Machine Learning Informed Optimization Applied to Pumped Hydro Energy Storage

Pietro Favaro

Main academic Supervisor: Ph.D. Jean-François TOUBEAU, University of Mons

Academic co-supervisor: Professor François VALLÉE, University of Mons

A Master Thesis submitted for the Erasmus Mundus Joint Master Degree on Smart Cities and Communities (SMACCs)

June 2022

University of Mons, Heriot Watt University, International Hellenic University, University of the Basque Country

Acknowledgements

Firstly, I want to express my gratitude towards the European Union and the Erasmus Mundus program for funding this Master in Smart Cities and Communities (SMACCs). I have been granted a scholarship which has allowed me to take part in the life-changing experience this master is. I sincerely hope the master will endure through time, continuing to offer high-quality education to well-deserving students brought together from all over the world by a desire to learn and thrive.

A deep thank you to my supervisor and co-supervisor, respectively Jean-François Toubeau and François Vallée, for their help, support, guidance and, above all, the trust they placed in me from the beginning of my journey into the research world. Working by your sides makes every day inspiring.

The achievement of this master thesis concludes five years spent studying engineering. I want to thank my parents for having always supported me in my studies and provided me, as much as they could, with a healthy life environment. Thank you, Ir. Manon, for having done most of this journey by my sides.

I wish to thank my fellow students for the enriching discussions, discoveries, fun moments, and exchanges of all forms. I have grown by your side and increased my understanding of the world. A special thought for Juan and Irune, for the true friendship we have built. You have made me a better man.

Finally, for making my life in this world full of difficulties easy, I want to thank you, Laura.

Abstract

The increased contribution of uncertain and fluctuating renewable generation, originating mainly from wind and photovoltaic sources, is substantially impacting the operation of power systems. In order to efficiently hedge against these uncertainties, there is a growing need for **flexibility that can be provided by Pumped Hydro Energy Storage (PHES) plants** due to their ability to quickly and cost-effectively respond to mismatches between generation and consumption. PHES systems generally use the pumping and release of water between two reservoirs at different elevations, respectively to store water when the load demand is low (typically at night) and generate electricity when the demand is high.

Accurately modeling the PHES operation is a challenging problem, arising from the fact that the **PHES operation inherently couples electrical and water constraints** through a non-convex and non-concave relationship. Including these characteristics in optimization models is thus associated with high computational requirements [1]–[4].

In this Master Thesis, we propose a **new data-driven paradigm to encode the operating curves of PHES systems**. Practically, we leverage regression-based supervised machine learning (ML) to learn the intricate relationship between electrical and water variables.

Firstly, **multiple linear regression** is studied due to its modeling simplicity but, by imposing a simple linear form, this approach suffers from a limited explanatory power. Then, the modelling power of **neural networks** (NN) is leveraged. Different architectures and activation functions are studied. Both methods achieve better ex-post profits on average than the state-of-the-art in [5]. The foremost advantage is the increased reliability of the two developed data-driven methods. **The NN technique outperforms the linear regression approach in all the tested cases and can have a shorter solving time**.

Keywords: Underground Pumped Hydro Energy Storage System, Optimization, Deep Learning, Neural Network

Pietro Favaro

June 2022

Table of Contents

Nomenclature

Pumped-hydro energy storage	PHES
Unit performance curve	UPC
Neural Network	NN
Underground pumped-hydro energy storage	UPHES
Greenhouse Gas	GHG
Distributed energy resource	DER
Transmission network operator	TSO
Balance responsible parties	BRP
Day-ahead market	DAM
Capital Expenditure	CapEx
Distribution network operator	DSO
Mixed Integer Liner Programming Problem	MILPP
Input Convex Neural Network	ICNN
Linear Programming Problem	LPP
Non-Linear Programming Problem	NLPP
Mixed Integer Linear Programming Problem	MILPP
Frequency Containment Reserves	FCR
Automatic Frequency Restauration Reserves	aFRR
Manual Frequency Restauration Reserves	mFRR
Rectified Linear Unit	ReLU
Gradient Descent	GD
Stochastic Gradient Descent	SGD
Deep Neural Network	DNN

Introduction

Over the last decades, the awareness of the greenhouse gas emissions and the subsequent global warming has grown continuously. The necessity to keep those two phenomena under control has been leading to strong commitments from the international community.

To address these issues and build a sustainable world, it is necessary to phase out electricity generators powered by fossil fuels. To that end, distributed renewable resources have been fostered all around the world. The penetration of those resources, mainly solar and wind, has grown drastically since the beginning of the millennium. This exponentially growing penetration strongly affects the electricity grid because of the intermittency and unpredictability of these energies.

Storage is paramount to counterbalance the drawbacks of those resources and thus, enabling them to reach a higher penetration level. Nowadays, the energy storage system with the highest installed capacity worldwide is the Pumped Hydro Energy Storage System (PHES). PHES are able to provide upward (i.e., increase of the generation or diminution of the consumption) and downward (i.e., diminution of the generation or increase of the consumption) flexibility. However, operating those units in a cost-optimal way remains challenging because of the multidimensionality of the Unit Performance Curve (UPC) and its non-convexity. All those features hinder the performances of model-based optimization leading to substantial losses in unit commitment problems (i.e., problems in which the unit commits to deliver a certain amount of energy at a certain time).

In this work, a hybridization between model-based and model-free approaches is developed to take advantage of both techniques. The starting point of this thesis is a previously developed deterministic optimization model solving the unit commitment problem of an underground pumped storage hydropower system (UPHES). In this work, Neural Networks (NNs) are proposed to model the UPC of the UPHES unit. Not only are NNs expected to model accurately the UPC but also to discard the non-convexity due to the UPC. The trained NN is translated into equations that are embedded into the initial optimization problem of the UPHES commitment, replacing the initial UPC constraint.

This Master Thesis is organized as follows. Chapter 1 presents the context of this work which includes the ongoing energy transition, its main impacts on the grid, an overview of the short-term electricity markets (ancillary and day-ahead market) and the need for high-capacity storage. It concludes with the objectives and motivations of this work. Chapter 2 introduces the current state-of-the-art model for the scheduling of PHES which is an optimization model only (no machine learning). The methods to define the feasible set and the quality of the approximation are discussed. The third chapter starts by presenting briefly the multiple linear regression method followed by an introduction to neural networks (NN) and their data-driven learning procedure. The chapter ends by explaining how to reformulate a NN into a set of mixed-integer linear constraints that can be readily embedded into traditional optimization models. The last chapter, Chapter 4, applies the concepts previously developed to a real-life study-case and presents the results of a wide sensitivity analysis of the parameters. The Master Thesis closes with some key conclusions and perspectives for future works.

1 Context

This first chapter provides the thesis' background. Specifically, the energy transition, the integration of renewable energies and the associated challenges, a brief description of the short-term energy markets and the importance of storage, in particular, the PHES are described. Lastly, the thesis' goal and the supporting objectives are explained.

1.1 Energy Transition

The current episode of global warming has started about 150 years ago with the beginning of the industrialization era. The wide economic development of the 19th century was powered by the steam engine, itself fuelled by coal. The combustion of coal, akin to any other fossil fuel, emits Greenhouse Gases (GHG), mainly CO_2. Those GHG, as per their name, act like a greenhouse surrounding the Earth and thus, increase the Earth's temperature by trapping solar energy. Figure 1 displays the continuous increase (except for rare exceptions) in anthropogenic emissions of CO_2 into the atmosphere since 1750. Figure 2, extracted from the report of the Intergovernmental Panel on Climate Change (IPCC) published in 2021, depicts a clear correlation between the level of CO_2 in the atmosphere and the increase in temperature.

