University of Liège - School of Engineering and Computer Science

MASTER'S THESIS

Improving the simulation of variable renewable energy in integrated assessment models

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

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

Academic year 2022-2023

Keywords

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

Abstract

TODO

Acknowledgements

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

TODO

Todo-list

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

Contents

Warnings

I have 11 warnings:

– `Warning` No.1

– `Warning` No.2

– `Warning` No.3

– `Warning` No.4

- `Warning` No.5
- `Warning` No.6
- `Warning` No.7
- `Warning` No.8
- `Warning` No.9
- `Warning` No.10
- `Warning` No.11

1 Introduction

Our societies run on energy. Energy is required to power pretty much whatever we do and need. Fossil fuels provided us with relatively easy to access, store and use energy for decades, bringing a solution to a problem.

Now, the awareness of the climate change, mostly caused by the release of large amounts of greenhouse gases, among them carbon dioxyde, produced by the combustion of these fossil fuels, is a game-changer. In order to acheive set targets in terms of climate evolution, bounds on carbon emissions have been defined, and therefore limiting the prevously endless source of fossil energy in the very long run.

The solutions to compensate for the lacking energy generation, that from now on should not originate from fossil resources, are renewable energy. These refer to every energy generation technique originating from a renewable source, such as the sun, the wind and the rivers. However, the creation of these units may not be completely renewable in itself, 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, pays off the invesment made at their construction.

In this context, a significant increase in the energy produced from such energy sources is to be expected, in particular, from:

- the sun, through photovoltaic panels,
- the wind, through on-shore and off-shore wind turbines, and
- rivers, through hydroelectric dams.

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.

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 becomes a huge deal, because the amount of energy that can be generated by exploiting these is variable, hence their designation as variable renewable energy sources, VRES in short.

It is to be mentioned that these variabilities comprise some 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 also be take advantage of in order to predict wind turbines' production.

The larger variability of the power output of VRES is causes a problem, because modern electrical system is dictated by consumers' demand. And to do so, electricity generators are dispatched in real-time, so that the production always matches the demand on the network. This technique works because the concerned power plants can be started and shut down on demand, almost at any time. With VRES, this assumption becomes false, as one cannot start an extra wind turbine if there is no wind, nor use a photovoltaic panel during the night.

1.1 The first side

There exists 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. 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.

Among these tools, the Dispa-SET model will be considered in 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.

1.2 The other side

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 most 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.

In this subset of IAM, the MEDEAS model is chosen for this work, being open-source as well and providing a EU-specific 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 comes down to solving a set of differential equations.

1.3 The linking

As MEDEAS does not extensively simulates the dispatch of power plants units for the sake of its estimate of the curtailment and lost load, that are used in the model, its estimations of these values are not as precise as those computed by Dispa-SET. Therefore, MEDEAS would benefit from a linking between the two models.

An high-level illustration of the positionning of MEDEAS and Dispa-SET on the timescale is provided on Figure 1.1 [8].

1.3.1 Linking types

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

– Soft linking: the models communicate between each other with feedback. 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 there is no guarantee of converging.

– Hard linking: the models are combined into a single, unified mathematical formulation. This newly created model can then be solved all at once. But this comes with a lower chance of computational feasibility.

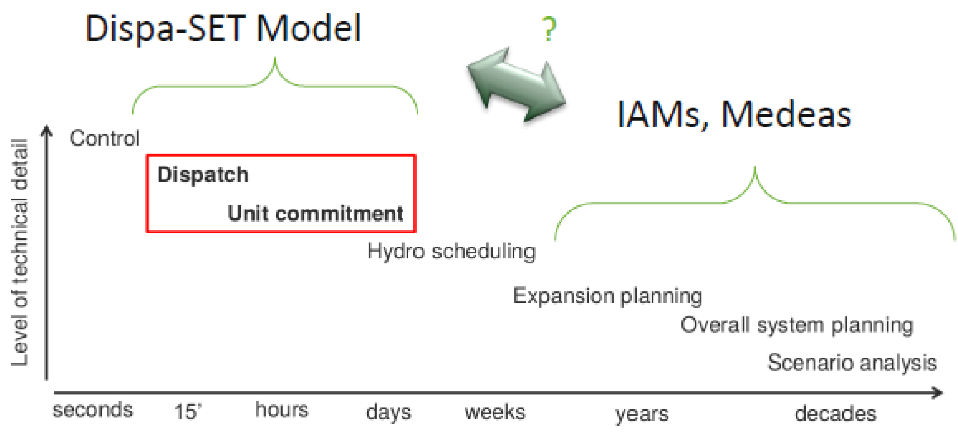


Figure 1.1: An illustration of where Dispa-SET and MEDEAS operate on the timescale

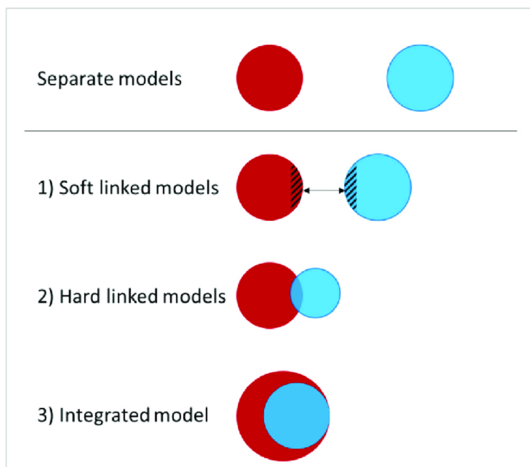


Figure 1.2: Illustration of the different linking methods [7].

These linking types are illustrated on Figure 1.2.

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 due to the different formulations of Dispa-SET, in linear programming, and MEDEAS, in differential equations.

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

1.3.2 Surrogate models

As there are no good, viable linking options between Dispa-SET and MEDEAS, surrogate models are considered.

