be required [4].
- Large interconnected electricity networks, that are able to smooth the VRES power output. There may be not enough sun in some region, creating a deficit, while there is too much in the neighboring country. By connecting them, the overproduction will compensate the unerproduction of the other [5].
- Energy storage facilities. Of course, storing the produced energy for later use is an easy way to account for the intermittency of the production. Typically, storing solar energy during day time to be used in the night. These technologies include pumped hydro-storage, batteries, compressed air. While pumped hydro-storage units are the most common, their very limited expansion options make them unlikely to grow in the future [6].
- Acting on the demand, in the extent that it can change its shape by promoting policies to the end users. Such policies focus on flattening the daily demand curve, facilitating the energy production dispatch. Typically, asking to delay greedy devices like dishwashers until night, where the overall demand is lower. But this could extend to other domains, such as electric vehicles charge, heating and cooling etc [6].

The two main consequences of insufficiently flexible energy systems are curtailment, when there is too much energy produced, and load shedding, when there is not enough electricity to satisfy the demand. In case of load shedding, parts of the grid may be entirely shut down.

1.2 Short-term dispatch models

There exist tools built in order to assess the behaviour of large electrical systems, that are subject to higher share of VRES. These tools typically aims at predicting the electricity flows, dispatching available power plants in order to match the production to the demand.

For these purposes, such tools typically set low time steps, e.g. 1 hour, and their simulation period spans up to one year [7]. This level of granularity is required in order to capture sufficiently well the variations in the availability of variable RES. In association, significant levels of details are modelled, like simulating every existing units.

Their scope might range from the electrical system only, to the entire energy system, hence encompassing for example heating, transportation, and industry matters. Some also include economical considerations [8].

And from there on, some higher level metrics can be computed, and in particular, we will be interested by:

– the curtailment, that is, the energy produced in excess while the electricty demand was already met, that end up wasted, and
– the lost load, that is, the energy that could not be produced, hence some demand could not be served.

The Antares simulator, PLEXOS and MAON models are examples of such tools. Among them, the Dispa-SET model [9] is selected for this work. Dispa-SET is open-source, and focused on balancing problem in the European grid specifically.

This model is formulated in linear programming, that is, a set of linear constraints are defined and an objective function is given. The solver inputs both of these and computes the parameters values that maximize the objective function while matching the constraints.

1.3 Integrated assessment models

On another level, integrated assessment models (IAM) aim at estimating the evolution of large, intricated systems involving a lot of different interconnected areas and actors. These are often multidisciplinary and require a lot of modelling choices.

In particular, some IAM attempt to model the evolution of the whole society, from a socio-economical perspective, including environmental aspects and energy concerns. MESSAGEix, GCAM and MAgPIE are examples of such models that have been used in the IPCC reports [10] [11] [8] [12].

Within this category of IAMs, the MEDEAS model [13] is selected for this work, being open-source as well and providing a specific european model.

MEDEAS is expressed in terms of systems dynamics, that is, the evolution of the state of the simulation is computed as a function of its current state. And this involves solving a set of differential equations.

1.4 Model linking

Due to computational constraints, IAMs often have a pretty low level of temporal, or spacial accuracy [14]. This is not the case for dispatch models, that carry out more extensive simulations. Therefore, establishing a link between two of these models is interesting, as the IAM would benefit from the better accuracy of the energy models.

An high-level illustration of the position of MEDEAS and Dispa-SET on the timescale is provided in Figure 1.1 [9].

1.4.1 Linking types

There are several strategies that may be used in order to link two models together [14].

– Soft linking: the models communicate between each other. This communication may be uni-directional or bi-directional. Both of the models are run iteratively, thus keeping their separate efficiencies in the same order of magnitude. However, the iteration lead to low overall speed, and convergence is not guaranteed.
– Hard linking: the models are combined into a single, unified mathematical formulation. This newly created model can then be solved all at once. This approach is burdened by higher

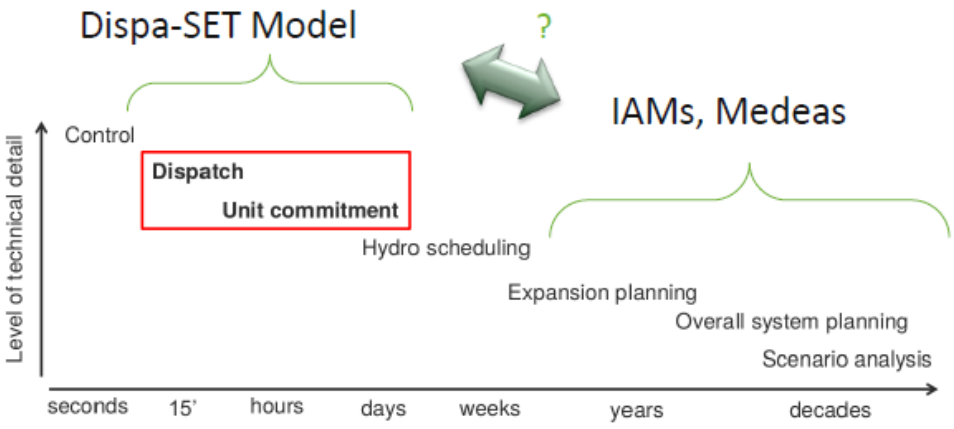


Figure 1.1: An illustration of where Dispa-SET and MEDEAS operate on the timescale

computational costs and lower chance of feasibility.

These linking types are illustrated in Figure 1.2.

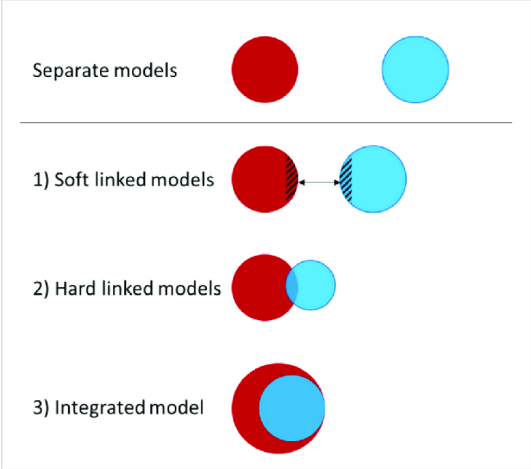


Figure 1.2: Illustration of the different linking methods [15].

One may also add model integration, consisting in completely embedding a model into the other. But this is not tractable in this setting, and would also requires compatible model formulations, as explained below.

In this case, hard linking is not possible owing to the different formulations of Dispa-SET (linear programming), and MEDEAS (system dynamics).

We also dismiss the soft linking strategy because of its slowness, and absence of convergence guarantee.

1.4.2 Literature review

Considering the effects of VRES in long-term integrated assessment models is a challenge since they don't have a sufficient time resolution to capture the rapid variations within the system.

Similar work has been done by Parrado-Hernando et al. in [16], that aims at "capturing features of hourly-resolution energy models". The methodology used to acquire the data presents two downsides that are addressed in this work.

First, the inputs of the energy model are handled as discrete variable, and simulation have been run using all possible combination for these values. This can be seen in some of their figures, where the data points seem to follow some lines. While it does not invalidate the process, this creates a bias in the repartition of the data. Second, linear approximation are used to fit the data obtained. Admittedly, this is identified as a limitation of the work.

To palliate these, the input space will be tackled as a continuous domain for the design of experiment, and other machine learning technique will be considered as to candidates for the creation of the surrogate model.

1.4.3 Surrogate models

The linking technique explored in this work is the surrogate model. In this case, an approximation of the Dispa-SET dispatch model will be integrated into the MEDEAS IAM.

The idea behind surrogate models is simple. First, a fast, convenient approximation of the model is made, then it its completely integrated in the other model. In this case, the relevant outputs of the Dispa-SET model will be approximated from relevant inputs regarding MEDEAS, then the approximator will be integrated in the model.

The soft-linking approach has already been explored [17], [18], and the hard-linking remains the hardest to investigate, notably due to incompatible formulations. Surrogate models are still unexplored and are of great interest for this use case.

As the combination of the model will use an approximation, this appriximation being fast will not burden the computations of the IAM, hence keeping it efficient.

Surrogate models then provide a promising strategy to tackle this integration problem.

1.5 This work

1.5.1 Objective

This master's thesis is dedicated to the integration of the flexibility constraints, the Dispa-SET surrogate model, into the MEDEAS model, with reduced details on this aspect. This includes the creation of a proper dataset, the definition, training and integration of the surrogate model into MEDEAS.

It extends previous work as such an approach, as discussed above, has not yet been implemented in this context.

1.5.2 Contributions

This work follows what was started by another student, Carla Vidal, that went until the surrogate model training, included. Given that improvements where implemented in Dispa-SET since then, the runs had to be re-done. However, there were no easy to use scripts to set up the simulation files etc., so that is has been chosen to write new ones.

Additionally, the present thesis also explores other machine learning algorithms, although the same choice of neural networks is made, this time based on better performance compared to the other options.

Another student, Jade Paris, was in responsible of the linking of the surrogate model, given as a functions of some inputs variables into the MEDEAS model, and its actual use and runs of MEDEAS with the surrogate model integrated.

This work constisted in:

- The writing of scripts to run Dispa-SET on those experiments
- The choice, definition and implementation of an adequate machine learning model, and its training
- The integration the model in MEDEAS, by writing a C++ external function library for Vensim, or by inserting it into the PySD model of MEDEAS directly in python
- Runs and analysis of the improved MEDEAS model

The global workflow is depicted in Figure 1.3.

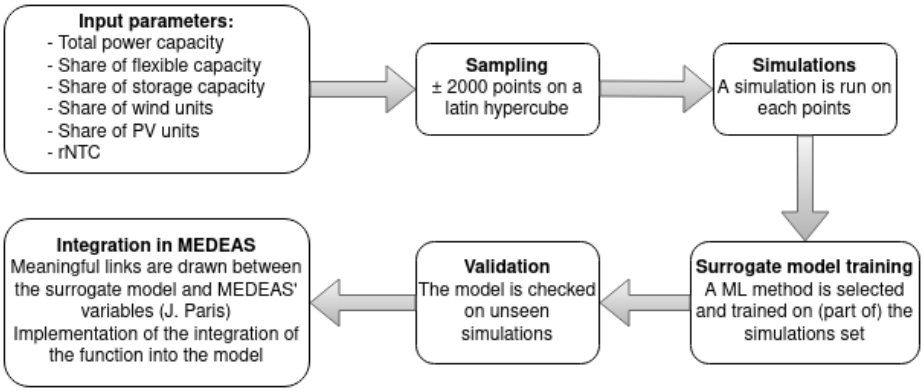


Figure 1.3: Worflow of the master thesis

All the produced work, and necessary data is available in the online github repository at the following address: https://github.com/Rayerdyne/master-thesis.

1.5.3 Outline

This document is organized into seven main sections. The second section presents an in-depth description of the Dispa-SET model, along with the tools employed within its framework. Following this, the third section provides a detailed account of the MEDEAS model, outlining its underlying principles and the broader framework in which it operates.

To facilitate the successful integration of models, the fourth section outlines the process of data generation. This encompasses a complete overview of the methodologies employed to produce the necessary training data, the design of experiments and the execution of Dispa-SET runs.

Subsequently, the fifth section describes the surrogate model, offering a comprehensive description of its design. The section further expounds on the training and validation processes undertaken to ensure accurate alignment of the surrogate model with the main models' outcomes.

The sixth section then shifts the focus to the crucial process of integrating the surrogate model into MEDEAS. A step-by-step description is provided, elucidating how the surrogate model becomes an integral component of the broader MEDEAS framework, and how it interacts with the existing models.

In the seventh, the document offering a comprehensive summary of the resulting model and outcomes. This section also provides interpretations of the observed results.

Finally, the last section concludes, reflecting on the work and identifying any limitations and outlining future areas of research for further enhancements.

2 The Dispa-SET model

2.1 Overview

The Dispa-SET model [9] is described in the source [19] 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. As such, 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 displayed in Figure 2.1.

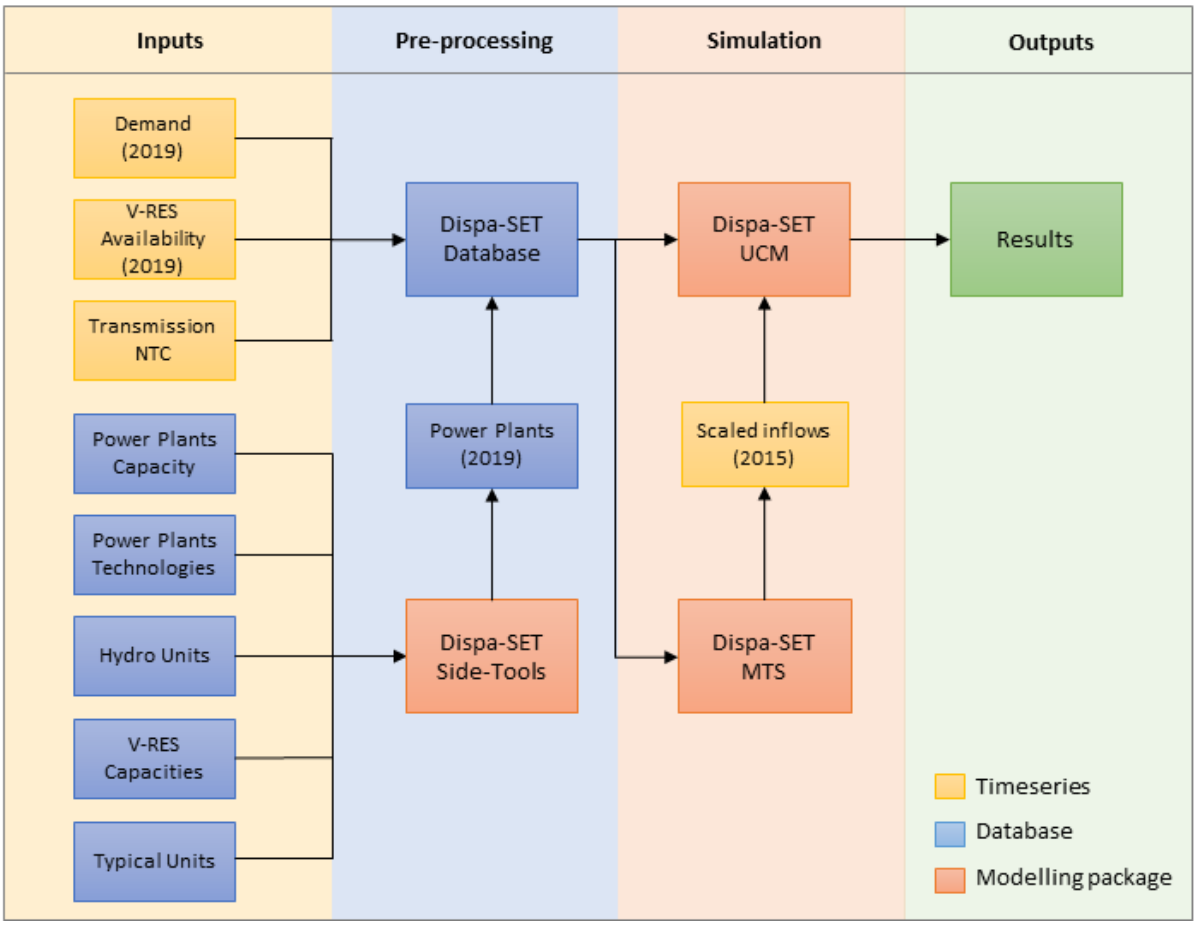


Figure 2.1: Block diagram of the architecture of the Dispa-SET model

Its interface is written in the Python programming language, and calls GAMS [20] 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 in this context. These costs typically include transportation, power and heating costs required to split efficiently the demand between the available generation units.

Symbol	Meaning
$Cost[StartUp\|ShutDown]_u$	Cost of the start up or shut down of unit u
$CostRamp[Up\|Down]_u$	Cost of the ramping up or down of unit u
$Cost[Fixed\|Variable]_u$	Fixed or variable cost of operating unit u
$Commited_u$	1 if unit u is turned on, else 0
$Power_u$	The power unit u operates at
$TimeStep$	The duration of one simulation time step
$CostLoadShedding_n$	Cost of the load shedding in zone n
$ShedLoad_n$	The amount of load being shed in zone n
$LostLoad$	Lost load, i.e., load that is neither produced, nor accounted as load shedding
$VOLL$	Value of lost load, i.e., its price per MWh
$CostOfSpillage$	Cost of spillage in storage units
$Spillage_s$	Amount of spillage in storage unit s
$Location_{u,n}$	Boolean, 1 if unit u is in zone n, else 0
$Flow_{l,n}$	The electric flow between zones l and n
$Demand_n$	The demand in zone n
$StorageInput_s$	The power being inputted into the storage unit s

Table 2.1: Description of the variables used in Equations 2.1 and 2.2

The total system costs is split 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 increase or decrease in power output of a unit.
- *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.

We can formulate, Equation 2.1 to represent the objective function, where u refers to the index on each units, i is the time index, and n the zone index.

Table 2.1 summarizes all the names appearing in the equations.

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

$$
\begin{aligned}
Min_{TotalSystemCost} = \\
&\sum_{u,i}(CostStartUp_{u,i} + CostShutDown_{u,i}) + \\
&\sum_{u,i}(CostRampUp_{u,i} + CostRampDown_{u,i}) + \\
&\sum_{u,i} CostFixed_u \cdot Comitted_{u,i} \cdot TimeStep + \\
&\sum_{u,i} CostVariable_{u,i} \cdot Power_{u,i} \cdot TimeStep + \\
&\sum_{n,i} CostLoadShedding_{n,i} \cdot ShedLoad_{n,i} \cdot TimeStep + \\
&\sum_{n} VOLL \cdot LostLoad \cdot TimeStep \\
&\sum_{s,i} CostOfSpillage \cdot Spillage_{s,i}
\end{aligned}
\tag{2.1}
$$

Equation 2.1: Objective function of the Dispa-SET model

– 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 (power to heat) units

Equation 2.2 expresses this target energy production and consumption balance.

$$
\begin{aligned}
&\sum_{u}(Power_{u,i} \cdot Location_{u,n}) + \sum_{l} Flow_{l,n,i} \\
&= Demand_{n,i} + \sum_{s}(StorageInput_{s,i} \cdot Location_{s,n}) + \\
&- ShedLoad_{n,i} - LostLoad
\end{aligned}
\tag{2.2}
$$

Equation 2.2: Supply-demand balance in Dispa-SET

2.4 Rolling horizon

The ideal solution would be to solve the entire system, for every time step in the complete duration of the simulation in one go. But this would create a system too large to be efficiently solved.