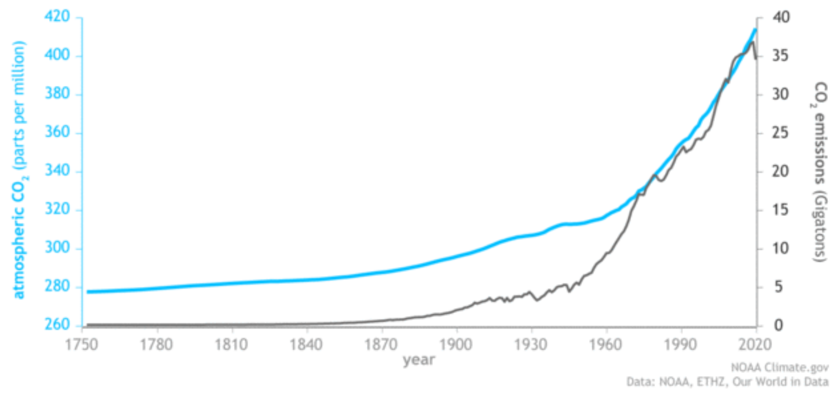


Figure 1: Evolution of (i) the atmospheric concentration of CO_2 from 1750 to 2020 in light blue, (ii) the anthropogenic CO_2 emissions over the same period in dark grey [6].

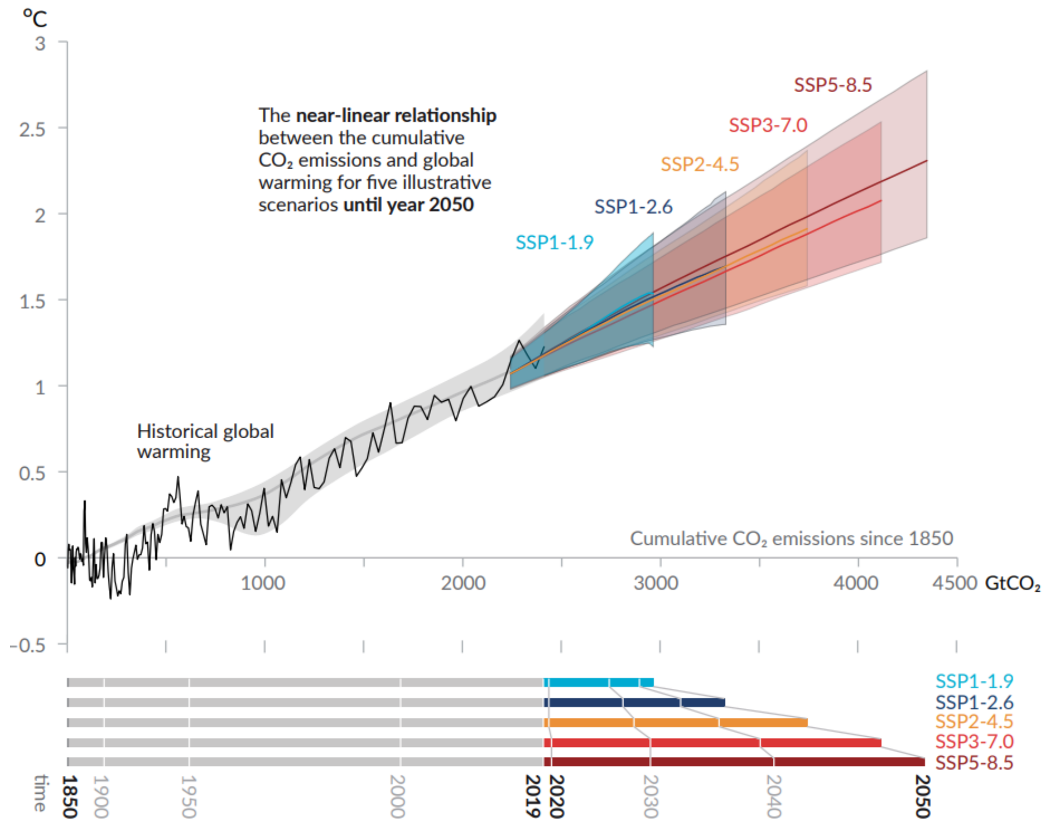


Figure 2: Evolution of the global Earth's surface temperature versus cumulative CO_2 emissions since 1850. The grey line represents the modelled temperature evolution due to the CO_2 discharges while the black line is made of the actual measurements. From 2020 on, various scenarios are depicted corresponding to different rate of CO_2 emissions. [7]

Fortunately, the recent decades have seen a drastic growth in the awareness of both the population and the policy makers with respect to global warming. Firstly, this increasing concern has been translating into the creation of scientific committees and politic gatherings occupying the international agenda. Commitments followed quickly.

In 1972, environmental issues were on the international agenda for the first time during the United Nations Conference on Human Environment held in Stockholm. Twenty years later, the United Nations Framework Convention on Climate Change (UNFCCC) opened for signature in the Rio's United Nations Conference on Environment and Development also known as the Earth's Summit. The goal of the convention is to fight "dangerous human interference with the climate system". In 1997, the Kyoto Protocol, was the first binding agreement obligating developed countries to lower their GHG emissions [8]. The 2015 Paris Agreement, designed at the 21st Convention of the Parties (COP), legally binds the signatory parties to fight climate change in order to limit the global warming well under 2°C, ideally below 1.5°C compared to preindustrial era by reducing their GHG emissions. According to the IPCC's latest report [7], the world has a carbon budget left of 900 $GtCO_2$ and 300 $GtCO_2$ for maintaining the temperature increase below 2°C and 1.5°C, respectively. At the current GHG emissions rate, this corresponds to 24 years (i.e., 2046) and 8 years (i.e., 2030), respectively, before all emissions should stop. Unfortunately, our emissions keep going up as visible in Figure 1. In parallel, the European Union sets its own climate strategy. The latest plan, the 2050 long term strategy, aims at a net-zero GHG emission economy by 2050.

In terms of actions, the reduction of the GHG emissions necessitates to reduce our use of fossil fuels. One of the main emitting sectors is the energy sector which accounted for 77% of the CO_2 equivalent emissions in 2019 in the European Union [9]. This reality has led to many changes in power systems. Massive investments in generation powered by renewable energies have been realized as described in the following section.

1.2 Renewable Energies and Impacts

In the previous section, the necessity of the energy transition and the importance of the energy sector in this transition have been explained. At the power system level, this transition has been implemented by investing in generation units based on renewable energies such as PV panels, wind turbines and the different ways to harvest hydro energy (dam, run-of-the-river scheme, river current turbine…).

The celerity of the ongoing energy transition is well-described by the figures of 2020. Over this year the global renewable generation capacity ramped up by 10.3% to 2 799 GW. Hydro generation represents 43% of the worldwide renewable power capacity making it the leading renewable energy with 1 211 GW. In terms of worldwide renewable energy generation, the hydro share stands even higher at 57.7% corresponding to 4 297 TWh out of 7 444 TWh [10]. Nevertheless, the current increase in renewable energies is mainly powered by solar and wind energy, hydro being historically a more mature technology. Indeed, over the last decade, 21% of the new renewable power capacity came from hydro energy technologies while 35% and 42% came from wind and solar energy technologies, respectively (Figure 3).