The idea behind surrogate models is simple. First, a fast, easy-to use 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 apprximated from relevant inputs regarding MEDEAS, then the approximator will be integrated in the model.

1.4 State of the art in flexibility assessment

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 changes in the demand and supply, either expected or not [16].

To improve the flexibility of an existing power system, some mechanisms already exists.

- Use of the regular dispatchable energy production plants to mitigate the energy not produced from VRES. Depending on the plant characteristics, it may be difficult to address short term drops in production, as there is some start up time required.

- 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.

- 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.

- 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 it may concern other domains, such as electric vehicles charge, heating and cooling etc.

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.

1.5 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, being less precise on the matter. This includes the creation of a proper dataset, the definition, training and integration of the surrogate model into MEDEAS.

1.5.2 Methodology

To do so, adequate design of experiment will be conducted on well-chosen input parameters, that will in fine be the surrogate model input features. This will give us a set of input points where a Dispa-SET run will be made, to create a training sample, and the samples are grouped into a single dataset.

Once this dataset is built, one can then train the surrogate model on it. This surrogate model is chosen after a short review of candidate machine learning options. It will be adequately fine tuned using hyper-parameter tuning.

When the model will be available, it will then be usable from inside MEDEAS, where some connection will have to be drawn between the variables that are already present in the model, and the surrogate model actual input and output features.

Finally, the resulting adapted MEDEAS model will be run, and observations and criticisms will be made.

1.5.3 Contributions

This works 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.

Furthermore, the present thesis also considers other machine learning algorithms, although they are not actually implemented, the choice of neural networks is justified.

This work constisted in:

- The writing of easy-to-use scripts to run Dispa-SET on those experiments
- The efinition and implementation of an adequate machine learning model (neural net), and training
- The integration the model in MEDEAS, by writing a C++ external function library for Vensim
- Runs and analysis of the improved MEDEAS model

All the produced work, and necessary data is available on the online github repository at the following address: https://github.com/Rayerdyne/master-thesis[1]

1.5.4 Outline

This document is structured as follows:

1. **The Dispa-SET model**: description of the Dispa-SET model and tools
2. **The MEDEAS model**: description of the MEDEAS model and framework
3. **Data generation**: description of the complete process to generate the training data, that is, the design of experiments and the Dispa-SET runs
4. **The surrogate model**: definition of the surrogate model, training and validation
5. **Integration**: description of the process of integrating a model into MEDEAS

[1]As it contains licensed file from Vensim, this repository is private. One may ask reading permission by email at f.straet@student.uliege.be.

6. **Analysis**: the analysis of the runs of MEDEAS with the integrated surrogate model
7. **Conclusions**

2 The Dispa-SET model

2.1 Overview

The online documentation describes the Dispa-SET [8] model as ”an open-source unit commitment and optimal dispatch model focused on the balancing and flexibility problems in European grids”.

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

A schematic of the Dispa-SET architecture is given in Figure 2.1.