To address this, the UCM simulation is split into smaller, tractable parts. The simulation is built for smaller time frames, called optimization periods, over which the simulations can be made easily.

The start of optimization period j overlaps the optimization period $j-1$, to that the simulation j is the correct prolongation of the same setting fixed by the simulations up to $j-1$. The period that overlaps is called the look-ahead, in which the values of the parameters for period $j-1$ is determined during simulation $j-1$, and are used as fixed context for simulation j. A depiction of rolling horizon is given in Figure 2.2.

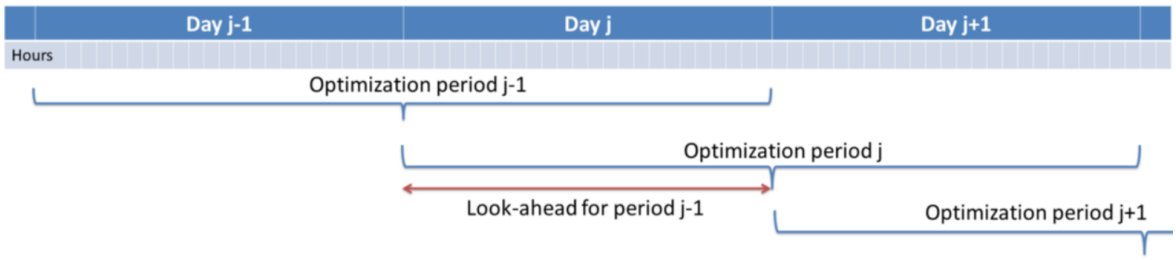


Figure 2.2: Depiction of the rolling horizon mechanism

2.5 Mid-term scheduling

Without further constraints, the optimization will most often leave all storage facilities empty at the end of the simulation horizon (typically a few days). This is a consequence of the variable operational cost of discharging these storage units being smaller than the cost of running another unit, thus charging extra fixed and variable costs.

To address this issue, the Dispa-SET model has to be run in Mid-Term Scheduling (MTS) mode. In this mode, the initial and final levels of the storage units (in particular, pumped hydro storage units) are given as exogenous input to the model. These levels are enforced with additional constraints.

However, this options has additional requirements. The formulation has to be set to LP, the rolling horizon is turned off and the time resolution is risen to one day.

In this work, the MTS mode is enabled, and the exogenous inputs, for both initial and final levels, are set to half of the storage capacity.

2.6 Problem formulations

Dispa-SET features several, different formulations, with a significant impact on the realism and accuracy of the output. These formulation differ in the way the simulation is created and the constraints defined.

2.6.1 Linear programming

In the LP formulation, every variable is considered to be continuous, and can take values continuously down to zero. In particular, this applies to the *Commited* variable, hence enabling any power output from 0 to the unit's maximum power.

However, the LP formulation lacks the capacity to enforce minimum operating levels for units effectively, leading to potentially unrealistic scenarios, for example a nuclear unit operating at 10% of its maximal power, what is not achievable.

2.6.2 Binary formulation

To mitigate this issue, the *Commited* variable is made boolean. That way, the minimum operating value can be enforced when the corresponding commited boolean is set to one.

This strategy, however achieving the better accuracy targetted, is unfortunately often intractable in realistic settings, due the large number of binary variables (one for each unit) yielding an exploding number (2^N) of options to be examinated.

2.6.3 Mixed integer linear programming

The MILP formulation is created to leverage the computational cost of the simulation while keeping a good level of accuracy. The main idea is to group the units that share similar properties together, and only keep track of the number of units in each group that are currently commited.

This permits the number of options to explore reasonnable, without hurting the precision of the output.

In this work, the MILP formulation is chosen due to its better efficiency-to-computational-cost trade-off.

2.7 Reference simulation

This work takes the simulation over the year 2019 as a reference. Accordingly, the several input timeseries, including the demand, the availability factors, and the flows between each zones are used.

Dispa-SET also detects congestion issues. These appear when a link transmits too much power for its capacity. For electric wires, congestion causes a wire to retain heat, what may threaten the wire integrity. While seemingly harmless, this phenomenon is likely to be on the rise, for example when an area produces a lot of renewable energy, and has to transmit it to some neighbour in deficit.

The annual dispatch plot generated by Dispa-SET for the reference simulation in Belgium is displayed in Figure 2.3.

The observations drawn from Figure 2.3 are as follows:

- The units "BE_STUR_NUC" and "BE_STUR_BIO_CHP" are almost always on. These correspond to turbines powered by nuclear energy and bio fuels respectively. The latter is a combined heat and power unit.
- Wind turbines, both on-shore and off-shore, have roughly the same availabilities.

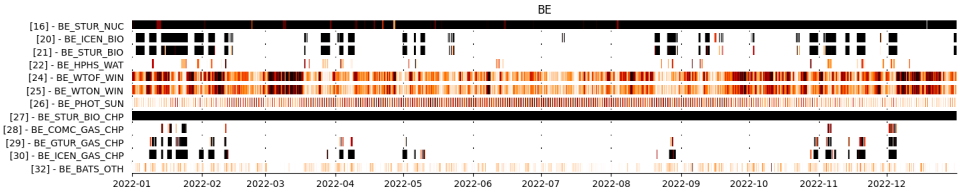


Figure 2.3: Dispa-SET reference simulation. Commited units over one year in Belgium

– The higher availability in PV energy during summer is apparent, along with the day-night cycle.

– Other units like "BE_ICEN_BIO", "BE_STUR_BIO", "BE_GTUR_GAS_CHP" and "BE_ICEN_GAS_CHP" seem to be turned on while wind units generate less power, especially during winter.

Figure 2.4 presents a dispatch plot over a week in Germany.

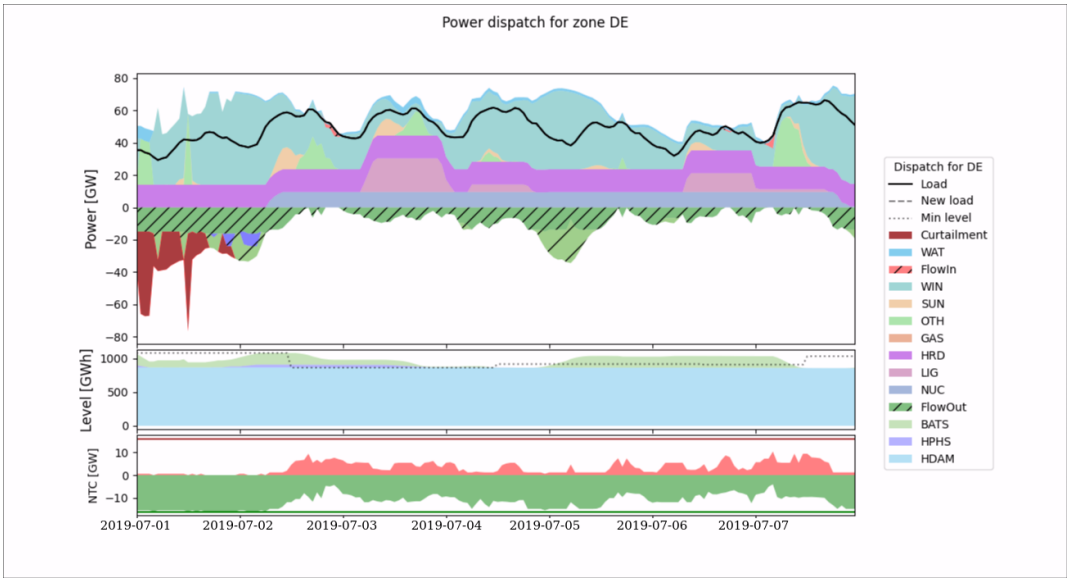


Figure 2.4: Dispa-SET reference simulation. Dispatch plot over a week in Germany

In Figure 2.4 we first observe some curtailment in the first day, indicated by the red area. The green striped region indicates that the zone was exporting excess production during the week. This is aligned with the depicted NTC, presenting a greater flow outwards than inwards. Moreover, the reservoirs of storage units were predominantly full. As the demand was entirely met by the production, no load shedding appeared. In the latter case, the load curve would be higher than the production and importations sum.

3 The MEDEAS model

3.1 Overview

This sections aims to describe the main components of the MEDEAS Integrated Assessment Model (IAM) and its components, with a focus on the parts that are the most relevant in this context.

Named "Modelling the Energy Development under Environmental And Socioeconomic constraints", the project endeavors to construct a modern computational model to predict the future of the energy systems in Europe, while integrating a wide range of physical and social constraints.

First, the IAMs are depicted, then a high-level description of the MEDEAS model is provided. The third subsection will explain more in depth in a key component of MEDEAS, the energy return on investment. In the fourth subsection, the modelling of the RES in MEDEAS is covered. Finally, the principal scenarios in MEDEAS are presented in the fifth subsection.

3.2 The MEDEAS integrated assessment models

MEDEAS is an open-source IAM, built to "guide the transition to a low carbon European socio-economy" [21].

Integrated assessment models are used to make general purpose analysis, amalgamating insights from diverse domains, such as economy, environment and energy, land use etc. These kinds of models, once properly defined from a mathematical point of view, can then be simulated by computer, for their result to be analysed.

Given the considerable uncertainty and intricate interdependencies among model components, there exists a multitude of IAMs with varying methodologies.

As an additional point of consideration, the model being open-source is probably one strength as an IAM, meaning that any expert in one domain may be able to contribute to the project.

As previously quoted from their website, the MEDEAS model has been built with the purpose of guiding decarbonation in Europe. It has been designed to compensate for the flaws of other available IAMs, in order to inform policy makers towards a transition to more carbon-independent, sustainable energy.

MEDEAS is built using the Vensim software, and can be used from the Python programming language through the `pySD` package.

3.3 Model overview

MEDEAS, funded by the EU's Horizon 2020 program, under the "Modelling and analysing the energy system, its transformation and impacts (social, environmental and economic aspects of the energy system)".

And to do so it models the long-term implications of the decisions made by the society. As one cannot predict them, several hypothesis are needed to fix the choices that will be made. Such a set of hypothesis on the evolution of the long-term policy of the society as a whole is called a scenario.

MEDEAS typically sets the simulation horizon between 1995 and 2060, though for longer term analysis it might be raised up to 2100. It also features different settings:

- MEDEAS-W, the global one,
- MEDEAS-EU, targeting the European Union,
- MEDEAS-AU and MEDEAS-BG, targetting Austria and Bulgaria respectively.

The MEDEAS IAM is organized into seven modules, encompassing economy, energy, energy infrastructures, materials, land use, climate change and socio-environmental impacts indicators. The general structure is illustrated in Figure 3.1.

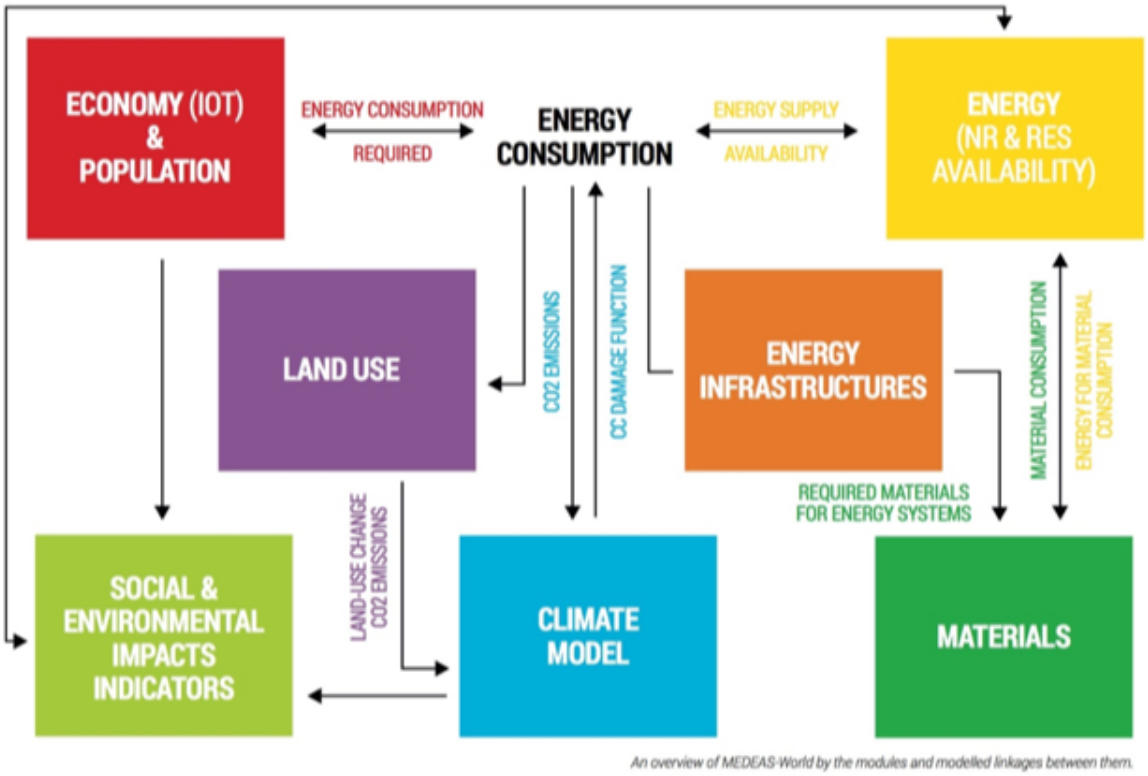


Figure 3.1: The MEDEAS IAM modules

In this work, the MEDEAS-EU version are considered, since it has the same geographical scope than the Dispa-SET model.

3.3.1 System dynamics

Formulated using system dynamics, the MEDEAS model leverages Vensim software to aggregate insights from diverse experts and domains. This also enable easy modelling of feedback between the different components.

System dynamics is a modeling language, made to obtain understandings of complex and dynamic systems. It achieves using nodes and arrows, corresponding to values and relations between those respectively. The former can be either a stock, a flow or a constant, while the latter makes the link through equations. It is therefore possible to create feedback loops and complex interconnected networks.

System dynamics have been popularized by the *world3* model, used to predict the consequences of long-term policies applied by the society on the planet in the Limits to Growth report [22].

This approach is evidently far from the linear programming paradigm, that has a too different mathematical formulation for both to be combined.

3.4 Energy return on investment

When having energy consideration in the long term, the energy return on investment (EROI) assumes a crucial role.

EROI signifies the exploitable energy obtained from a resource to the amount of exploitable energy spent to acquire that resource [23].

The EROI serves as a critical metric for the assessment of the energy sources efficiency, providing a measure of how efficient this energy source is to make use of. On most cases, the EROI of RES is lower, meaning a lower energy gain, than fossil fuels'.

It's important to differentiate between EROI and net energy gain, where the latter represents the surplus energy available for public use. Obviously, its value should be larger than zero for the operation to be profitable.

Equations 3.1 and 3.2 summarize their definitions and the relationship between the two.

$$EROI = \frac{Energy_{returned}}{Energy_{invested}} \tag{3.1}$$

$$NetEnergy = Energy_{returned} - Energy_{invested} = Energy_{returned}\Big(1 - \frac{1}{EROI}\Big) \tag{3.2}$$

However, this also requires to set a definition on what exactly is the energy invested on the aquisition of an other resource. The MEDEAS model thus ovvers several EROI values relating to different approaches [24].

- Standard EROI, that "includes the direct (i.e. on site) and indirect (i.e. offsite energy needed to make the products used on site) energy requirements to get the energy (e.g. build, operate and maintain a power plant)" [24].
- Point of use EROI, that includes the energy cost of obtaining and transporting the fuel to the actual location where it will be used by society.
- Extended EROI, that " considers the energy required to get, deliver and use a unit of energy, i.e. the energy required to produce the machinery and devices used to build, operate and maintain a power plant or a transportation facility (tank truck, pipeline, etc.) as well as the energy required for exploration, investment, communication, labour, etc. in the energy system" [24].

In this context, we focus on the standard EROI.

In MEDEAS, the EROI is dynamically computed, meaning it is an endogenous variable, as the ratio of the exploitable energy delivered to consumers, over the sum of the total energy costs required for operating the plant and the energy costs of handling the variability of the power output

and the costs of operating the energy transportation network. The total costs for an operating plant include the building, maintenance and disposal costs.

On Figure 3.2, a depiction of a high-level energy flow is given. The exploitable energy delivered to consumers is the flow labelled (1), and the operating costs of the plants (2), whereas the costs accounting for the handling of variabilities is labelled (3) and for energy transportation (4). Equation 3.3, 3.4 and 3.5 below shows the MEDEAS' computation of the EROI based on these.

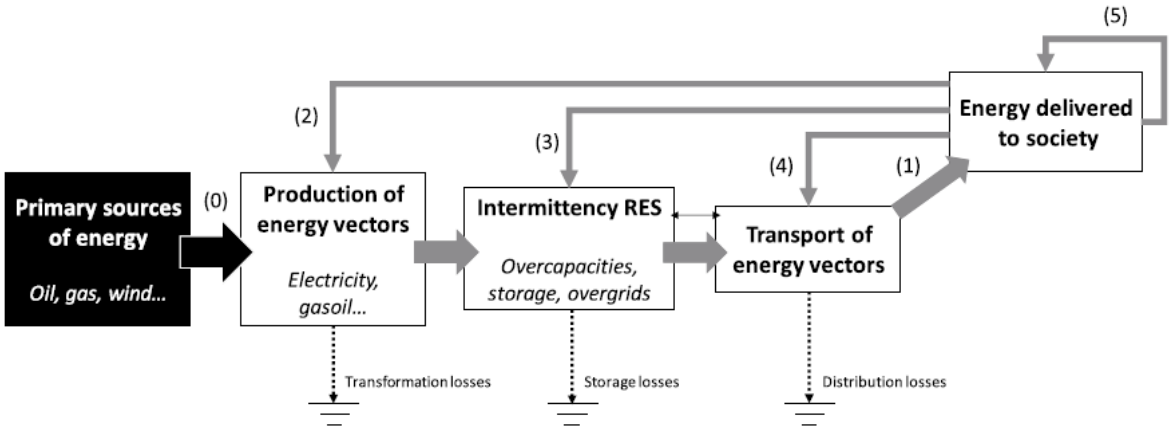


Figure 3.2: Representation of society's principal energy flows[25]

$$EROI_{st} = \frac{(1)}{(3)+(4)} \tag{3.3}$$

$$EROI_{pou} = \frac{(1)}{(2)+(3)+(4)} \tag{3.4}$$

$$EROI_{ext} = \frac{(1)}{(2)+(3)+(4)+(5)} \tag{3.5}$$

Furthermore, MEDEAS [25]:

- Assumes the EROI of non renewable energy sources to be constant over time,
- Dynamically estimates the EROI of RES producing electricity,
- Allocates technologies based on their EROI as a performance measure, meaning that higher EROI RES will be preferred,
- Computes overcapacities as a result of an increasing share of VRES endogenously,
- Takes additional losses into account for the use of storage units.