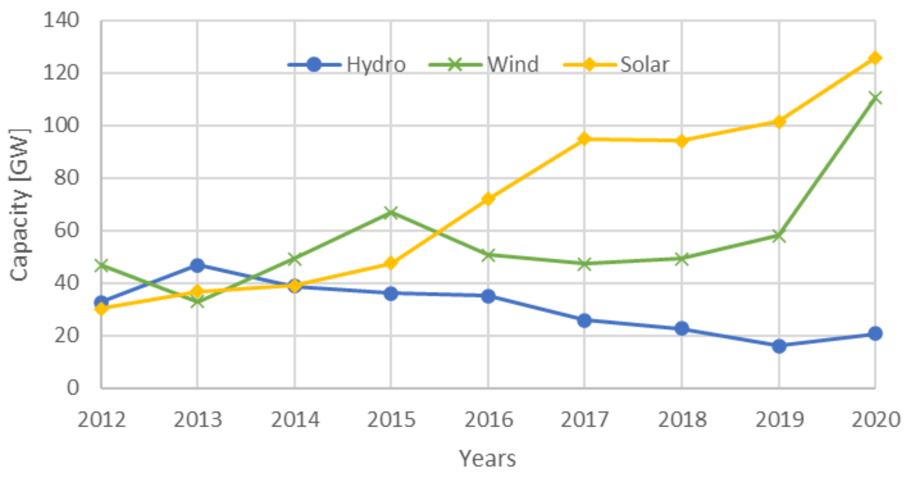


Figure 3: Yearly global additional capacity in GW in solar energy, wind energy and hydro energy (data from [11]).

Those technologies can be installed in large groups such as wind farms or solar panel fields but also on a smaller scale. Therefore, they belong to the group of Distributed Energy Resources (DERs). The growing penetration of DERs causes many issues to the power system. Historically, the grid has been designed to make power flow from the centralized generation units (i.e., large power plants) towards the end-user. Power was flowing from high voltage levels towards low voltage levels. However, with the DER, part of the total generation takes place at the end-user level leading to reversed power flows towards higher voltage levels. This phenomenon can cause overvoltage states, especially at the distribution level.

Furthermore, both solar and wind energies are intermittent and somewhat unpredictable. Hence, they tend to create discrepancies between the energy production and consumption. To ensure the efficient and reliable operation of the grid, both production and consumption must be equal at all times. If a mismatch occurs, it induces a deviation of the system frequency (set at 50 Hz in Europe) which can disturb the grid assets (isolators, capacitor benches, transformers…) and the end-user's equipment (e.g., industrial motors) [12]. This new paradigm in the generation has called for a reform of the electricity market detailed in the next section to gain reactivity and ensure a match between the production and consumption of electricity.

1.3 European Short-Term Electricity Markets

Historically, the electricity sector across Europe was organized as regulated monopolies. One (or several) companies, frequently state-owned (e.g., ancestor of Electrabel in Belgium), were responsible for the generation, transmission, distribution and the end-user supply of electricity. In the 1990's, moved by the neoliberalism current, the European Union decided to unbundle and promote the privatization of the energy sector. Three main objectives were advertised (i) opening the sector to competition through markets, (ii) reaching cost-effective reliability investment, (iii) increase renewable-based generation [13]. Those objectives can be dubbed as mutually exclusive in regard to the impacts of the renewable energies described in the previous section. Therefore, the liberalization of the electricity sector did not reach the expected results. The cost of electricity did not decrease as much as expected even though a cost-effective trade-off between risk and reliability was achieved. If the renewable energy share increased in the energy mix, it is due to additional incentive policies that were developed aside, but not the liberalization itself.

The consequences of the unbundling are regulated monopolies for the transmission and the distribution in most of the member states of EU whereas electricity generators and suppliers evolve in a liberalized market environment. Since the reform, the responsibility to ensure the balance between generation and consumption in real time and this, at all times, falls onto the Transmission Network Operator (TSO) (Elia in Belgium)[1]. However, before the actual delivery (i.e., the real time), the balance responsibility is carried by actors named Balance Responsible Parties (BRP). BRPs are private legal entities which can represent several generators, suppliers, and industrial consumers. They are responsible for composing a balanced portfolio (i.e., a portfolio where the generation is equal to the consumption). A portfolio may be made of own generation/consumption or traded electricity with other BRPs [14].

The major short term electricity market is the Day-Ahead Market (DAM). The European DAM is run by independent market operators named Nominated Electricity Market Operator (e.g.: EPEX SPOT) and ends on the day before the actual delivery at 12 am [15]. At that moment, every BRP must submit a balanced portfolio to the TSO. The DAM is

[1] A transmission network is a network presenting a meshed architecture and voltage levels ranging from 380 kV to 30 kV.

characterized by an hourly granularity, i.e. there is a market for every hour. However, the 24 hours of the following day are cleared at the same time. This means that every BRP knows how much energy it must produce (and thus consume, as both values are equivalent in absolute) over each hour of the following day.

Despite the effort made by every BRP to submit a balanced portfolio the day preceding the delivery, a deviation is very likely in real-time. Indeed, a generator can end up in unexpected, forced outage or, much more probable, the weather forecast turns out to be inaccurate affecting the generation profiles from solar and wind energies. As aforementioned, the actor responsible for the network balance in real-time is the TSO. To offset deviations, the TSO activates reserves it previously contracted through the Balancing Market.

Reserves are of two types: upwards or downwards. The upwards reserve consists of additional generation or reduction of the consumption which can be activated in a given time and amount upon order from the TSO. On the opposite, the downwards reserve consists of additional consumption or reduction of the generation which can be activated in a given time and amount upon order from the TSO. Those reserves can be purchased in the balancing market by the TSO only. Naturally, they must be bought anticipatively so that they can be activated in real-time. Several categories of reserve exist. They distinguish themselves by the time delay allowed to react once the unit has been solicited by the TSO. The billing of the reserves is split into two parts. Firstly, the operator of the reserves earns money for being ready to deliver power upon request from the TSO, even if the latter does not request this flexible power. Secondly, if the plant operator is instructed to supply energy, then (s)he[2] also is paid for this provided energy. Figure 4 summarizes the information about the temporal organization of the electricity markets in Europe.

[2] The masculine or feminine form chosen in the work always refers to male and female persons at the same time.

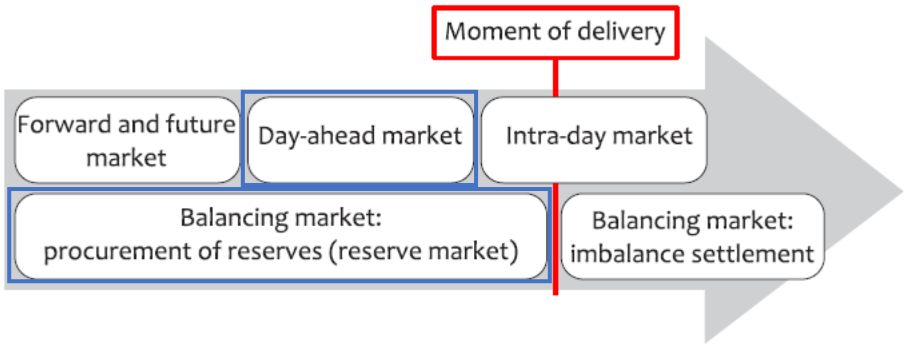

Moment of delivery
Forward and future market
Day-ahead market
Intra-day market
Balancing market: procurement of reserves (reserve market)
Balancing market: imbalance settlement

Figure 4: Temporal organisation of the electricity markets in Europe [14].

1.4 Pumped Hydroelectric Energy Storage

Because of the intermittency, the difficulty of accurately forecasting renewable energies and the obligation to always maintain the balance between electricity generation and consumption, energy storage becomes increasingly important to allow a high penetration of renewable-based DER.

Today, the best technology to store a large amount of energy is PHES. Therefore, this technology is often viewed as the key storage solution to improve renewable energy penetration both on a large (i.e., national grid) and small scale (i.e., disconnected island grid) [16], [17]. In addition, PHES is the unique technology currently commercialized for long-term lasting storage [17]. However, its high CapEx is often criticized [18].