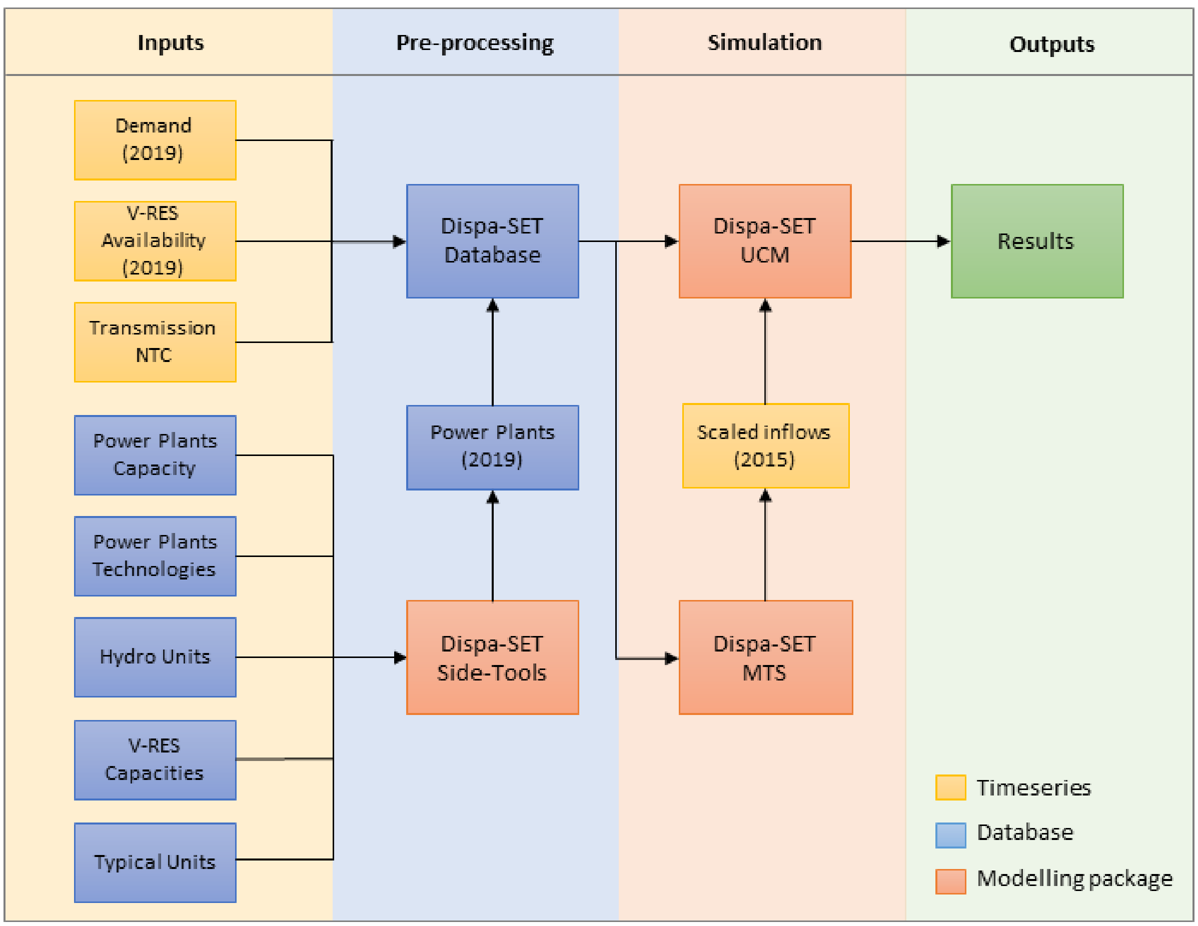


Figure 2.1: Block diagram of Dispa-SET architecture

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

– Linear programming (LP) optimization problem

– Mixed Integer Linear Programming optimization problem.

Its interface is written in the Python programming language, and calls GAMS [3] as the main solver engine.

2.2 Objective function

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

The total system costs is splitted as follows:

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

Adding these, Equation 2.1 is obtained, where u refers to the index on each units, and i is the time index.

2.3 Supply and demand balance

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

The supply sources are:

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

Whereas the demand originates from:

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

The Equation 2.2 expresses this target equality.

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

Equation 2.1: Objective function of the Dispa-SET model

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

Equation 2.2: Supply-demand balance in Dispa-SET

2.4 Rolling horizon

I'm not sure I understood all

2.5 Mid-term scheduling (MTS)

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

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

Talk about MTS option

2.6 Dispa-SET components and representation

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

In this work, the european setting is used.

2.6.1 Zones

Most of the EU contries are represented, for completeness they are reported on Table 2.1.

2.6.2 Technologies

Table 2.2 lists all the technologies taken into account by Dispa-SET, alongside with their main properties:

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

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

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

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

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

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

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

2.6.3 Fuels

Table 2.3 summarizes the fuel types in Dispa-SET.

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

Table 2.2: Technologies present in Dispa-SET

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

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

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

Table 2.3: Fuel types in Dispa-SET

A major consideration for the optimization problem is the fuel prices, that are summarized in Table 2.4.

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

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

Table 2.4: Fuel prices considered[€/MWh]

2.6.4 Other prices

Some other price values are relevant, such as the price of the load shedding per MWh. These are presented in Table 2.5.

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

Table 2.5: Other relevant prices [€/MWh]

2.6.5 Power plants

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

For performance reasons, some of the units initially described are merged into clustered units at the pre-processing step. Thus, the amount of variable in the simulation is reduced, while the accuracy is not significantly impacted [8].

Dispa-SET disposes of utilities to do so, but also needs to craft the new, aggregated units properties table. These are defined by the set of feilds that are shown on Table 2.6.

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

For the storage units, one needs some more parameters, given on Table 2.7. For the other unit types, these feilds will be left empty.

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. In previous work in this context, the number of hours a unit can run

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 2.7: Fields describing the storage capabilities of the units

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 comes from the pumped hydro storage, that can typically output their maximum power for a longer time than 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.

2.6.6 Notes on the other inputs

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

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

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

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

  Non-renewable technologies have their AF set to 1.

3 The MEDEAS model

3.1 Overview

This sections aims at describing the main components of the MEDEAS model and its components, with a focus on the parts that are the most relevant in this context.

The main flaw of MEDEAS, in its energy module, is its low temporal precision and poor assessment of the flexibility of the energy network. *Missing quote* The solution that this work will provide is the integration of the surrogate model for that purpose, that will be in charge of computing better approximation of (mainly) curtailment and lost load.

For contextualisation, the following description of MEDEAS is written.

3.2 Integrated assessment models

MEDEAS is an open-source integrated assessment model (IAM), built to "guide the transition to a low carbon European socio-economy" [10].

Integrated assessment models are used to make general purpose analysis, combining aspects from different disciplines, 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.

There is a large variety of IAMs, because there are a lot of ways to model the complex interactions, and the large amount of uncertainties between the different components of a model. Hence, there exist a lot of different approaches to the creation of an IAM.

As a side note, 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 may be used from the Python programming language through the `pySD` package.

3.3 MEDEAS Overview

MEDEAS funds come from the EU for its Horizon 2020 program, under the "Modelling and analysing the snergy 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 interactions between the society, that is expected to follow some policy. A choice of policy by the society is also called a scenario.

It is formulated using the system dynamics toolset—what Vensim is built for—which is to help the aggregation of the knowledge from different experts from different domains and backgrounds. This will also enable easy modelling of feedback between the different components.

MEDEAS typically sets the simulation horizon between 1995 and 2060, while for longer term analysis it may 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, that are economy, energy, energy infrastructures, materials, land use, climate change and socio-environmental impacts indicators. The general structure is illustrated on Figure 3.1.