3.5 Modelling of RES

MEDEAS addresses the implications of variability in electricity production technologies, though not as comprehensively as Dispa-SET.

3.5.1 Limitation

First, it is recalled that the time step used by MEDEAS is coarse. Its recommended value is 0.03125, expressed as a fraction of a month. This amounts to approximately a day: $0.03125 \times \frac{365.25}{12} = 0.951$.

This value is clearly too high to model extensively daily variations of the power output of solar photovoltaic units, for example.

In the following, the mechanisms implemented by MEDEAS in order to incorporate the RES variability are described.

3.5.2 Grid extension

MEDEAS estimates, per MW of VRES, the additional electricity grid extensions required in order to incorporate those in the existing network. The materials needed for these expansions are also computed, what ultimately affects the EROI.

3.5.3 Storage units

In MEDEAS, the foremost storage technology used is the pumped hydro-storage (PHS). It contrasts with the Dispa-SET setting where batteries are the main storage technology. As already discussed, that is to because the european region is almost saturated in terms of PHS, as one could not find suitable location for new units to be built.

An estimation of the storage needs as a function of the VRES share is depicted in Figure 3.3 [26].

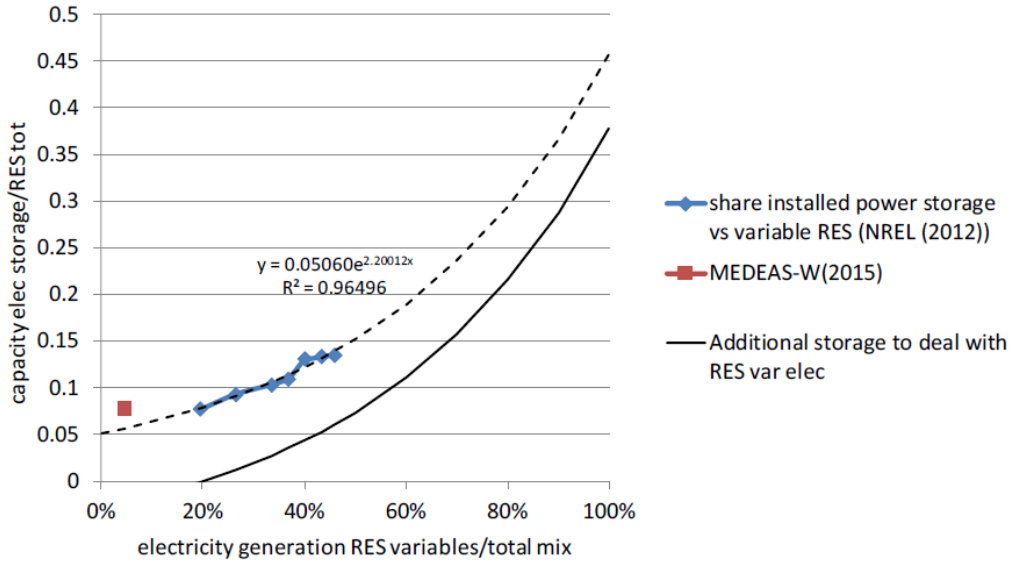


Figure 3.3: Energy storage capacity required as a function of the VRES share according to [26].

3.5.4 Dispatchable RES pants

The evolution of the capacity factor (CF) as the RES share among the electricity mix evolves is described. The capacity factor is the ratio of the electricity produced by a unit over a

period of time Δt over the maximum amount of energy that could have been produced. This is represented on Equation 3.6.

$$CF = \frac{Energy_{produced}}{\Delta t \times PowerCapacity} \tag{3.6}$$

The other meaningful metric in this context is the overcapacity, that is, the ratio of the energy that could have been produced, over the energy actually produced.

$$overcapacity = \frac{\Delta t \times PowerCapacity - Energy_{produced}}{Energy_{produced}} \tag{3.7}$$

An estimate of the overcapacity of dipachable RES is provided in [26], the resulting capacity factor evolution is depicted as a function of the VRES share in Figure 3.4. It can be observed that capacity factor decreases quadratically in the VRES share in the electricity mix.

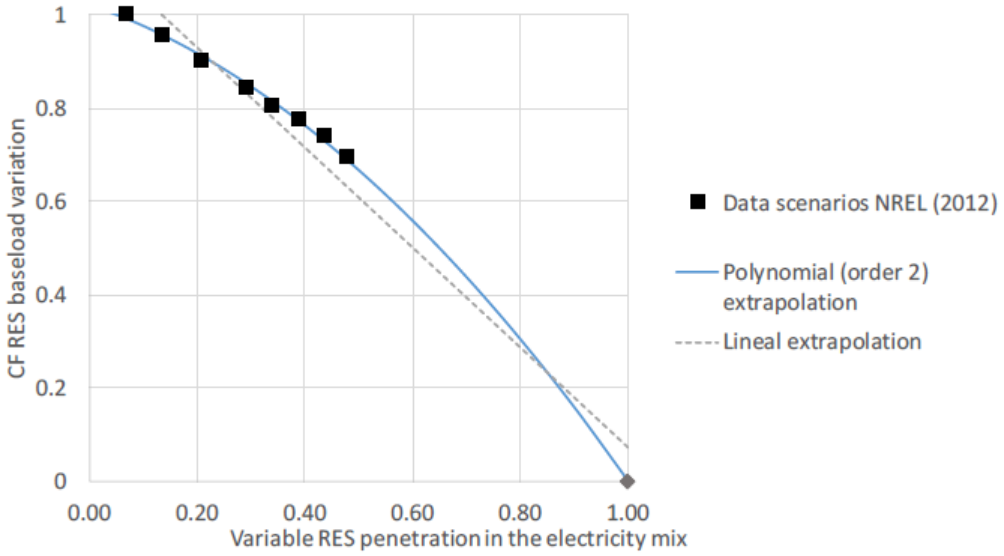


Figure 3.4: Capacity factor of RES evolution depending on the VRES share [26]

3.5.5 VRES plants

MEDEAS relies on estimates of the VRES induced overcapacities from [27]. These two main impacts of the VRES share are accounted for:

- The exponential growth of VRES overcapacities and
- The decrease of the VRES capacity factor.

The two estimate functions used in MEDEAS [25] from [27] are presented in Figure 3.5.

The capacity factor is evaluated as a function of the overcapacity, following Equation 3.8, that can actually be derived from Equations 3.6 and 3.7.

$$CF = \frac{1}{1 + overcapacity} \tag{3.8}$$

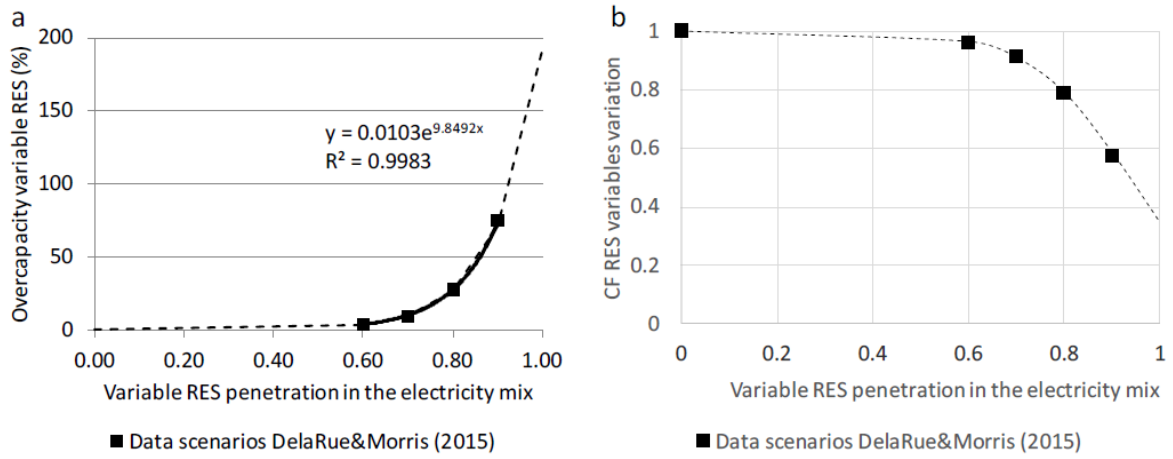


Figure 3.5: (a) Overcapacity estimate and (b) CF reduction of VRES plants on [25]

3.6 Scenarios

Given inherent uncertainties about the decision that will be made and will rule the society in the future, several sets of hypothesis, called scenarios, are defined.

When running the MEDEAS model, these scenarios are run in parallel, using Vensim tools for parallelism.

By default, the three following scenarios are available, but user-defined scenarios may be added and run [21].

- Business as usual, **BAU**: corresponds to no particular effort being implemented, the transition continues as it is.
- Optimal Level Transition, **OLT**: where all the resources available are allocated to the best renewable transition possible, that has become a social priority. The only constraints for faster transition are physical limitations.
- Mid-Level Transition, **MLT**: is a mix of the previous ones, some effort are made but not all. Actions towards renewable transition are delayed.

A depiction of these scenarios in terms of emissions over time is given in Figure 3.6.

These scenarios have been defined somewhat arbitrarily, but the custom scenario capability opens the door to anyone willing to use a more precise, more accurate scenario, given for example additional information brought by major events in the future.

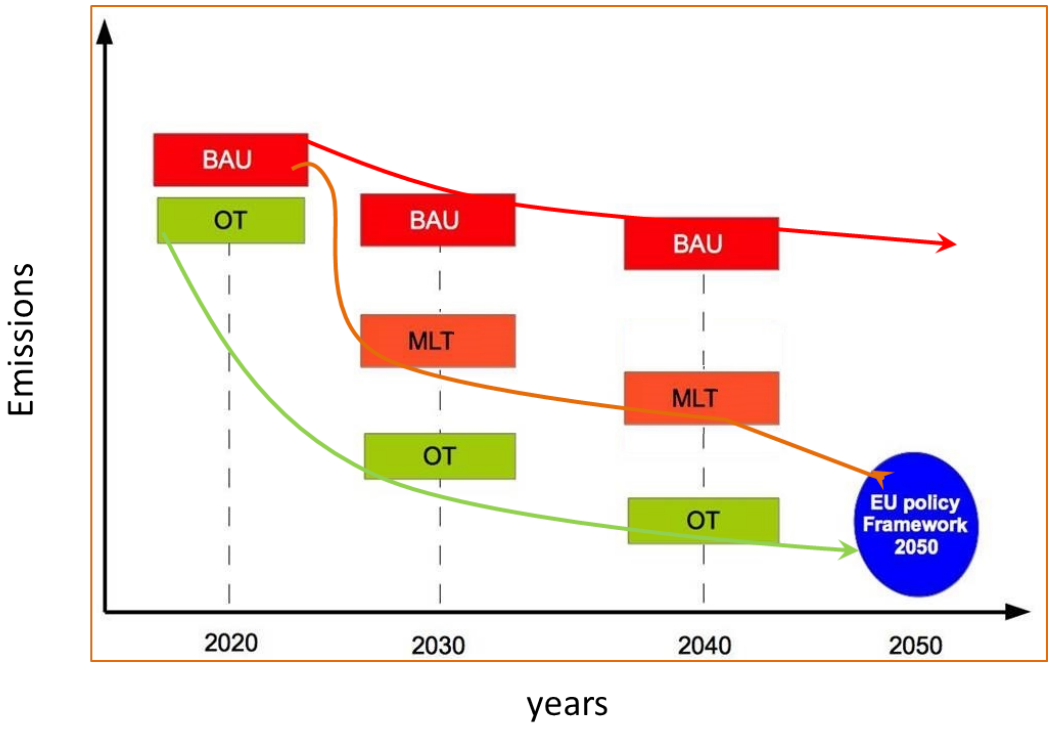


Figure 3.6: Qualitative illustration of the BAU, MLT and OLT (here, referred to as OT) scenarios for the greenhouse gas emissions in Europe from [21]

4 Dispa-SET simulation and database generation

4.1 Overview

This section describes 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 we properly select points to form the dataset features. Then, computationally expensive simulations are run on these points, and extract meaningful data is extracted from the simulation results to obtain the desired output features predicted by the model.

Finally, this dataset is used to train the surrogate model on [28].

4.2 Data preparation and initial parameters

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.

4.2.1 Unit groupings

In this context, the specific technology and fuel types of each plant hold no relevance, as they have no influence the input features of our dataset. There are less input features than technology-fuel pairs, and only the formers matter for the surrogate model training. Hence, the units are categorized into five groups: flexible units, slow units, storage units, PV units and wind units.

IRENA [3] 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". This criterion is be used to separate units into slow and flexible units, and their relative shares dictates the $Share_{flex}$ input. This criterion is presented in Table 4.1.

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 4.1: Classification of flexible and slow units

Refer to Tables 9.5 and 9.6 for identification of their respective names.

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

The groups 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

4.2.2 Parameters estimates

The availability factors of PHOT and WTON are also required, as well as the peak load. These values are given as time-series inputs to the simulations, for the year 2019. To give an order of magnitude, their averages, the derived capacity factor, are given in Table 4.2.

Variable	Value	Units
CF_{PV}	0.1314	[·]
CF_{WTON}	0.2604	[·]
CF_{WTOF}	0.3780	[·]
$PeakLoad$	440929	MW

Table 4.2: Average values of the capacity factor over all zones, and maximum total demand over the timeserie, i.e. sum of the demand in each country

4.3 Design space

For data sampling, the design space is of crucial importance. Its shape and dimensions, as well as the chosen sampling strategy, have a direct impact on the balance of the dataset, that can ultimately introduce some bias in the data.

4.3.1 Shape

The first necessary step in order to select our data points for our dataset, is to define the space in which we sample them. In our case, this space is 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 is 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 resemble some common shape like hypercube, hyperball or their combinations. Given that most of designs of experiments techniques assume these kinds of spaces, a mapping would be needed to benefit from more effective sampling strategies, designed for common shapes. 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, the hypercube is selected.

4.3.2 Input variables

The six adimensional variables are described in the following, alongside with their range. Each of these corresponds to one dimension of the hypercube.

The notation $PowerCap_x$ refers to the maximum power output of all the units in x, and $PeakLoad$ stands for the maximum total demand. See Table 4.2 for the values of the CF and peak load value.

1. **CapacityRatio [·]**

   Ratio of the maximum production over the maximum demand.

$$CapacityRatio = \frac{PowerCap_{flexunits} + PowerCap_{slowunits} + PowerCap_{storageunits}}{PeakLoad} \tag{4.1}$$

2. **ShareFlexibility [·]**

   Share of the units that are flexible.

$$Share_{flex} = \frac{PowerCap_{flexunits}}{PowerCap_{flexunits} + PowerCap_{slowunits}} \tag{4.2}$$

3. **ShareStorage [·]**

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

$$Share_{storage} = \frac{PowerCap_{storageunits}}{PeakLoad} \tag{4.3}$$

4. **ShareWind [·]**

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

$$Share_{wind} = \frac{PowerCap_{windunits}}{PeakLoad} \cdot CF_{WTON} \tag{4.4}$$

5. **SharePV [·]**

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

$$Share_{PV} = \frac{PowerCap_{PVunits}}{PeakLoad} \cdot CF_{PV} \tag{4.5}$$

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 4.6.

$$NTC_{z \to x} = \frac{1}{N_h} \sum_h NTC_{z \to x,h} \tag{4.6}$$

Then Equation 4.7 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.

$$NTC_z = \frac{\sum_x NTC_{z \to x}}{PeakLoad_z} \tag{4.7}$$

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 4.8.

$$rNTC = \sum_z \frac{PeakLoad_z}{\sum_x PeakLoad_x} NTC_z \tag{4.8}$$

4.3.3 Output variable

The target outputs of the systems are the curtailment and the load shedding. In order to make these values more scalable, we normalize them, by the maximum RES generation that could be produced (that is, the sum of the availability factors multiplied by the power capacity for each units), and the total demand respectively.

This normalization renders the output scalable, enabling its utilization in other systems, that have different scales. Without, the curtailment prediction is dependent on our specific training setting, and the resulting model, outputting absolute values, could not be used for any other setting. Scaling the outputs therefore enables easier generalization of the future model.

1. **Curtailment**

   Percentage of the curtailment to the maximum RES generation from all units:

   $$Curtailment = 100 \times \frac{EnergyCurtailed}{8760 \sum_{units} CF * PowerCap_u} \tag{4.9}$$

   Where $EnergyCurtailed$ represents the total amount of energy from VRES in MWh for all zones considered, CF is the yearly capacity factor and $PowerCap$ is the installed capacity for each unit in MW.

2. **LoadShedding**

   Percentage of the shedding to the total demand, that is, the part of the demand that has not been satisfied:

   $$LoadShedding = 100 \times \frac{\sum ShedLoad}{\sum Demand} \tag{4.10}$$

4.3.4 Reference values and ranges

The Dispa-SET simulation presented in Section 2.7 is taken as reference, and its main parameters are provided in Table 4.3.

A plausible variation range is then defined for the six important inputs (Table 4.3), covering the possible evolutions of the electricity system up to 2050.

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	0.5
rNTC	0.282	0	0.7

Table 4.3: Values of the different variable for the reference simulation in 2019.