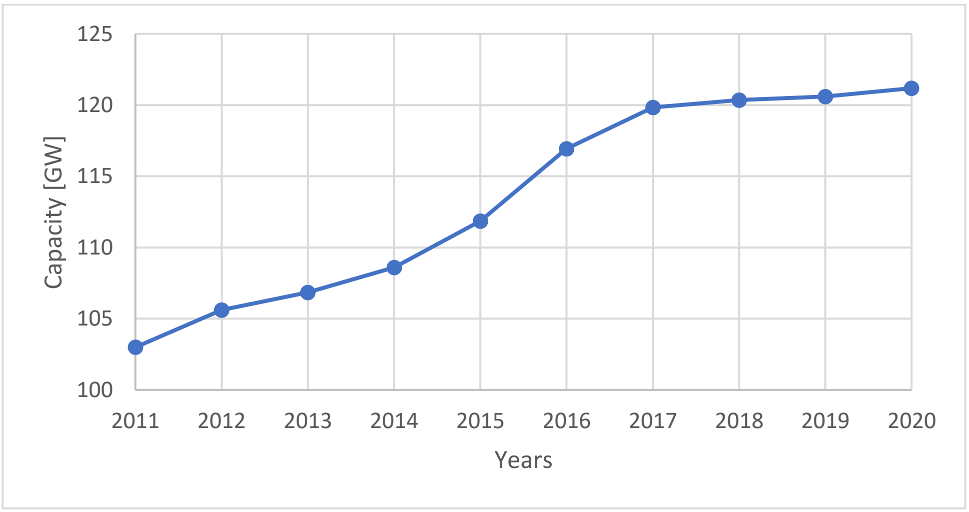


Figure 5: Global installed capacity of pure PHES from 2011 to 2020 (data from [11]).

PHES is a very mature technology which developed first in the Alps during the 1890's. In Belgium, the largest PHES unit is the dam of Coo built in 1972 which features a capacity of 1080 MW [19]. In USA, 95% of the utility-scale storage comes from PHES [17]. Despite its maturity and the complexity to find available appropriate sites, the global installed capacity of PHES has been growing over the last decade reaching slightly over 121 GW at the end of 2020 (Figure 5).

PHES units are able to store energy (even over long duration) by converting electrical energy into gravitational potential energy. In PHES, the body which stores the potential energy is water. Water is available in abundance, cheap and harmless to the environment. The computation of the gravitational potential energy can be expressed as (1a) - (1c).

$$E_{g.p.} = m \cdot g \cdot h \tag{1a}$$

$$= \rho \cdot V \cdot g \cdot h \tag{1b}$$

$$= K \cdot V \cdot h \tag{1c}$$

where $E_{g.p.}$ in [J] is the gravitational potential energy of a body;
m in [kg] is the mass of the considered body;
g in [m/s^2] is the gravity acceleration (~9.81 m/s^2 on Earth);
h in [m] is the height of the body's gravity centre with respect to a reference height arbitrarily chosen, h increasing in the opposite direction as g;
ρ in [kg/m^3] is the mass density of the body;
V in [m^3] is the volume of the body;
$K = \rho \cdot g$ in [kg/(m^{2}s^2)] is a constant.

Following the equations, in order to store energy under the form of gravitational potential energy, a body characterised by a volume V must undergo an increase of height h. For that reason, PHES are composed of two reservoirs differentiated by their height (Figure 6). The volume of water V to be exchanged from one reservoir to the other depends on the amount of energy to store/release since h is fixed by the unit topology and the water level of each reservoir. The reservoirs are connected by a pipe named penstock. Somewhere along this penstock the machinery room is fitted. There, one or several turbines harvest energy from the water falling from the upper reservoir towards the lower reservoir. This energy, harvested under mechanical energy (i.e., rotation of the turbine shaft), is transformed into electricity by an alternator connected to the turbine's shaft. On the opposite, when the unit is used to store electrical energy, water is pumped from the lower reservoir to the upper reservoir. Usually, the turbines equipping the PHES stations are reversible so that they can also operate in pumping mode (e.g., Francis turbine).

The working principle is similar for UPHES. The only difference is that the lower reservoir is located under ground level, typically in an abandoned quarry or mine. It reduces drastically the investment cost of the unit but creates additional difficulties. Firstly, the geometry of the underground reservoir is extremely complex and impossible to determine accurately. Secondly, this reservoir interacts with the surrounding aquifer which creates uncertainties on the water level [20].

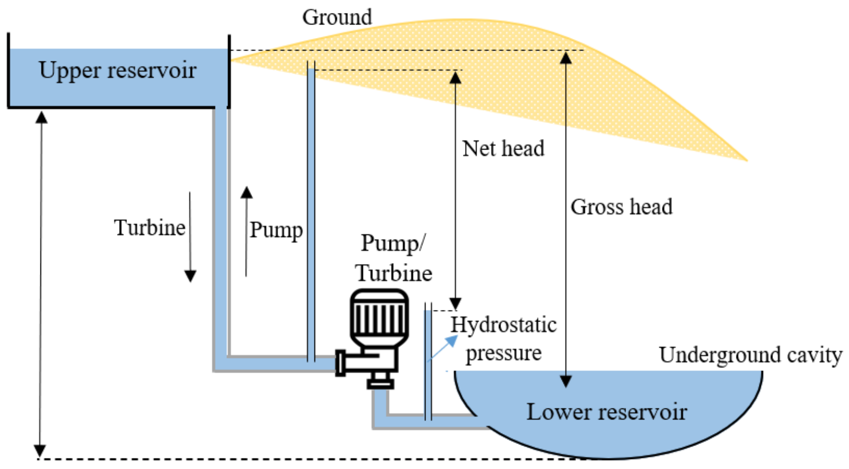


Figure 6: Typical sketch of underground pumped hydro energy storage (UPHES) [5].

The study case of this Master Thesis is a hypothetical UPHES unit on a suitable Belgian site named Maizeret. It is considered that the operator of the unit can take part in the electricity markets in (i) bidding electricity on the day-ahead market, (ii) offering reserves to the TSO. It should be noted that a (U)PHES unit can offer both upwards and downwards reserve in both pump and turbine modes, theoretically. The unit operator can place his bids how he sees fit. Obviously, it is the operator's interest to pump water (i.e., consume electricity from the grid) when the electricity price is low and turbine water (i.e., generate electricity) when the electricity price is high.

1.5 Objectives and Motivations

The importance of the energy sector in the GHG emissions is apparent from the discussion in Section 1.2. This observation calls for changes in the way power systems are operated. More specifically, the electricity generation is now expected to be powered by renewable energies such as solar or wind. The ultimate goal is to reach net-zero GHG emissions to stem global warming. This switch towards renewable energies brings many challenges due to the intermittency and limited predictability of the renewable resources. To alleviate those drawbacks, large storage capacities are needed. The most mature large-scale storage technology is PHES. A variant of the technology, UPHES is currently studied to reduce drastically the investment cost of the technology. A key element to foster (U)PHES is the ability to operate them in a cost-optimal manner.