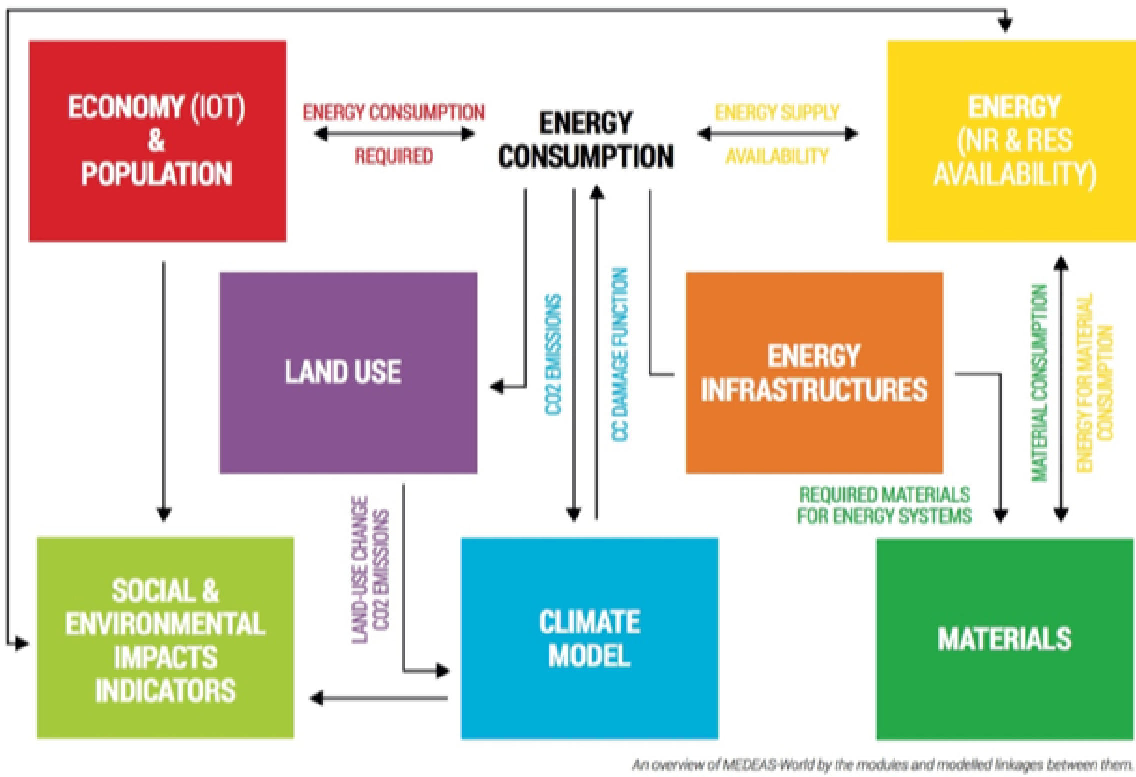


Figure 3.1: The MEDEAS IAM modules

Quite evidently, in this work, the MEDEAS-EU version will be considered, as this is the chosen setting for Dispa-SET, and with an obvious focus on the energy module.

3.4 Energy return on investment

When having energy consideration in the long term, the energy return on investment (EROI) becomes a key indicator.

It is defined as the ratio of the exploitable energy obtained from some energy resource to the amount of exploitable energy used to get that energy resource [19].

The EROI is of great importance 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 is not to be confused with the net energy gain, that is the difference between the exploitable energy obtained and the exploitable energy invested. The net energy is the amount of energy that has been made exploitable, thus now availible for public consumption. Obviously, its value should be larger than zero for the system 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 provides several EROI values relating to different approaches in this matter [2].

– 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)" [2].

– 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" [2].

In this context, we will 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.

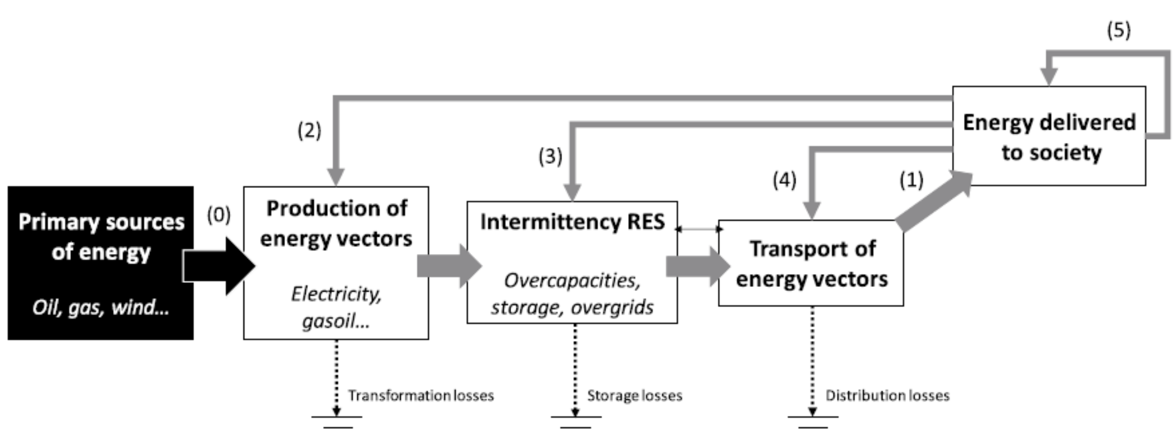


Figure 3.2: Representation of society's principal energy flows[1]

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

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

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

Furthermore, MEDEAS [1]:

– 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

The impacts of the variability of electricity production technologies are tackled in the MEDEAS framework. Indeed, not as extensively as they are in Dispa-SET simulation, what is the whole point of the present work.

3.5.1 Limitation

First, it is recalled that the time step used by MEDEAS is quite significant. Is 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 obviously 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 first storage technology in use is the pumped hydro-storage (PHS). One will notice that it contrasts with the Dispa-SET setting where batteries are used as the main storage technology to act on, mostly by creating new plants. As already discussed before, that is to account for the fact that 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 on Figure 3.3 [12].