4.4 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 achieve such sampling are [29] 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, and we have to set the strategy around the boundaries. It is depicted in Figure 4.1a.
2. The Monte-Carlo sampling: pick samples at random all over the design space. It is depicted in Figure 4.1b.
3. The Latin-hypercube sampling [30], maximizing a criterion that is either [31]:
   (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 in Figures 4.2a, 4.2b, 4.2c and 4.2d.

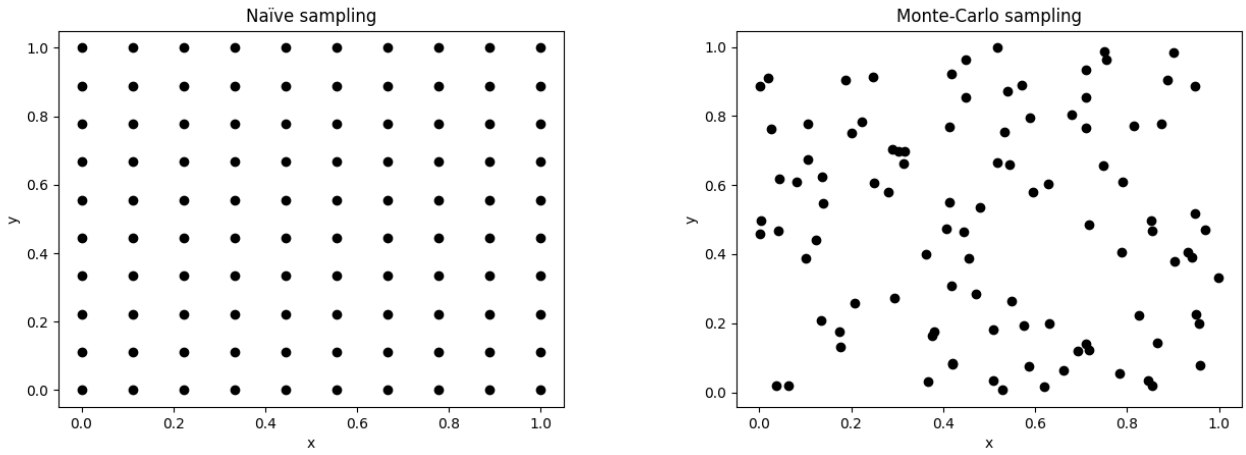
From these 2-dimensional illustration it is clear that the latin hypercube sampling performs best, the naive sampling featuring too much regularities that is not wanted, as they may introduce some bias, and Monte-Carlo sampling tends to make more clusters of samples, that would be inefficient (indeed,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

The number of points is set quite arbitrarily to 2000. This leads to an average of $\sqrt[6]{2000} = 3.550$ points per dimension, if the points had been placed on a 6-dimensional grid in the input space.

4.5 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 since the data that corresponds to these exact configuration is not available. 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 4.1: Basic sampling strategies

4.5.1 Adjusting functions

To address this, Dispa-SET provides utility functions that adjust the installed capacities within the power generation fleet.

– `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. Equation 4.11 gives an approximation of the update, but the full algorithm is presented in Algorithm 1.

$$PowerCap_{z,new} = PowerCap_{z,old} + \frac{PowerCap_{z,old}}{\sum_x PowerCap_{x,old}}(target - actual) \tag{4.11}$$

Depending on the sign of the flexibility difference δ, the remainder variable is set and the algorithm loops over each zones, computes a ratio corresponding approximately to how much the considered zone has to contribute to the change. Then the amount of flexible generation capacity is added or removed accordingly.

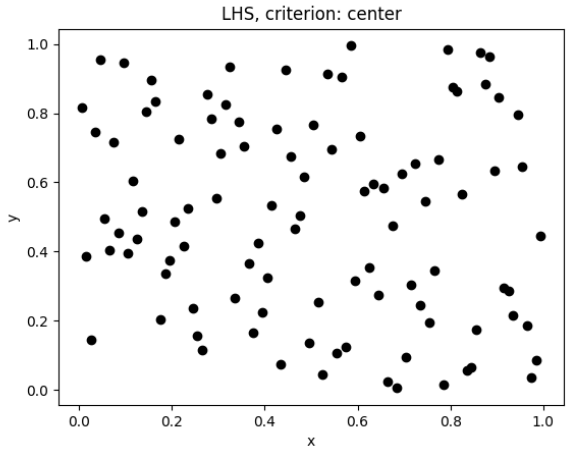
– `adjust_capacity` applies a linear scaling to the power output of some given set of units, in particular, it will be called multiples times to adjust the storage, PV and wind capacities.

Scaling factors applied to match desired values are summarized in Table 4.4.

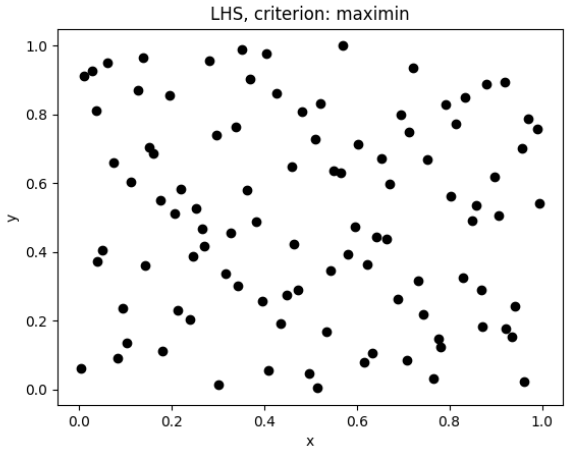
Units	Scaling factor
$Storage_{units}$	$Share_{storage}$
$Wind_{units}$	$\frac{CapacityRatio \cdot Share_{wind}}{CF_{wton}}$
PV_{units}	$\frac{CapacityRatio \cdot Share_{PV}}{CF_{PV}}$

Table 4.4: Scaling factors applied to different units

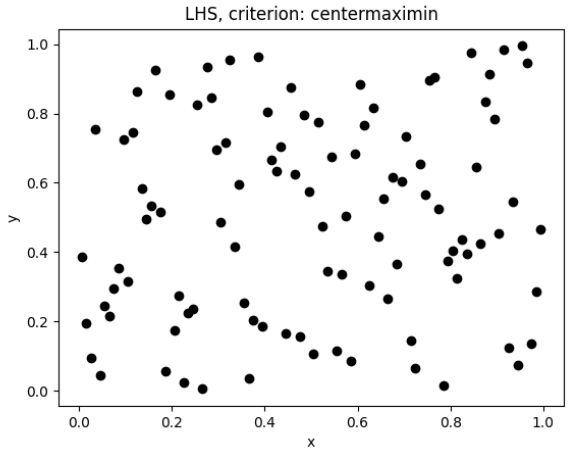
– `adjust_rntc` applies a linear scaling to each zonal NTC time series.



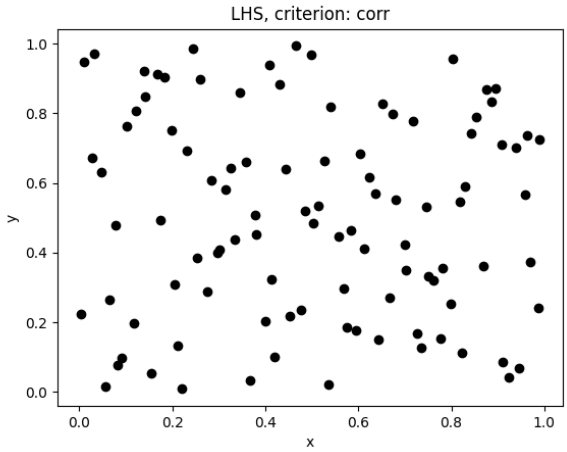
(a) LHS with center criterion



(b) LHS with max min distance criterion



(c) LHS with max min distance and center criterion



(d) LHS with correlation criterion

Figure 4.2: Latin hypercube sampling strategies with every criterion

4.5.2 Extracted outputs

Each simulation generates multiple outputs beyond curtailment and lost load values, some other outputs variables are extracted from the simulations, although not immediately exploited.

Of course, these may prove to be useful for future research.

All the outputs extracted from the simulations are displayed in Table 4.5.

Distinction should be made between the *Shedding* and the *LostLoad*. The former is a consequence of voluntary action, following set rules between consumers and producers. The latter is a consequence of additional variables added to Dispa-SET to reduce the infeasibility issues, assigning a high cost to the capacity the system is not able to generate due to maximum capacity and ramping constraint being reached. The Energy Not Served (ENS), is defined as the sum of *Shedding* and *LostLoad*, and thus accounts for the actual difference between the demand and the production predicted by Dispa-SET.

Algorithm 1 Adjust_flexibility algorithm

if $\delta > 0$ **then**:
    $remain \leftarrow \delta$
    for $z \in zones$ **do**
        $weight \leftarrow \frac{total_z}{current\_total\_cap - cum\_sum_z + total_z}$
        $added\_cap \leftarrow min(weight \cdot remain, total_z - flex_z)$
        $new\_flex\_cap_z \leftarrow flex_z + added\_cap$
        $new\_slow\_cap_z \leftarrow slow_z - added\_cap$
    end for
else if $\delta < 0$ **then**:
    $remain \leftarrow -\delta$
    for $z \in zones$ **do**
        $weight \leftarrow \frac{total_z}{current\_total\_cap - cum\_sum_z + total_z}$
        $removed\_cap \leftarrow min(weight \cdot remain, flex_z)$
        $new\_flex\_cap_z \leftarrow flex_z - removed\_cap$
        $new\_slow\_cap_z \leftarrow slow_z + removed\_cap$
    end for
else
    $new\_flex\_cap_z \leftarrow flex_z$
    $new\_slow\_cap_z \leftarrow slow_z$
end if

4.5.3 Dataset creation

With the help of the adjusting functions, creating a whole dataset now comes down to generate samples from LHS as previously discussed, adjusting the reference simulation setting to each sample, run the simulation and extract the desired features from the outputs.

As this whole process is be completely automated, it is pretty easy to obtain a second dataset, based on a different LHS. In particular, a smaller dataset is interesting for the validation and testing stages of the surrogate model, as will be elaborated in subsection 5.3.1.

4.6 Implementation

Running all of these simulation is not feasible on a basic hardware. A single MILP simulation typically takes around two hours to complete on the cluster, as the latin hypercube sampling targeted 2000 simulations, yielding to 4000 hours or 166.6 days if the simulation are run sequentially. Moreover, the testing phase also requires a significant amount of runs. 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 multiple of times, that are jobs independent of each other, running the linear programming software used by Dispa-SET, GAMS (General Algrbraic Modeling Language) [20].

4.6.1 Steps

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

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

Table 4.5: Values extracted from each simulations

1. Generating the reference simulation (cfr Section 2.7), 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

4.6.2 Scripts and code

All the files for this section lie in the `data-generation` folder.

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 (Simple Linux Utility for Resource Management) 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`, with serie index 0, 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.

This list is not exhaustive, see the Annex for a complete version.

4.6.3 Technical aspects

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

- The GAMS solver offers options to define the number of threads to do the computations, that number of threads cannot exceed the number of allocated CPUs by SLURM for that job, as it would disturb the node on which it is running, including affecting unrelated jobs.

  The thread setting in the `UCM_h.gms` file generated by Dispa-SET is set to 16 by default, this line may be removed if one wants to set it manually through the command line, as the file input takes precedence over the command-line value set.

- The total amount of each prepared simulation directory is too large to fit on the allocated disk memory on the `$HOME` partition on the cluster. Moreover, for efficiency it is better to use the `$GLOBALSCRATCH` file system.

- The Dispa-SET ajdusting function do not write adjusted data to a directory if this directory already exists before the function is called. That is, the directory containing the simulation files should not be created beforehand.

- Due to SLURM limiting the maximum number of jobs a user can have in the submit queue, queuing all the simulation jobs at once is not possible. Moreover, the number of simulation appeared to cause issues with the SLURM maximum array size.

  This is the reason why `launch-job-series.sh` exists. The script takes as an argument the serie index i, then starts a $[0:399]$ array of `launch-simulation-jobs.sh` jobs, with i as an argument. As the latter script has also access to its array index via SLURM environment variables, it can compute the intended simulation index, that covers the $[400i:400i+399]$ range. The next series is queued with the job array as a dependency, so that it is not launched before the current series terminated.

4.6.4 Unsuccessful simulations

During the execution of the simulation, it appeared that for some of them GAMS was completely stuck at some point, hence wasting resources and preventing the other simulations to start by keeping available CPUs busy.

A timeout on the GAMS simulation has therefore been set, after looking at the typical duration of a "regular" simulation. The simulation that were timed out had thus the corresponding error message in their output, that can be checked when reading the results.

Hence, we can label the samples whose simulation failed with an additional error field in the dataset, and compare them to the other. Typically, around 10% of the simulation fail, in this case 218 over 2400 total.

Loading this dataset and separating the successful simulations from failing ones, and extracting meaningful metrics from both, it appeared that the failing simulations featured an average share flexibility significantly higher than the other, while the other input parameters seemed around the same. The share of flexible generation is taken in the $[0.01, 0.99]$ interval, and the average of failing simulations is around 0.49 higher than the average of the successful ones.

To further validate our observation, the quantiles of the distribution of the share of flexible generation in the stalling settings are explored, and it appears that they lie really high, the first one (25%) is around 0.9, the median lies near 0.95 and the third one (75%) is around 0.96.

More decisively, the maximum value of a successful simulation is 0.9011, while the minimum value of a stalling simulation is 0.9015, that is, slightly more than the max of valid simulations. Thus, there exists a net boundary between the simulation that run successfully and the other.

The distribution of *ShareFlex* values in both part of the dataset is depicted as boxplots in Figure 4.3

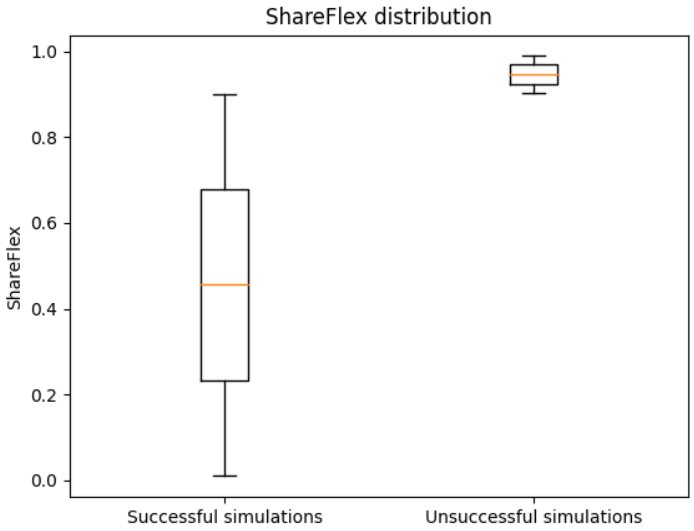


Figure 4.3: Distributions of the *ShareFlex* parameter value in the unsuccessful simulations against successful simulation. The distributions of the other parameters are similar.

As a consequence, the validity region for the surrogate model has to be updated, as there is no way of producing an estimate of the outputs on a region where no sample was simulated correctly.

This issue is interpreted as the simulations with too high flexibility are harder to solve than the other, as the system has a lot of options to choose from when confronted to a change in demand, yeilding an exponential growth of the realistic evolutions of the system.

It is noteworthy that in about 30% of the simulation, a division by zero error was also stated. Since this happens in the preprocessing, it has no influence on the results and this error is discarded.

4.6.5 Dataset fields

For completeness, all the fields in the created dataset in `dataset.csv` are shown in Table 4.6.

Only two of these, *LoadShedding* and *Curtailment* are actually used to train the surrogate model, and six as the model features. As the marginal cost of including all the other outputs is null, they are saved as well in case someone would find a use for it in the future.

Parameter	Unit	Parameter	Unit
Cost	€/MWh	Curtailment	%
Congestion	h	MaxLoadSheddingShare[1]	%
PeakLoad	MW	CF gas	[·]
MaxCurtailment	MW	CF nuc	[·]
MaxLoadShedding	MW	CF wat	[·]
Demand	TWh	CF win	[·]
NetImports	TWh	CF sun	[·]
Curtailment	%	Capacity ratio	[·]
Shedding	%	Share flex	[·]
LostLoad	TWh	Share sto	[·]
MaxRESGeneration	MW	Share Wind	[·]
TotalGeneration	TWh	Share PV	[·]
ShareRESGeneration	%	rNTC	[·]
LoadShedding	[·]		

Table 4.6: Dataset fields. "%" as a unit means a unitless ratio multiplied by 100. Elements displayed in green correspond to the six features, and elements in blue the two target outputs.

[1]The MaxLoadSheddingShare is taken as a fraction of the demand at the time.

5 The surrogate model

5.1 Overview

This section presents the training process that have led 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 regression problems, aiming to predict some output features from the input features, given a number of training example in a dataset. This is a typical machine learning problem, and is be addressed accordingly.

Initially, the performance of different methods are measured and the best one is selected. Then, a model is trained and evaluated, forming the core surrogate model.

5.2 Machine learning methods

This subsection assesses some of the most common machine learning algorithms for regression problems [32].

5.2.1 K nearest neighbors

In this context, the K-nearest neighbors (k-NN) methods is among one of the easiest to implement, using tools like the scikit-learn [33] module in python. 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 features the drawback of not being able to learn quick variations in the outputs, as the averaging over the nearest sample points will filter out high frequency variations.

Results obtained with k-NN for various values of k are shown in Table 5.1.

k	Validation error
4	0.05082837
5	0.051765025
6	0.047083396
7	0.046183977
8	0.049501333
9	0.049218103
10	0.04949624
11	0.05036374
12	0.050881956
13	0.05243453
14	0.052941263
15	0.05362743
16	0.05440363

Table 5.1: Results obtained with k-NN

The optimal performance is achieved when k is set to 7. Decreasing values of k lead to an increase in error due to overfitting, whereas increasing values of k result in higher errors attributed to underfitting.

5.2.2 Decision trees

Decision trees, along with extremely randomized trees [34] are easy to implement[2] machine learning methods as well, as they also involve a constrained and fixed number design parameters.

These methods, however, exhibit the limitation of producing 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 using many successive steps, being peicewise constant will now introduce non linearities in the outputs, that is, unwanted steps in the prediction.