Conventionally, model-based optimization is used to maximize the economic benefits obtained by operating a unit. However, modelling PHES units, especially UPHES ones, is extremely challenging. Moreover, the Unit Performance Curves (UPCs) of such plants are non-convex, non-concave relations. Consequently, it is not possible to guarantee the finding of the global optimal solution in the actual form of the problem because of the non-convexities of the constraints. Some convexification methods and hypotheses can be applied to find a solution at the expense of the solution accuracy. Furthermore, solving optimization problems with binary constraints can quickly become time-consuming. **Therefore, in this Master Thesis, a novel model-based approach enriched with data-driven constraints is studied to take advantage of both worlds and increase the profits of UPHES units participating in the DAM and the reserve market.** More specifically, the objectives of this work are the following:

1. Updating and analysing the results output by an already-existing deterministic optimization algorithm modelling the operations of a candidate (Maizeret site in Belgium) UPHES plant.
2. Improving the current state-of-the-art model by carrying out some sensitivity analyses on the number of head and power subintervals which defines the quality of the UPC approximation. The influence of the shape chosen to define the UPC bounds is also investigated.
3. Replacing the UPC in the initial optimization problem of objective 1 by a simple multiple linear regression. To carry out this task, a scatterplot of the UPC is available.
4. Replacing the UPC in the initial optimization problem of objective 1 by piecewise linear equations (with the use of binaries) derived from a NN trained to fit the UPC.

The achievement of those objectives will deliver a brand-new hybrid model to operate and schedule UPHES with an expected enhanced accuracy and reduced computing time. On a wider scale, the successful combination of model-based and model-free methods opens the door to significant improvements in a wide variety of fields including power systems.

2 Optimization

This chapter introduces a state-of-the-art model for optimizing the scheduling of a PHES plant. The reader is supposed to be familiarized with the main theoretical concepts related to optimization. If it is not the case, please refer to Introduction to optimization in Appendix A.

2.1 Deterministic Optimisation Model For a Fictitious UPHES Unit In Maizeret

This section explains the deterministic optimization modelling of a fictitious UPHES unit that would be located at the Maizeret site in Belgium and that would take part into both the ancillary market and the DAM one. The model, available in [5], was developed a few years ago but needed to be updated to the new software version. It is the starting point of this Master Thesis.

2.1.1 Nomenclature

Sets and indices

R	Set of the reserve categories, index r.
$R^+ \subseteq R$	Set of upwards reserve categories.
$R^- \subseteq R$	Set of downwards reserve categories.
T	Set of the time steps, index t.
H	Set of UPHES units, index h.
N	Set of subintervals of net head, index n.
M	Set of subintervals of power levels, index m.
I	Set of the modes available (pump (P) or turbine (T)), index i.

The order of a given set (i.e., the number of elements contain in a set) has the same symbol as the set considered.

Decision variables

Φ	Profit of the UPHES operator.
res_r	Total reserve capacity (or availability) allocated in the reserve category r, [MW].
$res_{h,r}^T$, $res_{h,r}^P$	Reserve capacity (or availability) allocated in turbine (T) and pump (P) modes of the unit h in reserve category r, [MW].
$v_{h,t,r}^{res}$	Extra volume of water displaced due to the activation of reserves of the unit h at time step t in reserve category r, [m^3].
e_t^{DA}	Energy exchanged on the DAM at time step t, [MWh].
$z_{h,t}^T$, $z_{h,t}^P$	Binary variables flagging the operation mode (turbine (T) or pump (P)) of unit h at time step t.
$z_{h,t,n}^{(1)}$	Binary variable flagging the subinterval of head in which the variable $h_{h,t}^{net}$ is for the unit h at time step t.
$z_{h,t,m,n}^{(2),T}$, $z_{h,t,m,n}^{(2),P}$	Binary variables flagging the subinterval of power associated to the head subinterval n in which the variables $p_{h,t}^T$, $p_{h,t}^P$ are for the unit h at time step t.
$p_{h,t}^T$, $p_{h,t}^P$	Power output in turbine (T) and pump (P) modes by unit h at time step t, [MW].
$q_{h,t}^T$, $q_{h,t}^P$	Water flow rates in turbine (T) and pump (P) modes of unit h at time step t, [m^3/s].
$v_{h,t}^{up}$, $v_{h,t}^{low}$	Water volume in upper (up) and lower (low) reservoirs of unit h at time step t, [m^3].
$h_{h,t}^{up}$, $h_{h,t}^{low}$	Water head in the upper (up) and lower (low) reservoirs of unit h at time step t, [m].

$h_{h,t}^{loss}$	Head loss of unit h at time step t, [m].
$h_{h,t}^{net}$	Net head of unit h at time step t, [m].
$\overline{h}_n$	Upper limit of the n^{th} head subinterval [m].
$d_{h,t,n}^{PHES}$	Binary variable equal to 1 if the net head of unit h at time step t is in head interval n, and 0 if not.

Parameters

λ_r^{res}	Price (constant over one day) for reserve availability in the reserve category r, [€/MW].
λ_t^{DA}	Price for the electricity on the DAM at time step t, [€/MWh].
C_h^{op}	Operating cost of the unit h, [€/MW].
Δt	Time span of a time step of the optimization problem, [h].
$\Delta P_{h,r}^P$ and $\Delta P_{h,r}^T$	Ramping abilities in turbine (T) and pump (P) modes for unit h in reserve category r, [MW].
$\overline{Q}_h^P, \overline{Q}_h^T$	Maximum water flow rate in turbine (T) and pump (P) modes for unit h, [m^3/s].
$\overline{V}_h^P, \overline{V}_h^T$	Maximum volumes of water in the upper (up) and lower (low) reservoirs of unit h, [m^3].
V_h^{target}	Target water volume in the upper reservoir at the end of the optimization time horizon for unit h, [m^3].
$\underline{F}_{h,n}^T, \overline{F}_{h,n}^T$	Stepwise approximation of the lower and upper power bounds of the safe operating zone in turbine mode (T) for the neat head interval n, [MW].

$\underline{f}^{T}_{h,n}, \overline{f}^{T}_{h,n}$	Piecewise linear approximation of the lower and upper power bounds of the safe operating zone in turbine mode (T) for the neat head interval n, [MW].
$\overline{A}^{T}_{h,n}, \overline{B}^{T}_{h,n}$	Slope and constant term of the upper bound of the safe operating zone in turbine mode (T) for the net head interval n, [MW/m], [MW].

2.1.2 Goal of the model

The model aims at optimizing the participation of an UPHES operator in the reserve market and the DAM market over a 24-hour period. Naturally, the operator wants to maximize its profit. No uncertainty over the parameters or the data fed into the model is considered. They are all supposed to be known precisely, in particular, the hourly DAM prices for the next 24 hours, leading therefore to a deterministic model.

When determining the operating schedule of any (U)PHES, the constraints of the system must be considered. As any generator, the capacity (i.e., the maximum power that can be delivered) is bounded. The same applies to the ramping abilities which define the maximum possible increase or decrease in power per unit of time. In addition, a three-dimensional relationship binds the water flow, the net head (i.e., difference of height between the level of water in the upper reservoir and the lower reservoir minus the losses) and the power delivered. The relationship, named Unit Performance Curve (UPC), governs the operating conditions of the hydraulic machine for both the turbine and the pump modes. Figure 7 depicts the UPC for a conventional variable-speed Francis turbine. The reader is invited to click on the figure to access an interactive plot of the curve and realise the complexity of the relationship, which is neither linear or convex, nor concave.

Moreover, projecting the UPC with respect to the water flow lets two forbidden zones appear within which the unit cannot operate to avoid mechanical issues (Figure 8). The boundaries of the UPC are extremely difficult to model as they are non-linear. Their existence can be explained by the physics of the problem. For a given head, increasing the output power implies to increase the water flow rate which can lead to the apparition of cavitation (i.e., vaporization of a liquid due to a sharp drop in pressures). Cavitation can cause severe damage to the equipment and must be avoided [21]. On the contrary, a too small water flow rate and, thus, a too low partial load can trigger undesired erosion and mechanical vibrations [22].