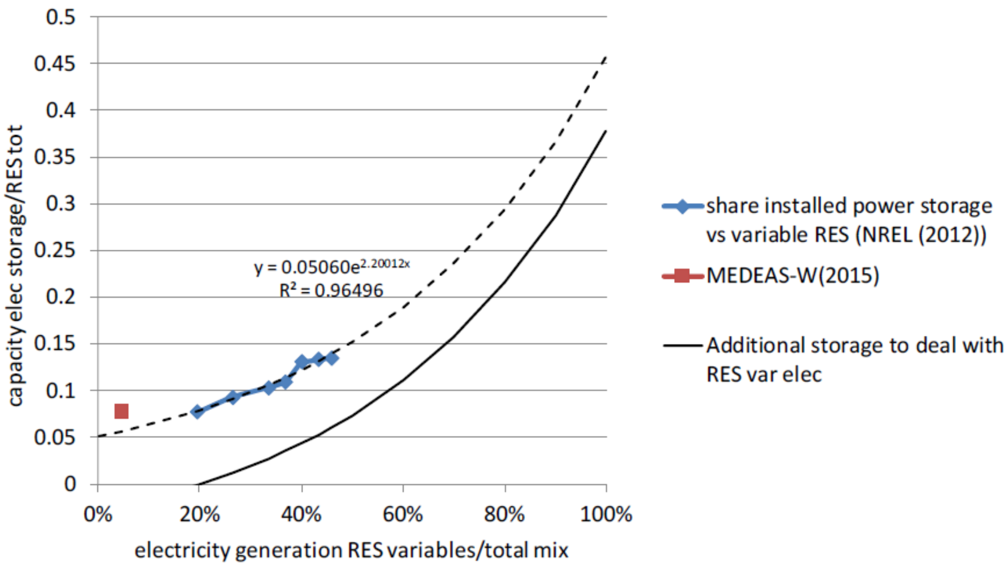


Figure 3.3: Energy storage capacity required as a function of the VRES share according to [12].

3.5.4 Dispatchable RES pants

Here will be describe the evolution of the capacity factor (CF) as the RES share among the electricitym mix changes. 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.

The other meaningfull metric in this context is the overcapacity, that is, the ratio of the energy that could have been produced, over the energy actually produced, depicted on Equation 3.7. *Needs verification (I derived this from Carla, CF=1/(1+overcap))*

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

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

An estimate of the overcapacity of dipachable RES is provided in [12], the resulting capacity factor evolution is depicted as a function of the VRES share on Figure 3.4. It can be observed that capacity factor decreases quadratically in the VRES share in the electricity mix.

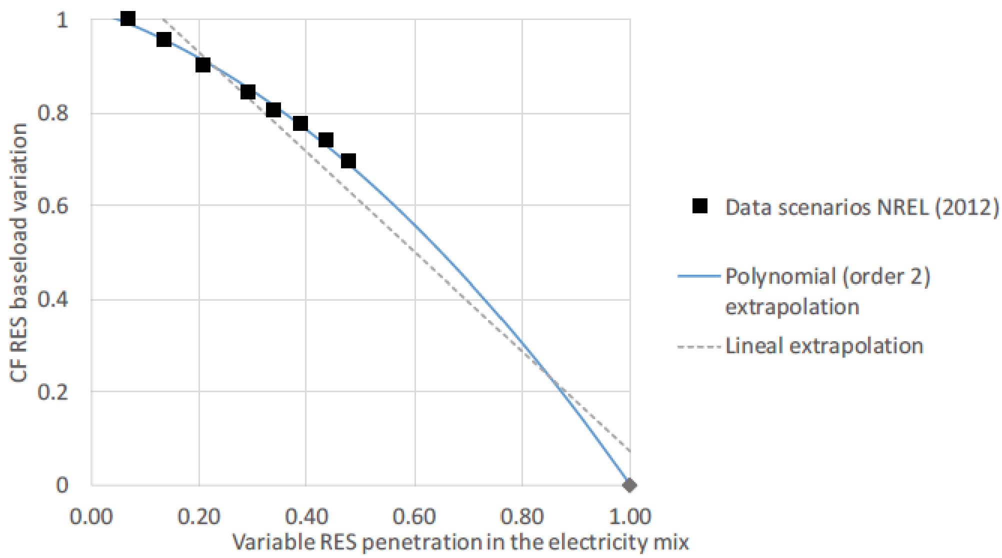


Figure 3.4: Capacity factor of RES evolution depending on the VRES share [12]

3.5.5 VRES plants

MEDEAS based its estimates of the VRES induced overcapacities on [4]. These two main impacts of the VRES share are taken into account:

– The exponential growth of VRES overcapacities and

– The decrease of the VRES capacity factor.

The two estimate functions used in MEDEAS [1] from [4] are presented on Figure 3.5.

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

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

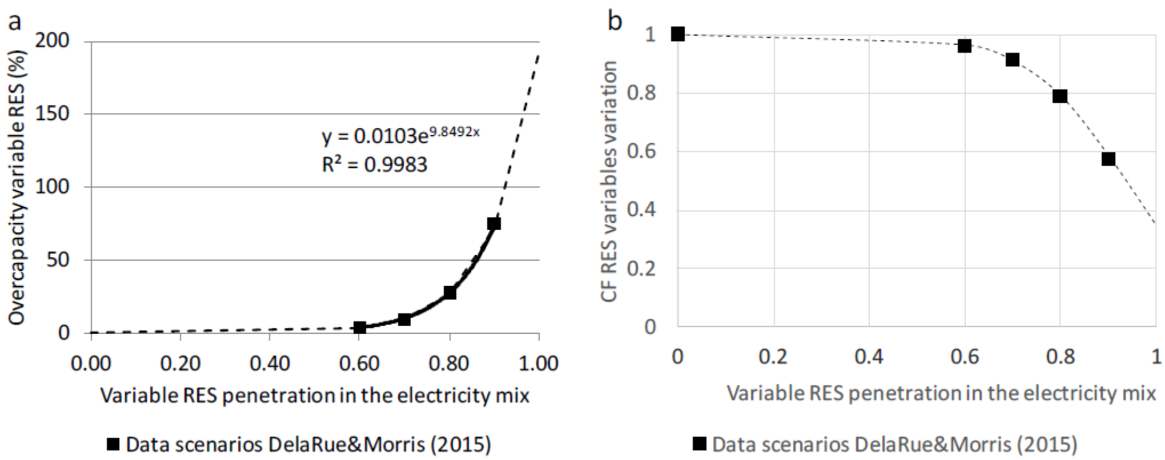


Figure 3.5: (a) Overcapacity estimate and (b) CF reduction of VRES plants on [1]

4 Data generation

4.1 Overview

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

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

This dataset will afterwards be used to train the surrogate model on [22].