Results obtained with decision trees are shown in Table 5.2.

Number of trees	Validation error
25	0.02385
35	0.02347
45	0.02275
55	0.02342
65	0.02361
75	0.02272
85	0.02360
95	0.02255
105	0.02294
115	0.02379
125	0.02263
135	0.02232
145	0.02317

Table 5.2: Results obtained with random forests

The highest accuracy is achieved with 135 trees. However in this case, the results are not as straightforward as for the k-NN technique, that showed a monotonic decrease, the minimum then an increase. As randomness is involved when choosing the splits when building the trees, the results are subject to a slight variance, inducing less obvious results. However, after 145 trees, the error starts steadily increasing.

These performance are higher than those of the nearest neighbors method, but not as strong as the following techniques.

5.2.3 XGBoost

XGBoost is currently one of the most popular machine learning technique [35]. It actually is an efficient implementation of a machine learning method, tree boosting.

Boosting aims to construct a string predictive model from so-called weak learners, which in this case, are trees. Each learner is trained on the error of the output of the previous model, so that the previous output plus the newly trained learner output is a better prediction of the target output. Moreover, the samples are weighted in favor of the ones that were badly predicted by the previous model, in order to make sure we correct past mistakes.

In tree boosting, the main hyper-parameters are:

[2]Again, using scikit-learn

– the number n of weak learners stacked

– the *learning rate*, a constant factor applied to the target difference (target output minus previous model output)

– the maximum depth of the weak learners

Resulting performance for some values of the hyper-parameters are reported in Table 5.3.

XGBoost's results are really good, achieving at best around 0.006 mean squared error on the validation set. This is not surprising given that this technique is quite popular in the literature.

It is important to note that the method was tested over a previous version of the dataset (the one produced in [36]) and not on the one generated in this thesis because the results were not yet available. The performance metrics should however not very dramatically between the former and the latter version of the dataset.

The best performance is observed with a maximum depth of 4, a learning rate set to 0.03 and 2000 trees. Further fine-tuning may slightly improve this value, in particular reduce the learning rate and increase the number of trees even more, but further investigation showed that the performance stagnate starting from 1800.

5.2.4 Kernel-based methods

One might consider applying 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, nevertheless, finding a suitable kernel is a difficult.

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

5.2.5 Artificial neural network

Artificial neural networks (ANN) are used in this work to build our surrogate model. These types of methods offer a large flexibility, due to their entirely customizable architecture, as well 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 [37].

In this case, the inputs boils down to the 6 variables values, listed in Table 4.3. 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

d	lr	n	err	d	lr	n	err	d	lr	n	err
3	0.03	10	0.5248	4	0.1	10	0.1564	5	0.2	10	0.03299
3	0.03	20	0.3367	4	0.1	20	0.04522	5	0.2	20	0.01088
3	0.03	50	0.1100	4	0.1	50	0.01059	5	0.2	50	0.007287
3	0.03	100	0.033627	4	0.1	100	0.007168	5	0.2	100	0.006907
3	0.03	200	0.01294	4	0.1	200	0.006018	5	0.2	200	0.006681
3	0.03	500	0.007804	4	0.1	500	0.005489	5	0.2	500	0.006624
3	0.03	1000	0.006434	4	0.1	1000	0.005325	5	0.2	1000	0.006623
3	0.03	2000	0.005553	4	0.1	2000	0.005294	5	0.2	2000	0.006623
3	0.1	10	0.1886	4	0.2	10	0.04046	6	0.03	10	0.4870
3	0.1	20	0.06469	4	0.2	20	0.01313	6	0.03	20	0.2861
3	0.1	50	0.01586	4	0.2	50	0.007588	6	0.03	50	0.06760
3	0.1	100	0.009350	4	0.2	100	0.006579	6	0.03	100	0.01544
3	0.1	200	0.007291	4	0.2	200	0.006219	6	0.03	200	0.007516
3	0.1	500	0.005978	4	0.2	500	0.006029	6	0.03	500	0.006548
3	0.1	1000	0.005558	4	0.2	1000	0.006007	6	0.03	1000	0.006381
3	0.1	2000	0.005375	4	0.2	2000	0.006007	6	0.03	2000	0.006343
3	0.2	10	0.059425	5	0.03	10	0.4916	6	0.1	10	0.1368
3	0.2	20	0.02037	5	0.03	20	0.2894	6	0.1	20	0.03335
3	0.2	50	0.009474	5	0.03	50	0.07119	6	0.1	50	0.008382
3	0.2	100	0.007554	5	0.03	100	0.01647	6	0.1	100	0.006850
3	0.2	200	0.006287	5	0.03	200	0.006834	6	0.1	200	0.006594
3	0.2	500	0.005976	5	0.03	500	0.005326	6	0.1	500	0.006504
3	0.2	1000	0.005935	5	0.03	1000	0.005069	6	0.1	1000	0.006491
3	0.2	2000	0.005894	5	0.03	2000	0.004952	6	0.1	2000	0.006491
4	0.03	10	0.5013	5	0.1	10	0.1425	6	0.2	10	0.0302
4	0.03	20	0.3042	5	0.1	20	0.03577	6	0.2	20	0.01046
4	0.03	50	0.08414	5	0.1	50	0.008340	6	0.2	50	0.007929
4	0.03	100	0.02168	5	0.1	100	0.006145	6	0.2	100	0.007743
4	0.03	200	0.008364	5	0.1	200	0.005640	6	0.2	200	0.007651
4	0.03	500	0.005841	5	0.1	500	0.005433	6	0.2	500	0.007622
4	0.03	1000	0.005203	5	0.1	1000	0.005371	6	0.2	1000	0.007622
4	0.03	2000	0.004922	5	0.1	2000	0.005371	6	0.2	2000	0.007622

Table 5.3: Results for some values of the XGBoost parameters. The learning rate is labelled "lr", the maximum depth "d" and the number of estimators n, while the validation error is denoted by "err".

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

Some experiments are run to provide some baselines for getting a first approximation of what performs well. These results are presented in Table 5.4.

These result outperform every other method. The fact that different but close architecture show similar performance brings some confidence about the reproducibility: the random initialization of the weights in the network could have led to an exceptionally good model. Moreover,

Architecture	Validation error
(180, 'relu', 0.4), (100, 'tanh', 0.4)	0.00484
(180, 'relu', 0.4), (100, 'tanh', 0.4)	0.00499
(180, 'relu', 0.4), (100, 'tanh', 0.4)	0.00480
(70, 'relu', 0.5), (70, 'relu', 0.5)	0.04538
(100, 'relu', 0.4), (100, 'relu', 0.4)	0.02506
(100, 'relu', 0.5), (100, 'relu', 0.5)	0.04111
(100, 'relu', 0.6), (100, 'relu', 0.6)	0.04481
(100, 'relu', 0.7), (100, 'relu', 0.7)	0.10410
(80, 'relu', 0.7), (80, 'relu', 0.7)	0.06514
(150, 'relu', 0.6), (100, 'relu', 0.6)	0.02693
(250, 'relu', 0.4), (125, 'relu', 0.4)	0.01295
(200, 'relu', 0.5), (125, 'relu', 0.5)	0.01734
(200, 'relu', 0.5), (125, 'tanh', 0.5)	0.00532
(200, 'relu', 0.4), (125, 'tanh', 0.4)	0.00555
(200, 'relu', 0.4), (100, 'tanh', 0.4)	0.00581
(150, 'relu', 0.45), (100, 'tanh', 0.45)	0.00585
(150, 'relu', 0.5), (100, 'tanh', 0.5)	0.00603
(150, 'relu', 0.4), (80, 'tanh', 0.4)	0.00597
(220, 'relu', 0.5), (125, 'relu', 0.5)	0.01086
(250, 'relu', 0.5), (125, 'relu', 0.5)	0.02421
(250, 'tanh', 0.4), (125, 'tanh', 0.4)	0.04193
(200, 'relu', 0.5), (150, 'relu', 0.5), (100, 'relu', 0.4)	0.07372
(120, 'relu', 0.4), (120, 'relu', 0.4), (120, 'relu', 0.4) (120, 'relu', 0.4), (80, 'relu', 0.4)	0.12163
(50, 'relu', 0.3), (50, 'relu', 0.3), (50, 'relu', 0.3), (50, 'relu', 0.3)	0.08267
(50, 'relu', 0.4), (50, 'relu', 0.4), (50, 'relu', 0.4), (50, 'relu', 0.4)	0.14292
(50, 'relu', 0.2), (50, 'relu', 0.2), (50, 'relu', 0.2)	0.02622
(50, 'relu', 0.2), (50, 'relu', 0.2), (50, 'relu', 0.2), (50, 'relu', 0.2)	0.05221
(50, 'relu', 0.3), (50, 'relu', 0.3), (50, 'relu', 0.3)	0.05723
(50, 'relu', 0.4), (50, 'relu', 0.4), (50, 'relu', 0.4)	0.08706
(150, 'relu', 0.5), (150, 'relu', 0.5), (150, 'relu', 0.5)	0.07769
(150, 'relu', 0.5), (100, 'relu', 0.5), (100, 'relu', 0.5)	0.08809
(150, 'relu', 0.5), (150, 'relu', 0.5), (100, 'relu', 0.5)	0.08098
(200, 'relu', 0.5), (200, 'relu', 0.5), (150, 'relu', 0.5)	0.07682
(190, 'relu', 0.5), (190, 'relu', 0.5), (140, 'relu', 0.5)	0.06489
(200, 'relu', 0.5), (200, 'relu', 0.5), (150, 'relu', 0.5), (30, 'relu', 0.3)	0.08749
(200, 'relu', 0.5), (200, 'relu', 0.5), (150, 'relu', 0.5), (50, 'relu', 0.3)	0.09094
(200, 'relu', 0.5), (200, 'relu', 0.5), (200, 'relu', 0.5)	0.07010
(150, 'relu', 0.5), (150, 'relu', 0.5), (150, 'relu', 0.5), (150, 'relu', 0.5)	0.13490

Table 5.4: Results for some architectures of neural networks. Architectures are formatted as a list of layers, which are written as (n, a, p) tuples, where n is the number of neurons, p the dropout value and a the activation function

several runs of the same architecture yield similar ouputs, validating this intuition.

It is worth mentioning that some larger network architecture, *e.g.* $(300,'relu',0),(300,'relu',0),(200,'relu',0)$ achieved really good results, even slightly better than the best one. However these were not implementing dropout, so that they are high changes of being subject to overfitting. If dropout is added, the observed performance decrease, confirming the overfitting hypothesis.

This is why the relative smallness of the network is a strength, as it will also act against overfitting, as having fewer parameters leaves less room for learning noise.

5.2.6 Selection of the machine learning technique and parametrization

In the end, neural networks are opted for. They benefit from the best performance in terms of mean squared error on the validation test, but neural networks MLPs also present the following advantages:

- They are relatively lightweight, in comparison to the K-nearest neighbors methods, that needs to store the entire dataset, and the randomized trees, that need to store its trees structures. Neural networks only needs their weights that are fairly small with this few input variables.
- Grasp non-linear behaviour with accuracy

Preliminary experiments basically consist in the test undertaken to assess the performance of different architectures (reported in Table 5.4). While the firsts steps of testing used another dataset that available at the time, ultimately the generated dataset was the only one in use for this assessments.

The architecture is selected as the optimal one among baselines, and is provided in Table 5.5. While hyper-parameter tuning may have produced another result, the latter could potentially suffer from overfitting, as described above.

Layer index	Layer type	Weights count
1	Fully connected linear layer, 6 inputs to 180 outputs	$6 \times 180 = 1080$
2	ReLU activation function	0
3	Dropout layer, with $p = 0.4$	0
4	Fully connected linear layer, 180 inputs to 100 inputs	$180 \times 100 = 18000$
5	tanh activation function	0
6	Dropout layer, with $p = 0.4$	0
7	Fully connected linear layer, 100 inputs to 2 outputs	100×2

Table 5.5: Description of the layers of in the selected architecture for the model

Consequently, the final parameterization of the model involves the setting of all its weights, that lie in layers 1, 4 and 7. The total count of parameters adds up: $1080 + 18000 + 200 = 19280$.

5.3 Machine learning aspects

5.3.1 Validation and testing

In machine learning, validation and testing are crucial steps in order to ensure the performance of a model. To implement these, one must separate the data into three sets.

– The validation set is used for hyperparameters tuning. Given its influence on the model, it can introduce bias into the model.
– The test set is used to evaluate the model's performance, on unseen data.
– The training set is used to train the model.

Typical machine learning applications dispose of a single, fixed dataset, that has to be split to constitute three desired sets. However, in this context, we can inexpensively run sets of simulations to obtain more data points, from another LHS. It is interesting to consider generating more than a single dataset from a unique LHS.

For the training set, joining different sets drawn with LHS is not expected to improve performance. As the LHS are independent of each other, there is a great chance to draw samples that are really close to each other, or worse, equal, what the LHS aims to avoid.

On the other hand, using a set drawn from a different LHS (than the one used for the training set) for testing and validation is interesting. This will provide data that spans the whole input space, but not yet exactly the same as the data used for training. Thanks to the input space coverage, the evaluation made at testing will be more extensive, and thanks to being independent of the training data, this evaluation should remain unbiased.

Conversely, the validation set being sampled from a LHS now providing a guarantee that it spans the whole space could make the work of the learning agent easier, giving it access to each of the LHS sample from the training set.

However promising, this approach will not be done in this work, to ensure that the unbiased property of the trained model persists, and that the term "validation error" keeps the exact same meaning as in the literature.

5.3.2 Overfitting

An inherent challenge 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 a result of excessive 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.

The main tool used to prevent it in a neural network is the dropout. During the training phases, each neuron on a layer will have some probability p, typically between 0.3 and 0.5, of being set to 0, independently of its value. This methods ensures that the network will not be excessively relying on some neurons in its output. During testing oviously, the dropout is removed and all neurons are functional.

5.3.3 Underfitting

On the other hand, underfitting means there is not enough learning happening. For example, a neural network with only 2 neurons corresponding to the ouputs could not learn any non-linear functions, with some tweaks due to the eventual activation function.

In this setting, although underfitting is still a shortfall, it is not regarded as bad as the overfitting. The latter introduces errors that come from noise that is completely random, inevitably introducing imprecisions. But we keep in mind that the target to be approximated, that are outputs from the Dispa-SET model, are themselves an approximation, though accurate, of the reality. So if these are subject to some bias or imperfections, the best model would learn them as well.

Because an underfitting model of the dataset created using Dispa-SET would still be a decent estimate of the reality, which is the primary objective in this context, underfitting is preferable to overfitting.

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

5.3.4 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 implement 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.

Hence, the primary source of bias that can be addressed pertains to model training inaccuracies resulting from suboptimal 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.

5.4 Training

5.4.1 Implementation

All the files for this section lie in the `nn` folder.

The implementation of the training process comprises the following files:

- `config.py`, that holds all the high-level specifications of the training, such as the names of the outputs, the train-test-validation set ratios, the number of epochs etc.
- `model.py`, that contains the function building the model, thus the definition of the neural network's architecture.
- `baselines.py`, that contains code to train models with pre-defined architectures, in order to quickly and easyly have an overview of the order of magnitude involved.
- `train.py`, that contains the code for the hyperparameter tuning and model training.
- `view.py`, that contains the utilities to visualize the results.

– `tests.py`, that contains the testing of the other machine learning methods (k-NN, trees, XGBoost)

– `data`, `logs`, `models` directories, that contain the datasets, the runs' logs, and the trained models respectively.

5.4.2 Results

The selected architecture is a two-layer network:

1. 180 neurons, ReLU activation, 0.4 chance of dropout
2. 100 neurons, hyperbolic tangent activation, 0.4 chance of dropout

The mean squared error over the training epochs is shown in Figure 5.1.

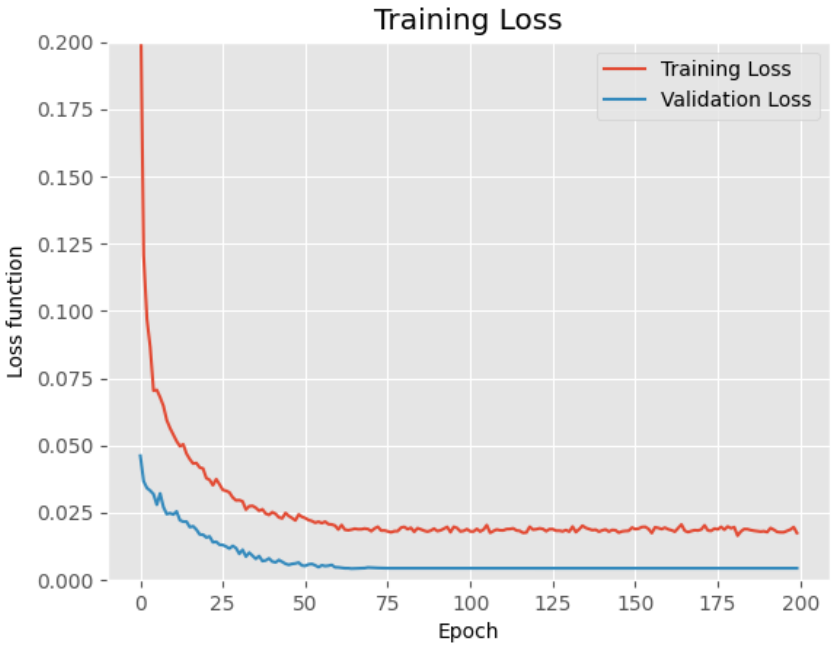


Figure 5.1: Mean squared error on the validation set

The occurrence of overfitting appears unlikely, as evidenced by the trend observed in Figure 5.1, wherein the validation error exhibits no notable increase beyond a certain number of training epochs. This pattern contrasts with the ongoing reduction in training error.

5.4.3 Observations

To further confirm the absence of overfitting in this result, the `view.py` file is used to browse across the multi-dimensional function, with surface plots.

These plots draw one of the outputs against two of the inputs, keeping the four other inputs constant. These constant values are summarized in Table 5.6. Attention should be paid to the scale of the plots, as these do not start at 0.

Several surfaces are depicted in Figure 5.3. The following observation are made from the latter illustrations.