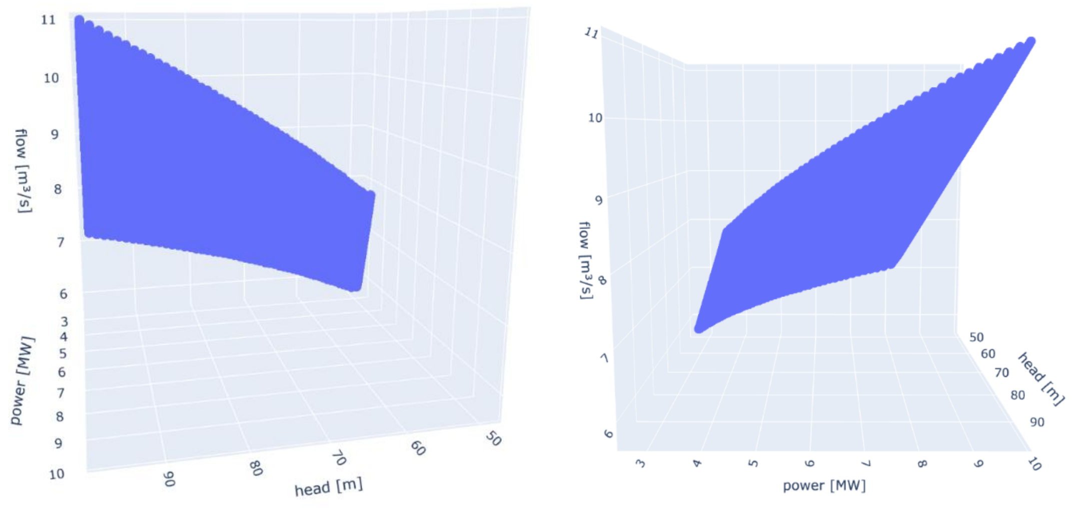


Figure 7: Snapshot of the three-dimensional turbine unit performance curve (UPC) (click on the figures and download for an interactive view).

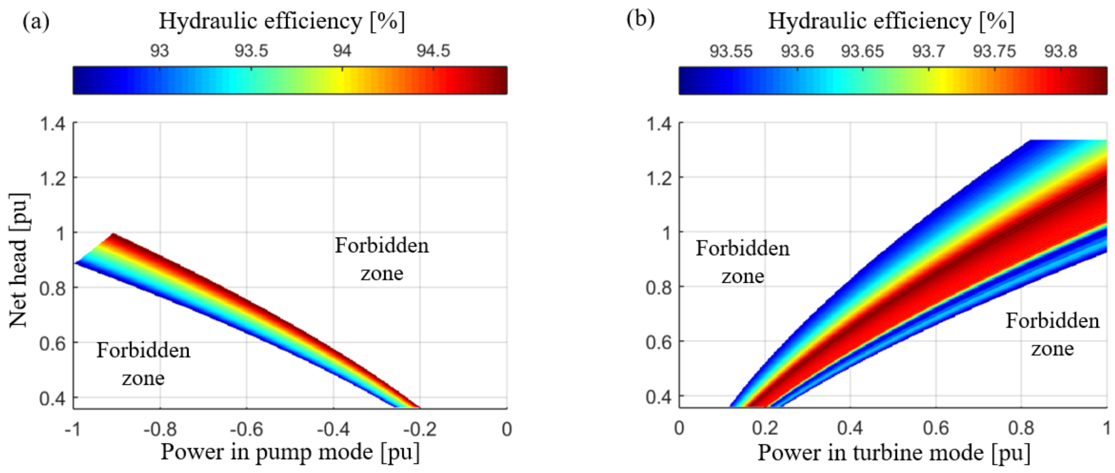


Figure 8: Unit performance curve (UPC) of a Francis turbine in pump mode (a) and turbine mode (b) projected w.r.t. the water volume flow [14].

2.1.3 Methodology

Hypotheses and data

To formulate the scheduling problem of an UPHES operator which aims to maximize its profits in the DAM and the balancing market, several hypotheses have been identified.

- The operator is a price-taker in both markets (i.e., the bid of the UPHES operator has no influence on the market clearing price).
- The prices on both markets are supposed to be fully predictable. In other words, the operator knows the exact value of electricity on the day before delivery.

- The daily clearing of the reserve market which, in Europe, occurs shortly before the clearing of the DAM during independent auctions is here formulated as a single decision.
- The operator is supposed to be always in balance, meaning that offers on either market must actually be realised.
- If the UPHES operator is solicited by the TSO to supply/consume energy due to its participation in the reserve market, the money earned for supplying/consuming this energy offsets exactly the operating cost for generating it. Therefore, those operations are money-neutral for the UPHES operator and thus, transparent in the objective function (see below).
- The reservoirs have a rectangular geometry. Therefore, the volume of water stored in the reservoir is directly proportional to the head. The factor linking both is the surface of the reservoir.
- A scatter plot of the UPC giving the water flow for 51 different levels of head and 1001 levels of power (51 051 points in total) is available.

Objective function

The UPHES operator participates in the markets by bidding quantities which correspond to charging (pump) power and discharging (turbine) power. The objective function (Equation (2)) is the maximization of the operator's revenues and consists of three terms: (i) the profits earned by selling up or down power capacities in the reserve market, (ii) the money streams associated with the purchases and sells of power on the DAM and (iii) the operating cost of the unit proportional to the power.

$$\max \Phi = \sum_{r \in R} 24 \cdot \lambda_r^{res} \cdot res_r + \sum_{t \in T} \left[\lambda_t^{DA} \cdot e_t^{DA} - \sum_{h \in H} C_h^{op} \cdot (p_{h,t}^T + p_{h,t}^P) \right] \tag{2}$$

Constraints

Energy balance and reserve allocation constraints

Constraint (3) enforces the energy balance between the energy exchanged on the DAM and the one generated by the plant.

$$e_t^{DA} = \Delta t \cdot \sum_{h \in H} p_{h,t}^T - p_{h,t}^P \qquad \forall t \tag{3}$$

The UPHES has some flexibility which can be valued by taking part in the reserve market. Constraints (4) defines res_r which appears in the objective function.

$$res_r = \sum_{h \in H} (res_{h,t,r}^{T} + res_{h,t,r}^{P}) \qquad \forall t, r \qquad (4)$$

In this work, three different categories of reserve are considered. Each of those types can be upward or downward. Upward reserve can be achieved by increasing the power generation or reducing the consumption. Similarly, downward reserve consists of reducing the power generation or increasing the consumption.

1. The Frequency Containment Reserves (FCR) are automatically activated and must response within 30 seconds (fu for upward FCR, fd for downward FCR).
2. The automatic Frequency Restoration Reserves (aFRR) are brought into play after the intervention of the FCR to free it for other possible contingencies and restore the frequency to its value of 50 Hz. The reserve must be able to be fully activated within 7.5 minutes (au for upward aFRR, ad for downward aFRR).
3. Manual Frequency Restoration Reserves (mFRR) is needed if the problem endures. These reserves stay on as long as necessary and are fully activated in 15 minutes (mu for upward mFRR, md for downward mFRR).