4.2 Preparatory work

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

4.2.1 Unit groupings

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

IRENA [16] describes flexible units as "units that can ramp up and down quickly, have a low minimum operating level and fast start-up and shutdown times", what criterion will be used to separate regular units into the slow and flexible units. This criterion is presented in Table 4.1.

<table>
<tr><td>Units</td><td>Fuel</td><td>Condition</td></tr>
<tr><td>$Flex_{units}$</td><td rowspan="2">GAS, HRD, OIL, BIO, LIG, PEA, NUC, GEO</td><td>$PartLoadMin < 0.5$ and $TimeUpMin < 5$ and $RampUpRate > 0.01$</td></tr>
<tr><td>$Slow_{units}$</td><td>$PartLoadMin \geq 0.5$ or $TimeUpMin \geq 5$ or $RampUpRate \leq 0.01$</td></tr>
</table>

Table 4.1: Flexible and slow units classification criterion

Refer to Tables 2.2 and 2.3 for their names.

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

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

– $Storage_{units}$ with (OTH, BATS)

– PV_{units} with (SUN, PHOT)

– $Wind_{units}$ with either (WIN, WTON) or (WIN, WTOF), the latter not being considered in this work

4.2.2 Parameters estimates

The availability factors of PHOT and WTON are also required, as well as the peak load. These values are computed from the reference simulation (2019), and are shown in Table 4.2.

needs check

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

Table 4.2: Values of availability factors and peak load

4.3 Design space

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 will sample them. In our case, this space will be the product of 6 ranges, that is a 6 dimensional hypercube.

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

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

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

For these reasons, an hypercube will be used.

4.3.2 Variables

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

The notation $PowerCap_x$ refers to the maximum power output of all the units in x. See Table 4.2 for the values of the AF and peak load value.

1. **CapacityRatio** [·]

   Ratio of the maximum production over the maximum demand.

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

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.

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

5. **SharePV** [·]

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

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

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 Reference values and ranges

Given the 2019 input data, a reference simulation is run and one obtains the values presented in table 4.3.

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

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

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

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 acheive such sampling are [11] illustrated in the following.

1. The "naïve" sampling: take samples at regular intervals on the design space. Note that one may not choose the same intervals for different dimensions. It is depicted on Figure 4.1a.

2. The Monte-Carlo sapling: pick samples at random all over the design space. It is depicted on Figure 4.1b.

3. The Latin-hypercube sampling [20], maximizing a criterion that is either [5]:

   (a) centering samples in sampling intervals

   (b) maximizing the minimum distance between two samples

   (c) maximizing the minimum distance between two samples, but place sample in a random location in its interval

   (d) minimizing the maximum correlation between two samples

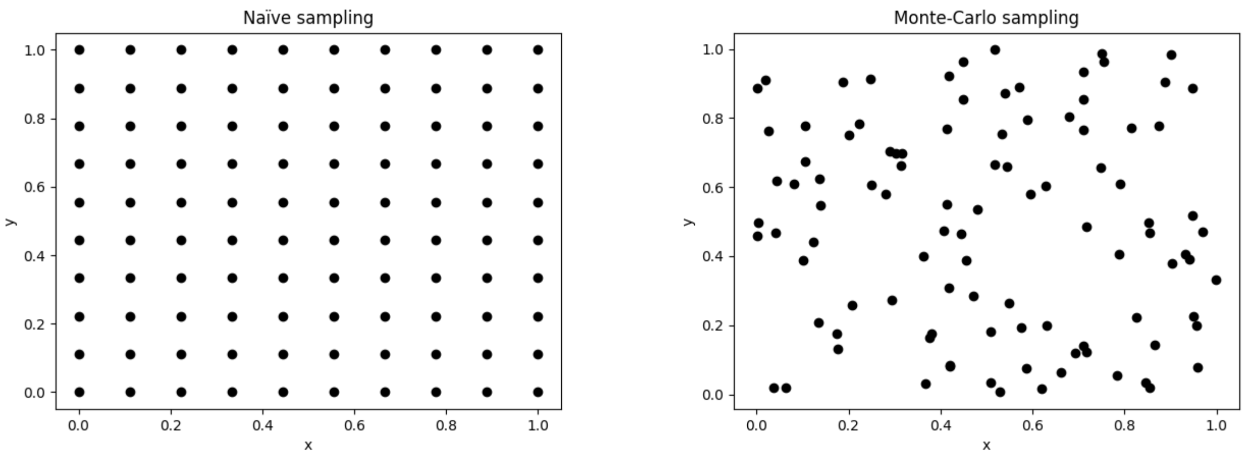
   These are depicted on Figures 4.2a, 4.2b, 4.2c and 4.2d.

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).

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



(a) Naïve strategy (b) Monte-Carlo strategy

Figure 4.1: Basic sampling strategies

4.5.1 Adjusting function

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

These adjusting functions have been specifically developped for this topic by Vidal in her master thesis [18].