Figure 5.2: An (unrelated) example of clear overfitting, the validation loss increasing after some time

Input name	Value
Capacity ratio	1.15
Share flexibility	0.5
Share storage	0.25
Share wind	0.25
Share PV	0.25
rNTC	0.35

Table 5.6: Default values for constant inputs. These are the middle of their base interval, see Table 4.3

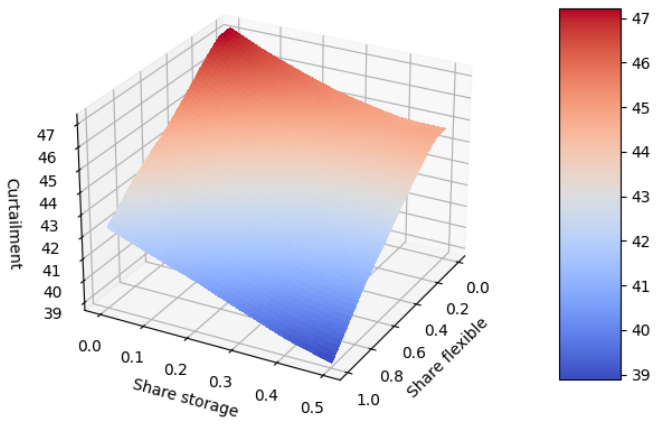
- Figure 5.3a follows the natural intuition, as higher shares of storage units and flexible units both contribute to the reduction of the curtailment.
- Figure 5.3b points out the crucial impact of the share of flexible units on the load shedding for small values. Naturally, when peaks in demand appear, if no unit is susceptible to be started, there is no other way than to cut the exceeding demand out.
- Figure 5.3c also confirms the intuition, as it outlines the increase in curtailment with an increase of either share solar or share PV.
- Figure 5.3d is the most intriguing one. First, it is noticed that the scale is the lower here. The most surprising part is that it features local minima, and that when the share PV is close to 0, the load shedding as a function of the share of wind energy is U-shaped. This phenomenon is hard to explain theoretically, the most likely cause remains a weak learning of the model.

  Moreover, the shape of the surface changes rapidly when changing the other constant values, therefore taking other values than disclosed in Table 5.6, which validates the hypothesis of a weak learning.

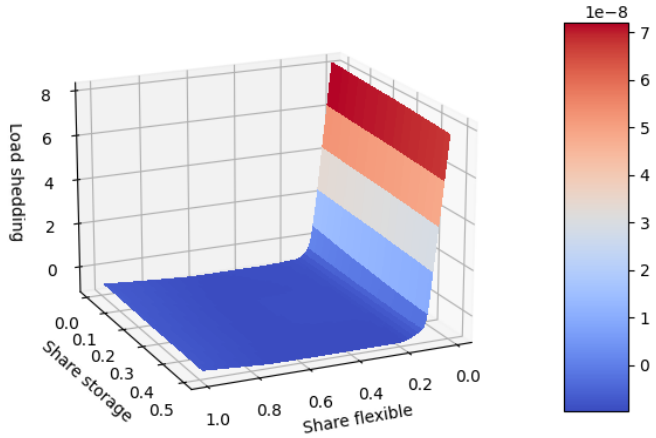
In comparison, the other surfaces do not move that significantly with similar change, coming down to a slight upwards or downwards shift, depending on the direction of the change, occasionnaly with a light amplitude change.

– Figure 5.3e follows the intuition as well, but also highlights the stronger impact of the capacity ratio on the curtailment compared to the rNTC. This is not surprising, as the share of electricity import is not that huge. However, this may change dramatically depending on the country considered.

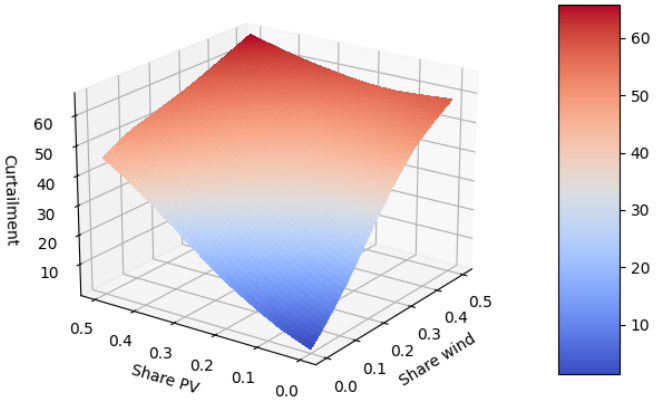
– Figure 5.3f is interesting, as it shows that the load shedding grows when both the rNTC and capacity ratio lower. As low rNTC means little importation possible, and low capacity ratio not much margin to fulfill the demand, this actually makes sense.



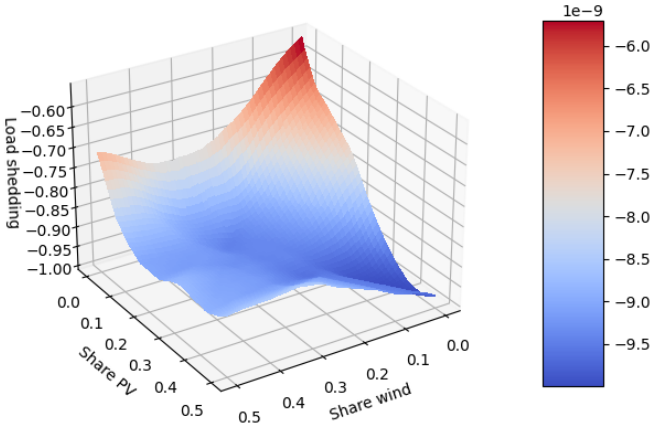
(a) Curtailment against share flexibility and share storage



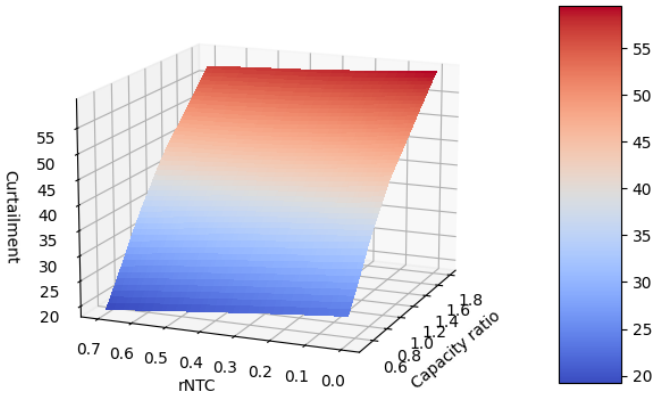
(b) Load shedding against share flexibility and share storage



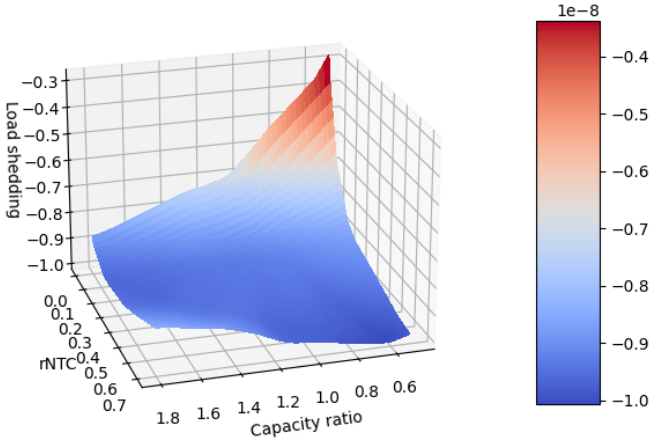
(c) Curtailment against share wind and share PV



(d) Load shedding against share wind and share PV



(e) Curtailment against capacity ratio and rNTC



(f) Load shedding against capacity ratio and rNTC

Figure 5.3: Different views of the multi-dimensional function, with four fixed inputs and two varying inputs.

6 Integration

6.1 Overview

This section elucidates the procedure for incorporating the trained surrogate model into the MEDEAS model.

The MEDEAS model is built with the Vensim software, and is available to the public as a Vensim model file. However, the python library `pysd` [38], focused on system dynamics simulations, is able to parse such files and run the simulations. The rationale behind not capitalizing on this opportunity is expounded upon in the initial subsection.

The task of integrating the surrogate model into MEDEAS is divided into two smaller steps:

1. **Vensim integration**, that is to make the surrogate model, available as a Tensorflow model, callable from within a Vensim model
2. **Variable linking**, Link the input and output features of the surrogate model to the actual variables used in the MEDEAS model

6.2 Vensim integration

6.2.1 The Vensim software

Vensim is a system dynamics simulation software, developped by Ventana Systems, Inc. It primarily solves the system of differential equation represented by the user-defined model, and is mainly used, according to its description, "for developing, analyzing, and packaging dynamic feedback models" [39].

Its most common application areas include [40]:

- Transportation and energy,
- Project management,
- Environment.

Vensim also comes in different distribution, such as Vensim PLE that is the free, personal learning edition. In this work, Vensim DSS is used, with an academic license.

6.2.2 Vensim external functions

This specific feature of the Vensim software is evidently of great interest for this work. It enables the user to provide and use custom functions in Vensim models and simulations.

To do so, the user needs to provide a dynamically linked library (DLL), packaged as a `dll` file, then provide its path in Vensim. These files are Windows specific (as Linux uses `so` files and macOS `dylib`), hence they have to be handled as such.

Dynamically linked libraries are pieces of compiled code that can be loaded at runtime by another program. User-defined functions necessarily need to be loaded at runtime, as the software cannot know them in advance. These are typically compiled from the C or C++ programming language. In this work, C++ has been chosen, as the library to load and call Tensorflow models that is used is written in that language.

To integrate user-defined functions in its simulation environment, the library is expected to provide some functions, that follow an interface that the Vensim software knows and can employ. This interface comprises of a set of functions that is described in Table 6.1.

Function name	Description
`version_info`	Provide information about the Vensim version this library has been built for
`set_gv`	Utility to set the global variable that depend on the Vensim simulation environment
`user_definition`	Provide a way to get all the necessary information about each user-defined function, mostly their names, number of arguments and a identifier code
`simulation_setup`	This function is called by Vensim on simulation startup, allowing the library to do some preparative work if needed, such as allocating memory
`simulation_shutdown`	Same as `simulation_setup`, but on simulation shutdown
`vensim_external`	This function is expected to, given an array of input arguments, their number and a function code, call the function associated to that code and write its return value into the first input argument.

Table 6.1: Description of the mandatory functions that a Vensim user library has to provide

Luckily, Vensim DSS ships with an example of such a library. As this file is not publicly disclosed, caution should be paid to keeping the library code private—actually, the external function capability is only available in Vensim DSS.

In the implementation of the library, this file was copied then adapted, as suggested in Vensim's documentation.

6.2.3 Calling a Tensorflow model

As a result, it becomes imperative to invoke our model, which is referred to as a Tensorflow model, utilizing either the C or C++ language."

In order to do so, one basically needs two things:

- the model in question, saved in a directory from the Tensorflow python API.
- the Tensorflow library, which is another DLL, to perform the actual computations from C/C++.

The complicated part being the linking between the two. In order to do so, the Cppflow tool is used. It serves as an intermediary layer between C++ and the Tensorflow model. This tool is not available in C, this is the primary reason why the library is implemented in C++.

The main purpose of Cppflow is precisely to run Tensorflow models from C++, and to achieve this it provides user-friendly functions for loading models and making predictions using input data.

For ensuring the correct linkage between the library code and the two employed tools, namely the Tensorflow DLL and CppFlow, an apt linking strategy is imperative. To facilitate this,

Makefiles were crafted using the GNU `make` utility for streamlined compilation[3].

On this, one thing is to be remembered: some code may compile and link successfully, but one also has to link properly the path to the DLL you linked to, that is, not only to the compiler.

6.3 The pysd option

As previously introced, the `pysd` python library can read and run Vensim model files. Of course, as the surrogate model is primarily defined in that language, one would deduce that its integration would be easier that way.

Though not explored in this work, integrating the surrogate model through `pysd` is expected to be relatively straightforward. Using the library loading functionality, a python module representing the simulation can be obtained. This module is then loaded with `pysd` to run the simulation. The linking can thus be done by editing the module file directly, and inserting the call to the model at the appropriate place.

However, this approach was not selected. Although contemplated at a subsequent stage of the project, subsequent to the integration into Vensim having been implemented, the principal rationale stems from the convenience afforded by the resultant model. If the model combination was made available sa python module, it would be much harder for external people to edit the original MEDEAS model from Vensim, then run the modified version while keeping the surrogate model integrated. The editor would have to manually re-insert the surrogate model into the python module, that also have to be recreated.

Keeping the surrogate model available as a Vensim external function is therefore a significant benefit for the further improvements of the model, as it enables edits to be made to MEDEAS independently of the surrogate model.

Another update has to be mentionned here. In facts, J. Paris had trouble making the MEDEAS model run successfully with the integrated surrogate model had contact with the MEDEAS developping team, that advised to wait for the release of a update of the MEDEAS port to python, making use of `pysd` and expected to be more stable and convenient. The main issue was that she were to leave before the anticipated release date, such that it was not acheivable.

This newer Python port with `pysd` might ultimately render direct integration in Python more portable and convenient. This would offer the same advantage as the Vensim external function. Still, this function was created and finished before learning the future availability of the new python port.

6.4 Variable linking

In its underlying workings, the MEDEAS model does not directly employ all the variables that appear in our surrogate model. This necessitates establishing connections between the two.

Fortunately, many of the desired values bear a close relationship to existing variables within the MEDEAS energy module, so these links are expected to be as simple as linear rescalings or

[3]Previously, a program calling a Tensorflow model was created, but this did not worked from the DLL. Then, a workaround was developed using a worker process (a seperate program) and Windows' tools for interprocess communication. Later, it was discovered that the issue with calling Tensorflow stemmed from errors in linker arguments during library compilation. Once these linker issues were resolved, the need for the worker process workaround became redundant.

combinations.

The input variables, that are summarized in Table 4.3, do not map directly to already existing variables in the MEDEAS model. Thus, some mappings have to be made between the inputs and outputs of the surrogate model, the Dispa-SET side, and the MEDEAS side. For clarity, all the variables in Dispa-SET are displayed in blue, while those from MEDEAS are coloured in red.

This work has been done with the help of Jade Paris, a student making her master's stage thesis on this specific topic as well.

6.4.1 Variables available in MEDEAS

The first step in this process is to list relevant variable present in MEDEAS, that are exploited in the following when drawing connections between the surrogate model and MEDEAS.

These variable are enumerated in Table 6.2.

MEDEAS variable name	Description	Unit	Notation
FE elec generation from solar PV TWh	Total yearly production from solar photovoltaic units	TWh	$Generation_{PV}$
Total FE Elec demand TWh	Yearly total electricity demand	TWh	$Demand_{tot}$
FE Elec generation from offshore wind TWh	Total yearly production from offshore wind turbines	TWh	$Generation_{wind-offshore}$
FE Elec generation from onshore wind TWh	Total yearly production from onshore wind turbines	TWh	$Generation_{wind-onshore}$
Total capacity elec storage TW	Total power output of storage units	TW	$Storage_{tot}$
Total FE Elec genetaion TWh EU	Total yearly electricity production from all units	TWh	$Generation_{tot}$
new capacity installed growth rate RES elec	Yearly growth rate of the electric network capacity	[·]	$Growth_{capacity}$

Table 6.2: Relevant variable in MEDEAS

Regrettably, certain values are entirely absent within MEDEAS, rendering their deduction directly from the extant model unfeasible. But since these are mandatory to run the model, a value needs to be provided. Therefore, a constant value will be set.

These have been introduced as parameters into MEDEAS, and their potential impact on the results should be evaluated. Since these may have an influence on the results, this influence will have to be assessed. This falls out of the scope for this thesis and is a lead for further works.

6.4.2 Linkings

The equations that establish links between the input variables from the surrogate model and MEDEAS variables are elucidated in Table 6.3.

Input variable (from surrogate model)	Linking equation
$share_{PV}$	$share_{PV} = \frac{Generation_{PV}}{Demand_{tot}}$
$share_{wind}$	$share_{wind} = \frac{Generation_{wind-onshore} + Generation_{wind-offshore}}{Demand_{tot}}$
$share_{flex}$	$share_{flex} = 40\%$
$share_{storage}$	$share_{storage} = \frac{Storage_{tot}}{Demand_{tot}} \times 365 \times 24$
$Capacity_{ratio}$	$Capacity_{ratio} = \frac{Generation_{tot}}{Demand_{tot}}$
$rNTC$	$rNTC = Growth_{capacity}$

Table 6.3: Variable linking equations. The left-hand side are Dispa-SET variables, and right-hand side MEDEAS variables.

These connections must be implemented within the Vensim model, or incorporated directly into the computation when utilizing the python version of MEDEAS.

7 Results analysis

7.1 Overview

This sections endeavors to describe the results observed when running the MEDEAS model while integrating the created surrogate model, and comparing them with the results previously obtained with MEDEAS.

This comparative assessment is to be conducted across the different scenarios that are defined in MEDEAS, that differ by the evolution of the electricity production mix. These scenarios were detailed in Section 3.6. To streamline the analysis, we narrow the focus on two scenarios: Busines As Usual (BAU) and Optimal Level Transision (OLT). The Mid-Level Transition scenario (MLT), falling between the two, does not contribute significantly to gaining valuable insights.

7.2 Electricity production

The purpose of this comparison is to highlight the impact of integrating the Dispa-SET surrogate model on electricity production values between two scenarios: the default MEDEAS runs and the modified MEDEAS runs integrating the Dispa-SET model.

To make this comparison, four runs are needed, default MEDEAS and modified MEDEAS, integrating the Dispa-SET surrogate model, both with BAU and OLT scenarios.

7.2.1 Photovoltaic units

The photovoltaic electricity production predictions from the four simulations are graphically presented in Figure 7.1.