For each reserve category, it must be ensured that the unit has the ability to ramp its power within the dictated time interval.

$$res_{h,t,r}^{i} \le z_{h,t}^{i} \cdot \Delta R_{h,r}^{i} \qquad \forall h, t, i \in \{H, T, I\}, r \in \{fu, fd\} \qquad (5)$$

$$res_{h,t,fu}^{i} + res_{h,t,au}^{i} \le z_{h,t}^{i} \cdot \Delta R_{h,au}^{i} \qquad \forall h, t, i \in \{H, T, I\} \qquad (6a)$$

$$res_{h,t,fd}^{i} + res_{h,t,ad}^{i} \le z_{h,t}^{i} \cdot \Delta R_{h,ad}^{i} \qquad \forall h, t, i \in \{H, T, I\} \qquad (6b)$$

$$\sum_{r \in R^{+}} res_{h,t,r}^{i} \le z_{h,t}^{i} \cdot \Delta R_{h,mu}^{i} \qquad \forall h, t, i \in \{H, T, I\} \qquad (7a)$$

$$\sum_{r \in R^{-}} res_{h,t,r}^{i} \le z_{h,t}^{i} \cdot \Delta R_{h,md}^{i} \qquad \forall h, t, i \in \{H, T, I\} \qquad (7b)$$

Constraint (5) ensures that the FCR allocated by the UPHES owner is not greater than the power variation the plant can reach within 30 seconds. Due to Equations (6a)-(6b), the remaining ramping capacity within a time span of 7.5 minutes can be allocated as aFRR. Similarly, Equations (7a)-(7b) allocate the remaining ramping capacity within a time span of 15 minutes to the mFRR.

Technical constraints

The UPHES operator faces a high number of technical constraints. Equation (8) states that there can be a pumped flow of water only if the unit is in pump mode and similarly for the turbine mode.

$$q_{h,t}^{i} \leq z_{h,t}^{i} \cdot \overline{Q}_{h}^{i} \qquad \forall h, t, i \in \{H, T, I\} \tag{8}$$

The time sequentiality in the decisions is obtained by updating the volume of water in the upper basin (9a) and the lower basin (9b) between each time step.

$$v_{h,t}^{up} = v_{h,t-1}^{up} + (q_{h,t}^{P} - q_{h,t}^{T}) \cdot \Delta t \qquad \forall h, t \in \{H, T\} \tag{9a}$$

$$v_{h,t}^{low} = v_{h,t-1}^{low} + (q_{h,t}^{T} - q_{h,t}^{P}) \cdot \Delta t \qquad \forall h, t \in \{H, T\} \tag{9b}$$

Consequently, the volume of the reservoirs must be constrained. Obviously, the volume of each basin must be within a range bounded by a minimum and a maximum amount of water. Moreover, if the UPHES owner participates in the reserve market, the allowed range of volume is reduced because there must always be enough water stored to provide the power if the TSO requests it. In the lower reservoir, water can be pumped in case of activation of the downward reserve while water can be turbined when upward reserve is activated. A similar reasoning, but reversed, applies to the upper reservoir. Those constraints considering the impact of the reserve are enforced by Equations (10a)-(10b).

Equation (11) computes the additional volumes linked to the contracted reserves considering the efficiency of the unit, the gravitational acceleration ($g = 9.81$ m/s) and the density of the water ($\rho = 1000$ kg/m^3).

$$\underline{V}_{h}^{up} + \sum_{t'=1}^{t} \sum_{r \in R^{+}} v_{h,t',r}^{res} \leq v_{h,t}^{up} \leq \overline{V}_{h}^{up} - \sum_{t'=1}^{t} \sum_{r \in R^{-}} v_{h,t',r}^{res} \qquad \forall h, t \in \{H, T\} \tag{10a}$$

$$\underline{V}_{h}^{low} + \sum_{t'=1}^{t} \sum_{r \in R^{-}} v_{h,t',r}^{res} \leq v_{h,t}^{low} \leq \overline{V}_{h}^{low} - \sum_{t'=1}^{t} \sum_{r \in R^{+}} v_{h,t',r}^{res} \qquad \forall h, t \in \{H, T\} \tag{10b}$$

$$v_{h,t',r}^{res} = \frac{3600 \cdot 10^{6} \cdot (res_{h,t',r}^{T} + res_{h,t',r}^{P})}{\eta_{h,t'} \cdot \rho \cdot g \cdot h_{h,t'}^{net}} \tag{11}$$

To ensure the model considers events beyond its optimisation horizon and has some flexibility to act upon them, the volume of water in the upper reservoir, which is an image of the energy stored, can be imposed to be greater than a given value (Equation (12)).

$$v_{h,t'=T}^{up} \geq V_{h}^{target} \qquad \forall h \in \{H\} \tag{12}$$

As highlighted previously, the UPC of the hydraulic machines which governs the operation of the unit is a non-linear relation binding the net head, the flow rate and the power. Therefore, the net head of water must be computed. The water volume can be converted into head by using the relationship defining the surface of the reservoir (Equations (13a)-(13b)).

$$h_{h,t}^{up} = f_h^{up}(v_{h,t}^{up}) \qquad \forall h,t \in \{H,T\} \tag{13a}$$

$$h_{h,t}^{low} = f_h^{low}(v_{h,t}^{low}) \qquad \forall h,t \in \{H,T\} \tag{13b}$$

Due to friction losses in the penstock (Equation (14)), the net head of water is always smaller compared to the difference of height between the water levels (Equation (15)). However, for ease, the frictions loses are not considered.

$$h_{h,t}^{loss} = c_h^{loss} \cdot (q_{h,t}^T + q_{h,t}^P)^2 \qquad \forall h,t \in \{H,T\} \tag{14}$$

$$h_{h,t}^{net} = h_{h,t}^{up} - h_{h,t}^{low} - h_{h,t}^{loss} \qquad \forall h,t \in \{H,T\} \tag{15}$$

UPCs are nonlinear. To keep the problem piecewise linear, the UPCs are approximated by a piecewise linear function. The approximation of the UPCs is represented by Equation (16).

$$p_{h,t}^i = f_h^{UPC,i}(q_{h,t}^i, h_{h,t}^{net}) \qquad \forall h,t,i \in \{H,T,I\} \tag{16}$$

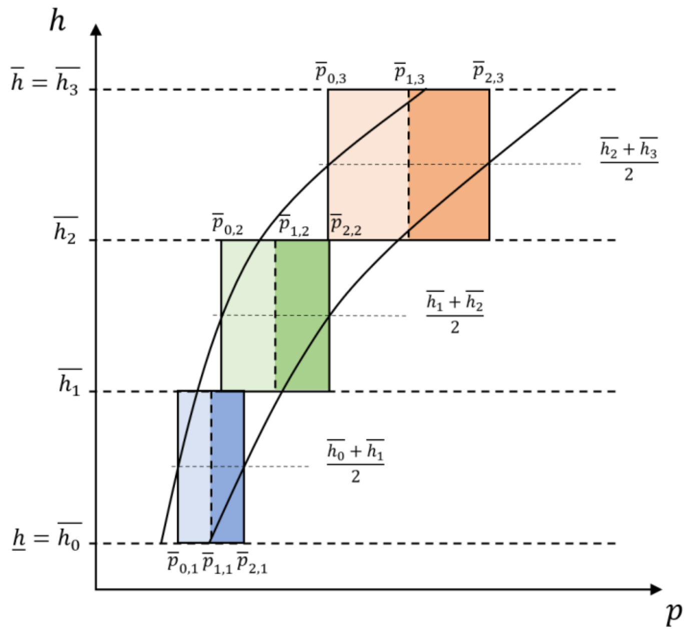


Figure 9: Sketch of the piecewise approximation method based on three head and two power subintervals. Firstly, the feasible head range is divided into three equals ranges. On each of these subintervals, a range of feasible power is obtained by referring to the original UPC for a level of head in the middle of the subinterval. Afterwards, the obtained power range is divided into two equivalent subintervals. On each subinterval of head and power, a plane whose slope depend only on p fits the UPC (each colour represents a different plane).