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

  To do so, it first computes the target capacity, by multiplying the total capacity by the desired $Share_{flex}$. It then add or subtracts the missing or exceeding flexible unit power capacity to each zone, weighting by their total capacity, following Equation 4.9.

$$capacity_{z,new} = capacityz, old + \frac{capacity_{z,new}}{\sum_x capacity_{x,old}}(target - actual) \tag{4.9}$$

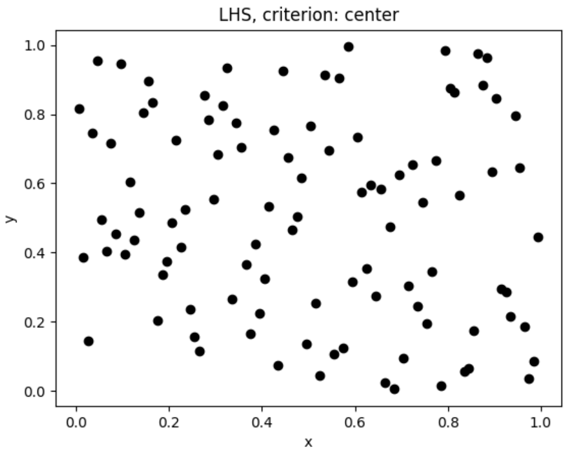
– `adjust_capacity` applies a linear scaling to the power output of some given set of units, in particular, it will be called multiples times to adjust the storage, PV and wind capacities.

  Scaling factors applied are summarized in Table 4.4.

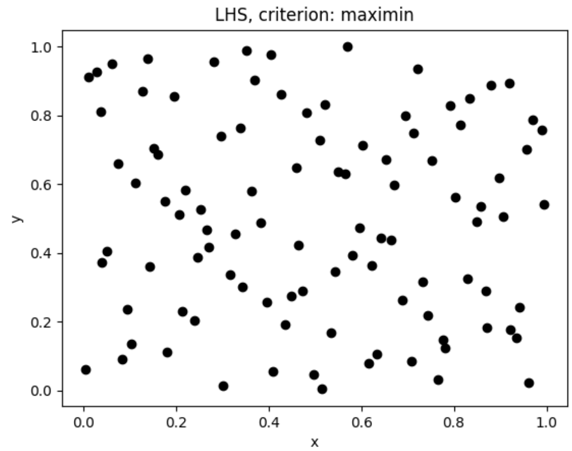
Units	Scaling factor
$Storage_{units}$	$Share_{storage}$
$Wind_{units}$	$\frac{CapacityRatio \cdot Share_{wind}}{AF_{wton}}$
PV_{units}	$\frac{CapacityRatio \cdot Share_{PV}}{AF_{PV}}$

Table 4.4: Scaling factors applied to different units

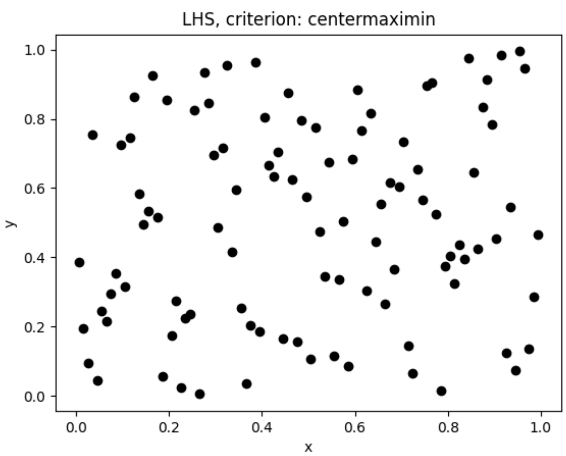
– `adjust_rntc` applies a linear scaling to each zonal NTC time series.



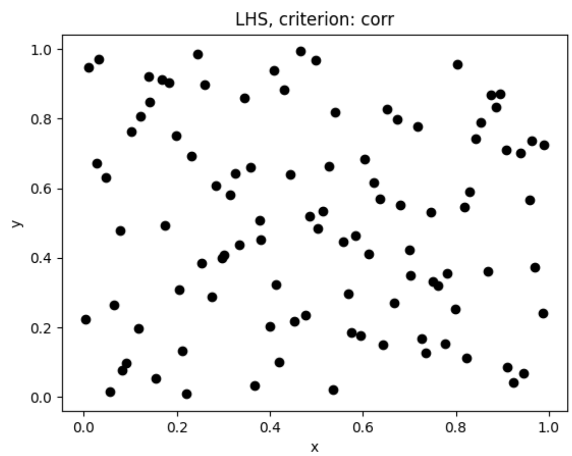
(a) LHS with center criterion



(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

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

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

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

4.6 Implementation

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

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

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

Table 4.5: Values extracted from each simulations

4.6.1 Steps

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

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

4.6.2 Scripts and code

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

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

(b) GAMS is called on the simulation directory.

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

(d) The simulation directory is cleaned.

4.6.3 Technical aspects

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

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

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

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

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

4.6.4 Dataset fields

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

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

Table 4.6: Dataset fields

5 The surrogate model

5.1 Overview

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

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

5.2 Machine learning methods

In the following, some machine learning algorithms for regression problems [6] are assessed.

5.2.1 K nearest neighbors

In this setting, the K-nearest neighbors (k-NN) methods would be one of the easiest to implement[2]. Indeed, these simply require a single parameter value k and output the average value of the output features of the k nearest neighbors, computing a distance on the input features.

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

5.2.2 Decision trees

Decision trees, and extremely randomized trees [14] another category of easy to implement[3] machine learning methods, as they also come with a limited, fixed number design parameters.

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

5.2.3 Kernel-based methods

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

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

5.2.4 Artificial neural network

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

[2]Using for example scikit-learn[15] module in python

[3]Again, using scikit-learn

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

In most recent applications of these ANNs, a lot of different strategies are used to process the data efficiently. For example, convolutional layers convey a lot of meaning in the context of image processing, or transformers are well suited to process sequences[9].

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

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

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

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

In the end, neural networks are opted for. In addition to the abovementioned advantages, neural networks will also:

- Be more 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 well.

5.3 Overfitting and underfitting

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

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

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

5.3.1 Bias

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

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

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

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

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

5.4 Comparison with MEDEAS state of the art

Well, requires the surrogate model

TODO

6 Integration

6.1 Overview

This section describes the process of integrating the trained surrogate model into the MEDEAS model.