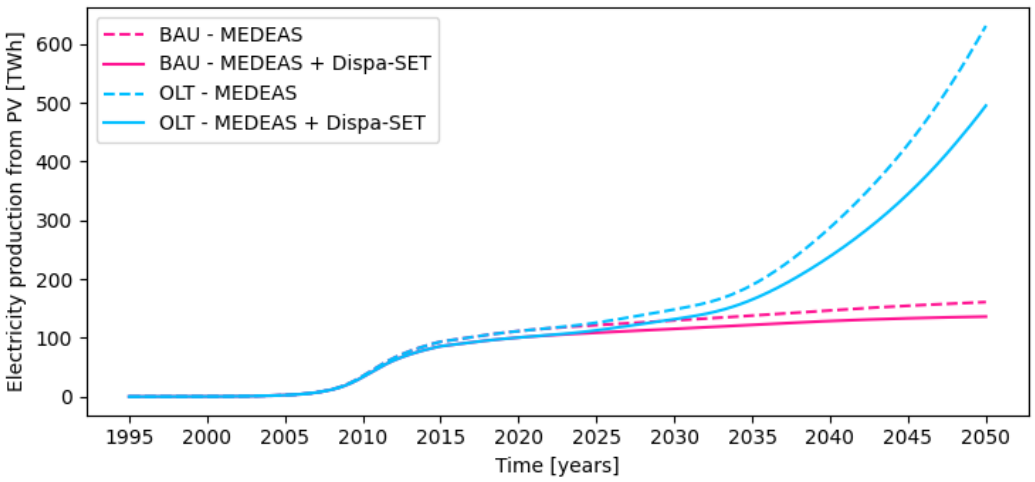


Figure 7.1: Electricity production predictions from photovoltaic units

We can observe that the modified MEDEAS prognosticates a diminished amount of PV energy relatively to default MEDEAS, for both scenarios. In OLT, the difference becomes more pronounced as the production level rises, resembling the effect of linear scaling.

7.2.2 Onshore wind

Figure 7.2 illustrates the varying predictions of the onshore wind production.

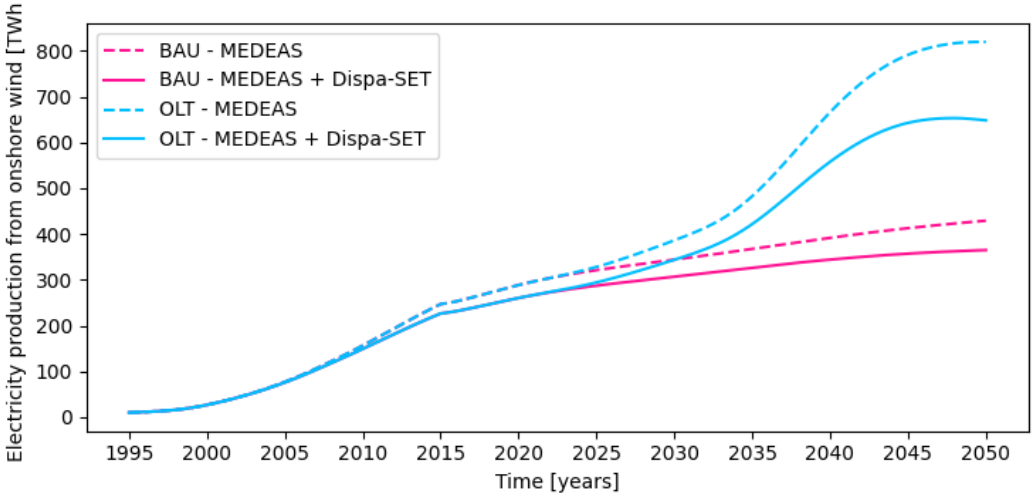


Figure 7.2: Electricity production predictions from onshore wind units

In a similar fashion to the photovoltaic production, the modified MEDEAS outputs what looks like a scaled version of the default MEDEAS output. Notably, a peak is discerned around the year 2050 within the Optimal Level Transition (OLT) scenario employing the default MEDEAS configuration. In contrast, this maximum is slightly shifted to around 2047 when utilizing the modified MEDEAS.

7.2.3 Offshore wind

Predictions of the offshore wind electricity production in the four scenarios considered are given in Figure 7.3.

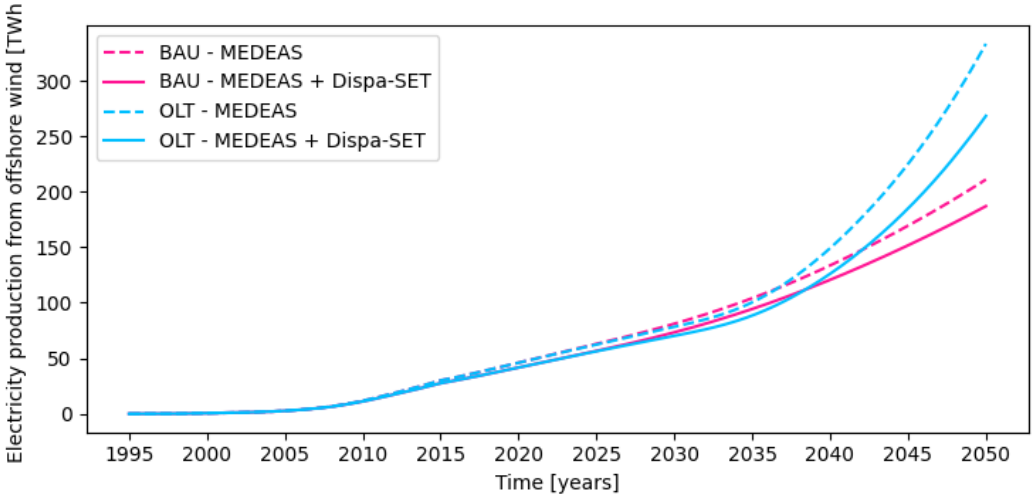


Figure 7.3: Electricity production predictions from offshore wind units

Offshore wind electricity results follow the trend observed previously, that is, the modified

model predicting a smaller amount of electricity and overall as well as a larger production decrease in OLT.

7.2.4 Hydroelectricity

The results for hydroelectricity production across the four scenarios are depicted in Figure 7.4.

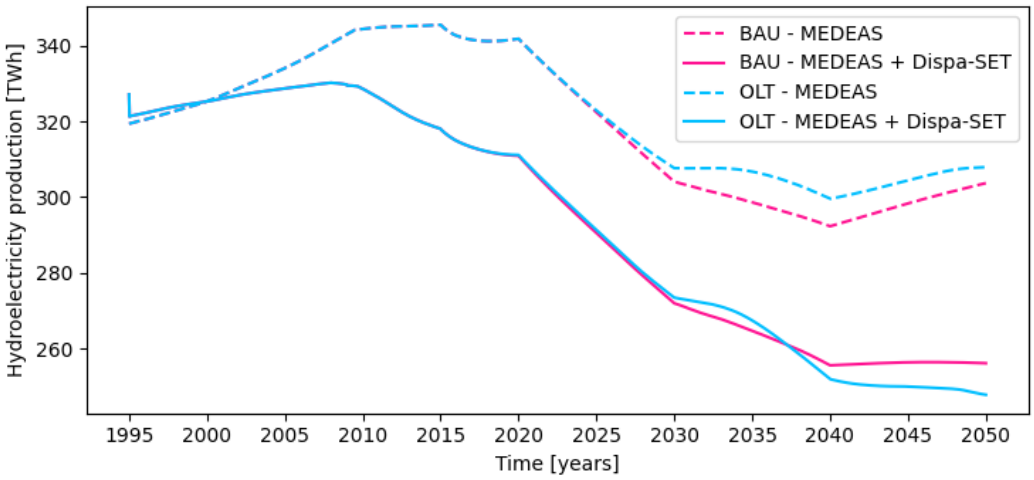


Figure 7.4: Electricity production predictions from hydroelectric units

Contrary to the previous results, where the results are grouped by scenario, that is, the shape of the curve is dictated by the scenario then is slightly changed by the model, these outcomes are grouped by model. In contrast, the outputs of the modified MEDEAS model for both the BAU and OLT scenarios are closely aligned with each other but distinctly diverge from the outputs of the default MEDEAS model.

This phenomenon can be attributed to the fact that hydroelectricity is not favored by Dispa-SET, therefore these units are avoided when possible, hence leading to a smaller use. This may be due to the geographical constraints these units are suject to, limiting their growth are there is no spot to build new units.

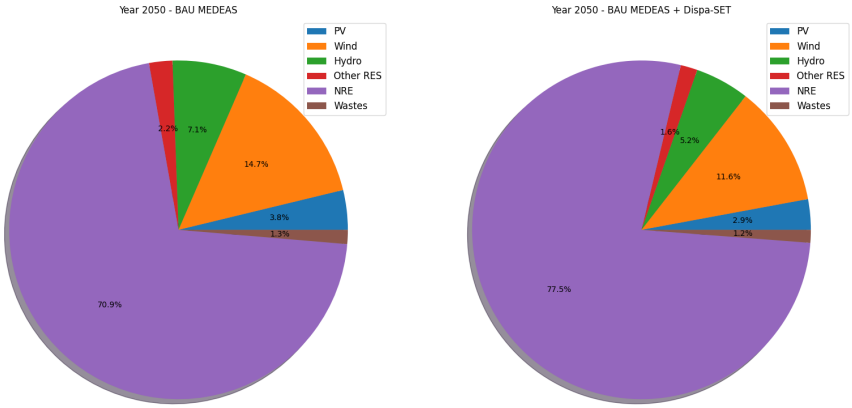
These aspects also pertain to pumped hydro-storage units, as such installations could potentially be erected on rivers. On average, the unit produces the amount of electricty dictated by the river's flow, but the total energy produced over a set period is dependent on the specific dispatch of that unit, used as a tool to manoeuvre the electricity network.

7.2.5 Electricity mix

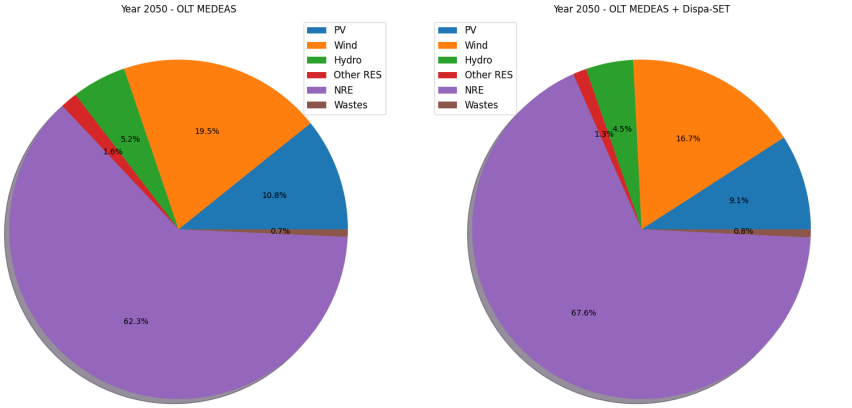
Figure 7.5 exhibits the different predictions for the electricity mix in 2050.

Similar to the earlier observation revealing reduced VRES production, the proportion of VRES in the overall electricity mix also diminished in the modified version of the MEDEAS model.

A noteworthy change is the decrease in the share of hydroelectricity in the OLT scenario compared to the BAU scenario, for both default and modified MEDEAS. This can likely be attributed to the overall higher total production in the OLT scenario, leading to a relatively decreased share as hydroelectricity generation remains constant.



(a) BAU using default MEDEAS (b) BAU using modified MEDEAS



(c) OLT using default MEDEAS (d) OLT using modified MEDEAS

Figure 7.5: Electricity mix projection in 2050 for the four considered cases

7.3 Curtailment

As the surrogate model explicitly outputs the curtailment, this variable can then be plotted over a run of the modified MEDEAS. The results obtained for both scenarios are displayed in Figure 7.6.

We can observe from Figure 7.6 that the OLT scenarios suffers from higher proportions of curtailment, with an higher growth rate, than in the BAU scenario. Nonetheless, this increased rate appears to demonstrate stability.

It might be concluded that curtailment is inevitable, nonetheless every option has not been covered. For instance, the consideration of various categories of storage units, predicated on the ratio of storage capacity to power output, has been notably absent from the analysis. Given that storage facilities wield a substantial impact on curtailment dynamics, the pursuit of a more precise modeling approach for these entities could potentially yield valuable insights.

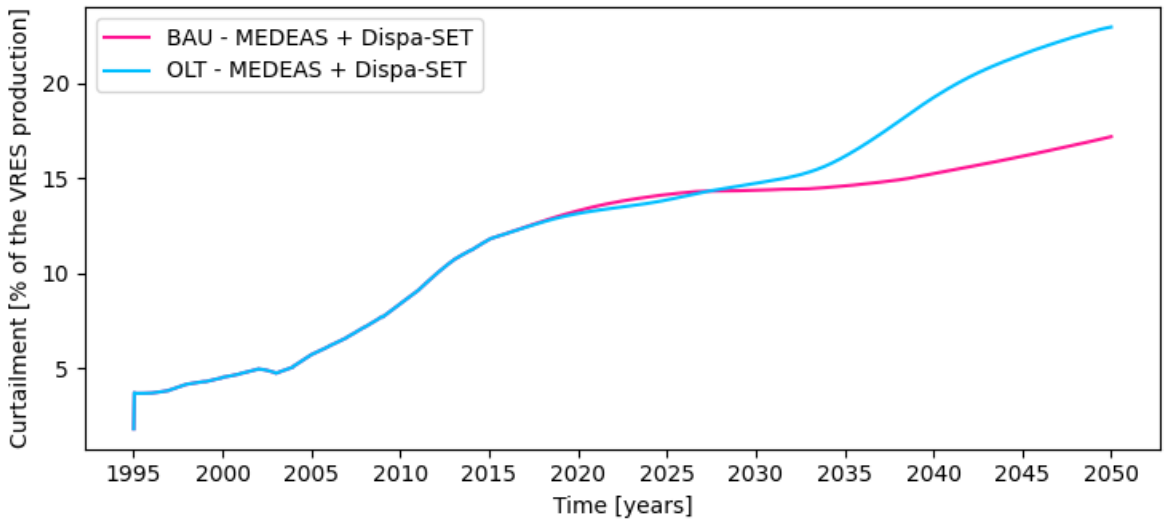


Figure 7.6: Curtailment prediction of the modified MEDEAS for BAU and OLT scenarios

7.4 Discussion

The figures presented consistently conclude that integrating the surrogate model, i.e. taking power systems constraints into account, leads to a lower prediction of electricity production from VRES. This outcome is somewhat counterintuitive, as the consideration of the increasing part of curtailment suggest larger amounts of wasted energy.

A plausible interpretation aligning with the observations is that due to the imposition of more restrictive constraints, the deployment of VRES is more challenging than initially estimated. Consequently, since greater efforts are required to achieve the same share of renewable energy production, e.g. because more storage unit need to be built, equivalent efforts result in a diminished proportion of VRES.

8 Conclusion

The objective of this thesis was to assimilate refined flexibility constraints, assessed via the Dispa-SET model, into an Integrated Assessment Model (IAM) known as MEDEAS, employing a strategy involving the integration of a surrogate model. This approach has been effectively executed, accompanied by judicious selections of input and output variables.

Then, the resulting modified version of MEDEAS has been run and observations have been made. In particular, they predicted lower shares of VRES in both optimal effort and business as usual scenarios.

In the process, a database of simulations has been created on points generated through latin hypercube sampling. The complete workflow has been automated and run on a cluster.

Once the database was available, it served as the starting point to train the surrogate model. The best method was selected after several had been reviewed. Although neural networks proved superior performance, XGBoost, among gradient boosting techniques, remained highly competitive.

Finally, the linking of the model to MEDEAS has been performed, by creating an external function in Vensim, in which it is described. Collaboratively with J. Paris, the model was incorporated into MEDEAS, establing connections between the surrogate model and MEDEAS variables.

8.1 Future work

In this work, the core mechanisms for the integration have been implemented. Therefore, the most interesting contributions lie in the accuracy of surrogate model and in the quality of choice of variable. Although six inputs and two outputs have been employed, the potential for enhancing accuracy lies in refining these choices. For example, adding one input to account for different kinds of storage facilities.

Concerning the data generation process, there remain a need to explore alternative LHS parameterizations. Another influential parameter whose impact has not been assessed is the number of simulations, that has been arbitrarily set to 2400 in this thesis.

Regarding the surrogate model creation, we discussed in Subsection 5.3.1 the eventual use of multiple LHS to generate multiple datasets. Delving into these alternatives could potentially contribute to the refinement of the training process. a notable observation from Subsection 5.4.3 pertains to one of the depicted surfaces, specifically Figure 5.3d, exhibiting swift transformations when the remaining input parameters are altered. This suggests some weakness in the learning of the surrogate model that could be explored.

Additionally, specific links between the surrogate model and MEDEAS have not been established, towards the share of flexible units and the rNTC inputs, given as constants. Evaluating the impact of their values on the results would bring valueable information. Moreover, relationships could be developed and incorporated into the MEDEAS model to account for these variables.

References

[1] OpenAI. *ChatGPT*. Version August 2023. 2022. URL: chat.openai.com/chat.

[2] IPCC. "Climate Change 2022: Impacts, Adaptation and Vulnerability". In: *Climate Change 2022: Mitigation of Climate Change. Contribution of Working Group III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Ed. by P.R. Shukla et al. Cambridge, UK and New York, USA: Cambridge University Press, 2022. DOI: 10.1017/9781009157926.

[3] *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.

[4] O.M. Babatunde, J.L. Munda, and Y. Hamam. "Power system flexibility: A review". In: *Energy Reports* 6 (2020). The 6th International Conference on Power and Energy Systems Engineering, pp. 101–106. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2019.11.048. URL: https://www.sciencedirect.com/science/article/pii/S2352484719309242.

[5] Wei Lin, Zhifang Yang, and Juan Yu. "Flexibility of interconnected power system operation: Analysis, evaluation and prospection". In: *Energy Conversion and Economics* 1.3 (2020), pp. 141–150. DOI: https://doi.org/10.1049/enc2.12013. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/enc2.12013. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/enc2.12013.

[6] Umar Taiwo Salman et al. "A Review of Improvements in Power System Flexibility: Implementation, Operation and Economics". In: *Electronics* 11.4 (2022). ISSN: 2079-9292. URL: https://www.mdpi.com/2079-9292/11/4/581.

[7] Stefan Pfenninger, Adam Hawkes, and James Keirstead. "Energy systems modeling for twenty-first century energy challenges". In: *Renewable and Sustainable Energy Reviews* 33 (2014), pp. 74–86. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2014.02.003. URL: https://www.sciencedirect.com/science/article/pii/S1364032114000872.

[8] M. Doquet et al. "A new tool for adequacy reporting of electric systems: ANTARES". In: *42nd International Conference on Large High Voltage Electric Systems 2008, CIGRE 2008* (Jan. 2008).

[9] 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).

[10] K. Calvin et al. "GCAM v5.1: representing the linkages between energy, water, land, climate, and economic systems". In: *Geoscientific Model Development* 12.2 (2019), pp. 677–698. DOI: 10.5194/gmd-12-677-2019. URL: https://gmd.copernicus.org/articles/12/677/2019/.

[11] V. Krey and P et al. Havlik. *MESSAGEix-GLOBIOM Documentation – 2020 release*. Tech. rep. Laxenburg, Austria: International Institute for Applied Systems Analysis (IIASA), 2020. DOI: 10.22022/iacc/03-2021.17115. URL: https://pure.iiasa.ac.at/id/eprint/17115.

[12] Valérie Masson-Delmotte and Intergovernmental Panel on Climate Change Working Group I. *Global warming of 1.5°C : an IPCC special report on the impacts of global warming of 1.5 °C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty*. eng. Cambridge, United Kingdom, 2022.

[13] Iñigo Capellán-Pérez et al. "MEDEAS: a new modeling framework integrating global biophysical and socioeconomic constraints". In: *Energy Environ. Sci.* 13 (3 2020), pp. 986–1017. DOI: 10.1039/C9EE02627D. URL: http://dx.doi.org/10.1039/C9EE02627D.

[14] Matija Pavicevic et al. "Bi-directional soft-linking between a whole energy system model and a power systems model". English. In: *2022 IEEE PES/IAS PowerAfrica, PowerAfrica 2022*. Kigali, Rwanda: Institute of Electrical and Electronics Engineers Inc., 2022. DOI: 10.1109/PowerAfrica53997.2022.9905392. URL: http://xplorestaging.ieee.org/ielx7/9904951/9905072/09905392.pdf?arnumber=9905392.

[15] Per Helgesen et al. "Using a hybrid hard-linked model to analyze reduced climate gas emissions from transport". In: *Energy* 156 (May 2018). DOI: 10.1016/j.energy.2018.05.005.

[16] Gonzalo Parrado-Hernando et al. "Capturing features of hourly-resolution energy models through statistical annual indicators". In: *Renewable Energy* 197 (2022), pp. 1192–1223. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2022.07.040. URL: https://www.sciencedirect.com/science/article/pii/S0960148122010357.

[17] Maarten Brinkerink et al. "Assessing global climate change mitigation scenarios from a power system perspective using a novel multi-model framework". In: *Environmental Modelling and Software* 150 (2022), p. 105336. ISSN: 1364-8152. DOI: https://doi.org/10.1016/j.envsoft.2022.105336. URL: https://www.sciencedirect.com/science/article/pii/S1364815222000421.

[18] J.P. Deane et al. "Soft-linking of a power systems model to an energy systems model". In: *Energy* 42.1 (2012). 8th World Energy System Conference, WESC 2010, pp. 303–312. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2012.03.052. URL: https://www.sciencedirect.com/science/article/pii/S0360544212002551.

[19] Sylvain Quoilin, Ignacio Hidalgo Gonzalez, and Andreas Zucker. *Modelling Future EU Power Systems Under High Shares of Renewables: The Dispa-SET 2.1 open-source model*. English. Publications Office of the European Union, 2017. ISBN: 978-92-79-65265-3. DOI: 10.2760/25400. URL: https://ec.europa.eu/jrc/en/publication/eur-scientific-and-technical-research-reports/modelling-future-eu-power-systems-under-high-shares-renewables-dispa-set-21-open-source.

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

[21] medeas.eu. *MEDEAS home page*. Accessed in May 2023. URL: https://medeas.eu.

[22] Donella H. Meadows, Jorgen Randers, and Dennis L. Meadows. "The Limits to Growth (1972)". In: *Documents of Global Change*. Ed. by Libby Robin, Sverker Sörlin, and Paul Warde. New Haven: Yale University Press, 2013, pp. 101–116. ISBN: 9780300188479. DOI: doi:10.12987/9780300188479-012. URL: https://doi.org/10.12987/9780300188479-012.

[23] Wikipedia. *Energy return on investment*. Accessed in May 2023. URL: https://en.wikipedia.org/wiki/Energy_return_on_investment.

[24] MEDEAS consortium. *Deliverable 6.2. Costs of the transition*. 2018. URL: https://medeas.eu/system/files/documentation/files/D6.2%28D20%29_Transition%20Costs.pdf.

[25] Iñigo Capellán-Pérez, Carlos de Castro, and Luis Javier Miguel González. "Dynamic Energy Return on Energy Investment (EROI) and material requirements in scenarios of global transition to renewable energies". In: *Energy Strategy Reviews* 26 (2019), p. 100399. ISSN: 2211-467X. DOI: https://doi.org/10.1016/j.esr.2019.100399. URL: https://www.sciencedirect.com/science/article/pii/S2211467X19300926.

[26] NREL. *Renewable Electricity Futures Study (Entire Report)*. Golden, CO, USA: National Renewable Energy Laboratory, 2012.
[27] Erik Delarue and Jennifer Morris. *Renewables Intermittency: Operational Limits and Implications for Long-Term Energy System Models*. MIT Joint Program on the Science and Policy of Global Change, 2015. URL: http://hdl.handle.net/1721.1/95762.
[28] 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.
[29] mit.edu. *Optimal Latin Hypercube Technique*. Accessed in May 2023. URL: https://abaqus-docs.mit.edu/2017/English/IhrComponentMap/ihr-c-Reference-OptimalLatin.htm.
[30] Wikipedia. *Latin hypercube sampling*. Accessed in May 2023. URL: https://en.wikipedia.org/wiki/Latin_hypercube_sampling.
[31] pyDOE documentation. *Randomized designs*. Accessed in May 2023. URL: https://pythonhosted.org/pyDOE/randomized.html#latin-hypercube.
[32] P. Geurts and L. Wehenkel. "Introduction to Machine Learning". ELEN062-1 class. 2021.
[33] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
[34] D. Ernst P. Geurts and L. Wehenkel. "Extremely randomized trees". In: *Machine learning* (2006). URL: https://doi.org/10.1007/s10994-006-6226-1.
[35] Tianqi Chen and Carlos Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2016. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145%2F2939672.2939785.
[36] Carla Vidal Montesinos. *Integrating short-term dispatch constraints in a system dynamics energy planning model*. 2022. URL: http://hdl.handle.net/2268.2/16566.
[37] G. Louppe. "Deep learning". INFO 8010 class. 2022.
[38] Eneko Martin-Martinez et al. "PySD: System Dynamics Modeling in Python". In: *Journal of Open Source Software* 7.78 (2022), p. 4329. DOI: 10.21105/joss.04329. URL: https://doi.org/10.21105/joss.04329.
[39] Inc. Ventana Systems. *Vensim software*. Accessed in May 2023. URL: https://vensim.com/vensim-software/.
[40] Wikipedia. *Vensim*. Accessed in May 2023. URL: https://en.wikipedia.org/wiki/Vensim.
[41] B.A. Williams and S. Cremaschi. "Surrogate Model Selection for Design Space Approximation And Surrogatebased Optimization". In: *Proceedings of the 9th International Conference on Foundations of Computer-Aided Process Design*. Ed. by Salvador Garcia Muñoz, Carl D. Laird, and Matthew J. Realff. Vol. 47. Computer Aided Chemical Engineering. Elsevier, 2019, pp. 353–358. DOI: https://doi.org/10.1016/B978-0-12-818597-1.50056-4. URL: https://www.sciencedirect.com/science/article/pii/B9780128185971500564.
[42] Tom O'Malley et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.
[43] medeas.eu. *MEDEAS Model, Structure of MEDEAS-World model*. Accessed in August 2023. URL: https://medeas.eu/model/medeas-model.
[44] J. P. Dietrich et al. "MAgPIE 4 – a modular open-source framework for modeling global land systems". In: *Geoscientific Model Development* 12.4 (2019), pp. 1299–1317. DOI: 10.5194/gmd-12-1299-2019. URL: https://gmd.copernicus.org/articles/12/1299/2019/.

Annex A: scripts and code

This annex documents briefly the roles of each scripts and code files.

Data generation

See Table 9.1.

File name	Description
`config.py`	Holds high level specification of the dataset to be created, such as the number of samples, the LHS strategy, and output files names.
`read_results.py`	Fetches the results from simulation directories, either one by one or all at once.
`reference.py`	Runs the reference simulation and serializes the results in a json file.
`sampling.py`	`-sample-only`: only run the LHS and store the samples in a CSV file. `-prepare-one idx`: prepares the simulation directory for one sample given its index. With no arguments, runs LHS and prepare all the simulation directories.
`utils_francois.py`	Stores the modified version of the `adjust_capacity` function of DispaSET, taking into account different storage unit types.
`main.sh`	Starts all the scripts in the right order in order to produce a dataset. Runs the reference simulation, the sampling, prepends the header to the dataset file (as CSV), and starts the first series.
`launch-job-series.sh`	Submits a series of simulation jobs, and a job that will submit the following series with the current one as a dependency. If the series index given as argument is too high, exits.
`launch-reference-job.sh`	Submits a job that runs the reference simulation.
`launch-simulation-jobs.sh`	Submits the jobs required to run a simulation from a series. It takes the series index as an argument and the number in that series from SLURM environment variables. Uses `sampling.py -prepare-one`, GAMS and `read_results.py` successively.
`gams-simulation.sh`	Submits a job running the GAMS simulation of an already prepared simulation, for testing purposes.
`read-one.sh`	Submits a job fetcing the results of an arleady ran GAMS simulation, for testing purposes.
`get-longest-simulation.sh`	Bash script that calls the `seff` utility on each of the simulation ran to extract the longest ones. This has mainly be used once to set the simulation timeout that prevent stalling simulation to waste resources.

Table 9.1: Description of the data-generation files

Neural network

See Table 9.2.

File name	Description
`config.py`	Holds the configuration of the network to be trained, name, data to use, inputs and outputs.
`model.py`	Holds the description of the model to be trained, via the `build_model` function.
`baselines.py`	Builds and trains different, predefined neural network architectures, and stores their performance. This eases the process of looking for a good architecture for the ANN, to guide the bounds in the hyper-parameter tuning. With `sort -k2 -t '>' -i logsbaselines-results.txt` one easily sorts the results by increasing order.
`train.py`	Executes the tuner search for the best model and training of that best model.
`view.py`	Holds different utilities to view the results of some model and its performance. Use `view.py -surface <in1> <in2> <out>` to create a 3D surface of the out-th output depending on the in1 and in2-th inputs. The other inputs are constant and parameterizable with sliders.

Table 9.2: Description of the files for the neural network part.

Integration

See Table 9.3.

File name	Description
`external.h`	Header file for `external.cpp`.
`external.cpp`	Main source file for the library.
`main.cpp`	Code for running a test program.
`Makefile`	GNU make file for automating compilation.
`tensorflow.dll`	Tensorflow library file for Windows, can be downloaded from here.

Table 9.3: Description of the integration files.

Annex B: 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, that is, the zones simulated correspond approximately to the European Union.

Zones

Most of the EU contries are represented, for completeness they are reported in Table 9.4.

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 9.4: 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.

Technologies

Table 9.5 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.

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

However, hydroelectric units with a reservoir are have some room for flexibility, due 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.

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 9.5: Technologies present in Dispa-SET

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

Fuels

Table 9.6 summarizes the fuel types 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. This grouping is of crucial importance, as it will define the behaviour of the simulation when using the MILP formulation (see **??**).

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 9.6: Fuel types in Dispa-SET

A major consideration for the optimization problem is the fuel prices, which are listed in Table 9.7.

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 9.7: Fuel prices considered, in €/MWh

Other prices

Some other price values are relevant, such as the price of the load shedding per MWh. These are presented in Table 9.8.

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

Table 9.8: Other relevant prices, in €/MWh

These values define the significance of each problem relatively to each other, hence what option is the least costly. For instance, the shedding cost could be so low compared to the carbon emissions that it is preferable not to run any coal unit to produce 1MW than to shed 1MW. This example is extreme, but outlines the fact that these prices impact the simulation outcome via their use in the objective function.

Power plants

As a dispatch model, Dispa-SET evidently 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 [9].

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 in Table 9.9.

For the storage units, one needs some more parameters, given in Table 9.10. These feilds will remain blank for the other unit types.

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 ramping up or down	value in €/hour
StartUpCost_pu	Start up cost per clustered unit	value in €
NoLoadCost_pu	Cost of having no load on a unit	value in €/hour
PartLoadMin	Ratio of the minimum nominal capacity	real in [0,1]
StartUpTime	Time to start up the plant	value in hour
CO2Intensity	Amount of CO_2 emitted per MW	value in €/MW

Table 9.9: The table fields used to describe a optionnally aggregated power plant unit

Field	Description	Type
STOCapacity	The total energy storage capacity	value in MWh
STOSelfDischarge	The discharge rate (w.r.t. to the total)	value in day^{-1}
STOMaxChargingPower	Maximum energy inflow	value in MW
STOChargingEfficiency	The unit's charging efficiency	real in [0,1]

Table 9.10: Fields describing the storage capabilities of the units

Their discharge efficiency will be assigned to the common Efficiency field, and the PowerCapacity will be assigned the power output on discharge.

For batteries units, the RampUpRate and RampDownRate fields are set to 1, while the others but efficiency are set to 0.

The crucial capability of storage units is their storage volume. In previous work in this context, the number of hours a unit can run at maximum output capacity is fixed as 4 hours, thus implicitly fixing a storage capacity given a power output.

This choice is arbitrary and leads to a simplification of the reality, where one could find huge differences in this ratio. To remove this, an option is added in Dispa-SET's adjusting function, to be able to filter the adjustments by range, making it now able to discriminate the units based on the storage capacity over maximum output power ratio, enabling its use to adjust storage units with different "longevity" separately.

However in reality, most of the difference is between the pumped hydro storage, that can typically output their maximum power for a longer time, and the other storage technologies, such as batteries.

At the end, this differenciation is not done, as it would also require the inputs of the

surrogate model to be changed, to take into account the share of "high-longevity" storage units with respect to the "low-lengevity" ones.

Notes on the other inputs

We can make a few miscellaneous remarks about Dispa-SET input data.

- 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.
- 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.

  In this work, AF denotes the hourly capacity factor, and CF (Capacity Factor) refers to the annual value. Hence, the capacity factor is the yearly average of the availability factor.

Annex C: Vensim models

Vensim offers a variety of tools to describe models, but at the end every model is an interconnection of variables, the math hiding in the connections between these.

Vensim provides the following types of variables:

- **Auxiliary variables**, that are regular variable that have no memory, that is, are independent from their value at the previous time step and are computed from every type of variable.

  For example, a *temperature* variable that is computed from some *sunshine* and *latitude*, that is used to compute the birth rate of $rabbits$ and $foxes$.

- **Constant variables**, that hold one value.

  For example, a mathematical constant like π.

- **Data variables**, or exogenous variables, whose value evolve over time but is not dependent of the model.

  For example, typical *sunshine* data over a year.

- **Stock variables**, that change only over time as a function of the incoming rates, i.e., they integrates the rates.

  For example, the population of some species at a given time.

- **Rate variables**, or flows, that directly impact the Stock variables.

  For example, the birth or death rate of some population at a given time.

The connections between the variables are virtually done by arrows. The only practical use of arrows is to make the variable at the origin appear in the selection of variables in the variable equation screen for the variable pointed by the arrow. But obviously they are of great utility in terms of visualization of the model.

An illustrative example of a Vensim model is depicted in Figure 9.1.

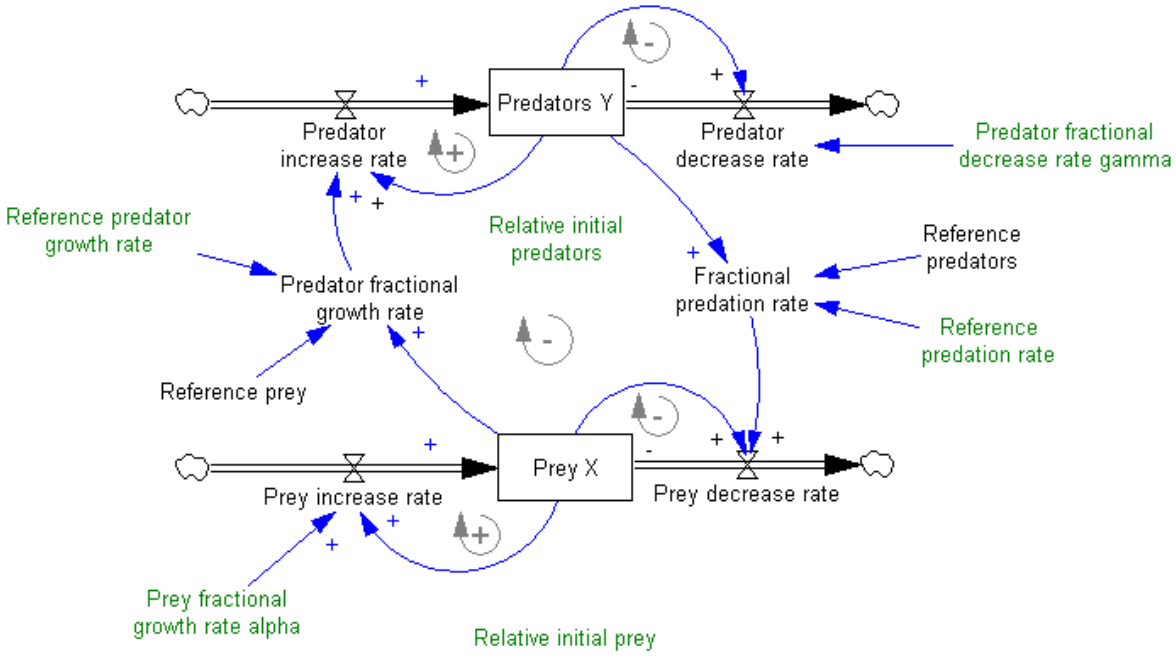


Figure 9.1: An example model in Vensim: Lotka-Volterra predator-prey model.