As the MEDEAS model is built with the Vensim software, this task is therefore splitted in 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" [17].

Its most common application areas include [21]:

- 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.

Vensim provides 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. *Split here into Vensim overview and Vensim models ?*

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 example of a Vensim model is depicted on Figure 6.1.

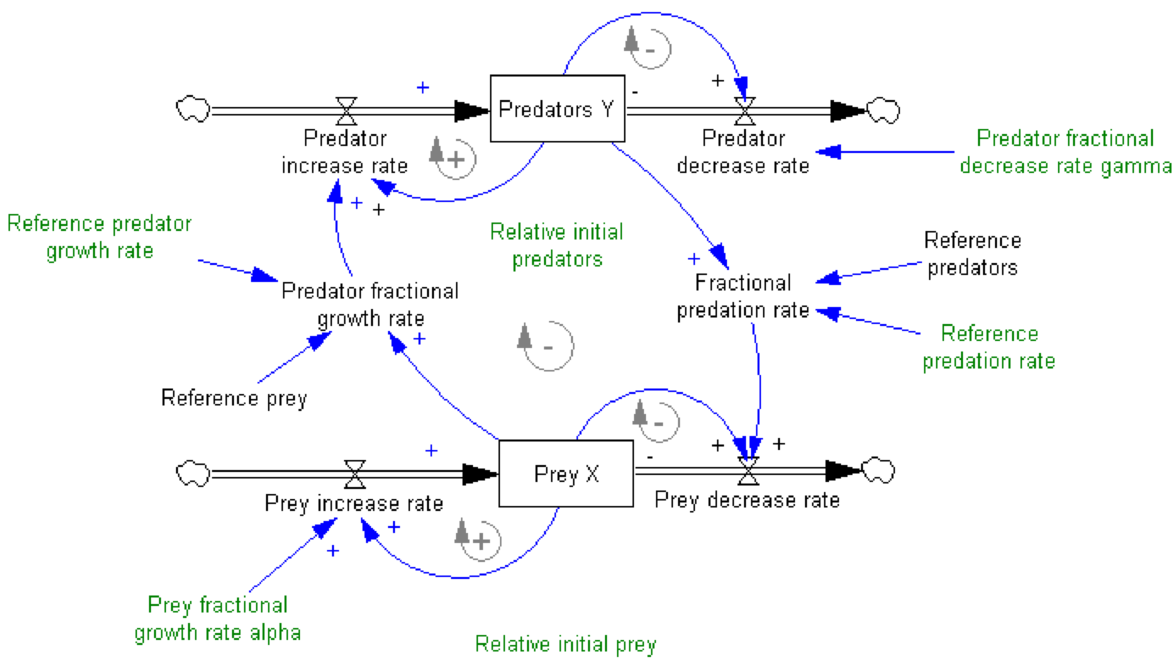


Figure 6.1: A model in Vensim: Lotka-Volterra predator-prey model.

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 any arbitrary function 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 typically compiled from the C or C++ programming language. In this work, C++ has been chosen for its easiest way to load and call the models that will be created in Tensorflow.

To be able to integrate the user-defined functions in its simulation environment, the user's library is expected to provide some functions, that are part of some interface that the Vensim

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

software knows and can communicate with. This interface comprises of a set of functions that is described on Table 6.1.

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

It is thus needed to call our model, presented to as a Tensorflow model, from the C or C++ language.

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

– the said model, saved in a directory from the Tensorflow python API.

– the Tensorflow library, that 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 is an "in-between" layer from the C++ side and the Tensorflow model side.

This tool is not available in C, that's why the library has been written in C++.

The main purpose of Cppflow is precisely to run Tensorflow models from C++, and to acheive this it provides easy to use functions to load such a model, and to call it with some given intput data.

To carry out all these operation in a library, it is thus required to link properly the library code to the two tools it uses, that are the Tensorflow DLL and Cppflow. Hence, using the GNU `make` facility to accomodate the compilation, a few `Makefile`s were written.

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

6.2.4 A small detour...

At some time in the library developpement process, it was successfully managed to produce a standalone program loading the Tensorflow model, and calling it on some inputs, but this was not the case for the library, that was able to compile and link, but not to find the DLL, once called from Vensim.

It has been thought it may not be possible, so a workaround was implemented.

The idea was thus to make the library communicate with another program, that would be launched as a separate process. The library would thus start the other program as an external process, that will be waiting for computation queries, on simulation setup, and end it on simulation shutdown. Then, when the model is called from the library, it sends the input to the worker, the worker calls the model, and sends back the output to the library, that can return.

As previously mentioned, DLL are specific to Windows, so the process management and inter-process commication had to be done in that context.

6.2.5 And a fix

But during some more testing, and adapting of `Makefile`s, the mistake was discovered: the compilation command used to produce the library missed the indication of the path to the DLL.

With that fixed, the library can directly use the Cppflow utilities, what greatly simplifies its architecture and code.

6.3 Variable linking

In its inner workings, the MEDEAS model does not directly uses all the variables that appear in our surrogate models. What arises the need for some links to be drawn between the two.

Hopefully, the desired values are often closely related to the other variables that are already present in the MEDEAS energy module, so these connection are expected to be as simple as linear rescalings or combinations, perhaps some more for the rNTC input.

6.3.1 Linking the input variables

The input variables, that are summarized on Table 4.3 ...

6.3.2 Linking the output variables

References

[1] 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.

[2] 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.

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

[4] 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.

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

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

[7] 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.

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

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

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

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

[12] NREL. *Renewable Electricity Futures Study (Entire Report).* Golden, CO, USA: National Renewable Energy Laboratory, 2012.

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

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

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

[16] *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.

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

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

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

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

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

[22] 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.