Let us have a deeper look into the piece-wise reformulation of the UPC which follows the work presented in [23] that gives rise to a set of mixed-integer constraints. The method is illustrated by Figure 10 for three head and two power subintervals.

Firstly, the neat head space $[\underline{h}, \overline{h}]$ is divided into subintervals $n \in N$ whose the $N + 1$ breakpoints fulfil Equation (17a). Each of the newly created head subinterval is associated with a binary variable $z_{h,t,n}^{(1)}$ which goes to one to allow the net head $h_{h,t}^{net}$ to take its value within this subinterval. $h_{h,t}^{net}$ takes its value within that subinterval through an intermediary variable $h_{h,t,n}$. This is ensured by constraints (17b)-(17e).

$$\underline{h} = \overline{h_0} < \overline{h_1} < \cdots < \overline{h_N} = \overline{h} \tag{17a}$$

$$z_{h,t,n}^{(1)} \in \{0, 1\} \qquad n = 1, 2, \dots, N \tag{17b}$$

$$\sum_{n=1}^{N} z_{h,t,n}^{(1)} = 1 \qquad \forall h, t \in \{H, T\} \tag{17c}$$

$$\overline{h_{n-1}} \cdot z_{h,t,n}^{(1)} \leq h_{h,t,n} \leq \overline{h_n} \cdot z_{h,t,n}^{(1)} \qquad \forall h, t, n \in \{H, T, N\} \tag{17d}$$

$$\sum_{n=1}^{N} h_{h,t,n} = h_{h,t} \qquad \forall h, t \in \{H, T\} \tag{17e}$$

Secondly, the power space of each head subinterval is divided into subintervals $m \in M_n$. In each of the power subintervals, a linear function approximates the relationship between the water flow rate and the output power. Constraints (18a)-(18e) below are the equivalent for the output power of constraints (17a)-(17e) on the net head.

$$\underline{p} = \overline{p_{0,n}} < \overline{p_{1,n}} < \cdots < \overline{p_{M_n,n}} = \overline{p} \qquad n = 1, 2, \dots, N \tag{18a}$$

$$z_{h,t,m,n}^{(2)} \in \{0, 1\} \qquad n = 1, 2, \dots, N; \; m = 1, 2, \dots, M_n \tag{18b}$$

$$\sum_{m=1}^{M_n} z_{h,t,m,n}^{(2)} = z_{h,t,n}^{(1)} \qquad n = 1, 2, \dots, N \tag{18c}$$

$$\overline{p}_{m-1,n} \cdot z_{h,t,m,n}^{(2)} \leq p_{h,t,m,n} \leq \overline{p}_{m,n} \cdot z_{h,t,m,n}^{(2)} \qquad \forall h, t, m, n \in \{H, T, M, N\} \tag{18d}$$

$$\sum_{n=1}^{N} \sum_{m=1}^{M_n} p_{h,t,m,n} = p_{h,t} \qquad \forall h, t \in \{H, T\} \tag{18e}$$

Now, a discrete grid has been defined on the net head and power dimensions. On each of the spaces defined as the intersection of a subinterval of head and power output, the water flow is approximated by a plane defined in Equation (19).

$$q = \sum_{n=1}^{N} \sum_{m=1}^{M_n} \left(z_{h,t,m,n}^{(2)} \cdot \overline{q}_{m-1,n} + \frac{\overline{q}_{m,n} - \overline{q}_{m-1,n}}{\overline{p}_{m,n} - \overline{p}_{m-1,n}} \cdot \left(p_{h,t,m,n} - z_{h,t,m,n}^{(2)} \cdot \overline{p}_{m-1,n} \right) \right) \quad \forall h,t \in \{H,T\} \tag{19}$$

Despite the complexity of the formulation, the piecewise linear approximation is far from being accurate for few head subintervals. Indeed, in Equation (19), the power varies with the water flow but not with the head; there is no slope w.r.t. the head. Figure 10 depicts the approximation for three head and two water flow subintervals. Whereas the power subintervals are barely visible, the head subintervals stand out very clearly. Therefore, for having a more accurate modelling of the UPC, one must define more subintervals on the head space than on the power space.

By looking at Figure 10 in an interactive way (click on the figure), one can see that the planes defined on the subintervals of power and head exist even outside of the original UPC boundaries thereby allowing the machine to operate into infeasible zones. To prevent it, the UPC boundaries must be modelled. The different methods to approximate those boundaries are discussed hereafter.

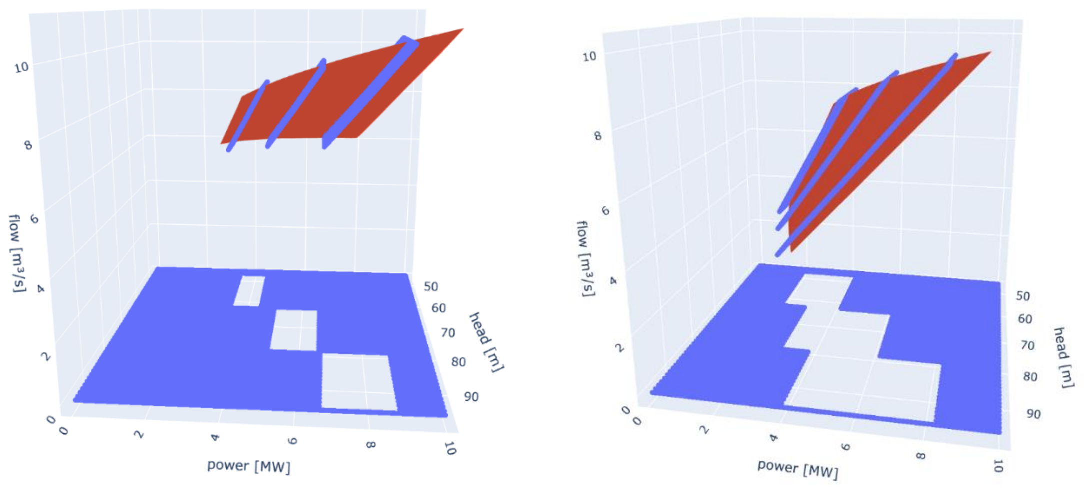


Figure 10: Plot of the original UPC (red) and the piecewise linear approximation (blue) considering three head subintervals and two flow subintervals for both the pump (left) and the turbine (right) mode. (click on the figures and download for an interactive view)

Constraint (20) enforces the discontinuity between the pumping mode and the turbine mode. It ensures the system is either in pump or turbine mode.

$$z^P_{h,t} + z^T_{h,t} \leq 1 \qquad \forall h,t \in \{H,T\} \tag{20}$$

The output power of the unit is bounded by the operation zones defined by the UPCs. Moreover, the range of power is more constrained to account for the capacity allocated to the reserve which can be called upon by the TSO. Constraints (21a)-(21b) restrict this power range for the pump and the turbine modes, respectively. They are to the power what constraints (10a)-(10b) are to the volume.

$$z^P_{h,t} \cdot \underline{p}^P_{h,t} + \sum_{r\in R^+} res^P_{h,r} \leq p^P_{h,t} \leq z^P_{h,t} \cdot \overline{p}^P_{h,t} - \sum_{r\in R^-} res^P_{h,r} \qquad \forall h,t \in \{H,T\} \tag{21a}$$

$$z^T_{h,t} \cdot \underline{p}^T_{h,t} + \sum_{r\in R^-} res^T_{h,r} \leq p^T_{h,t} \leq z^T_{h,t} \cdot \overline{p}^T_{h,t} - \sum_{r\in R^+} res^T_{h,r} \qquad \forall h,t \in \{H,T\} \tag{21b}$$

As clearly depicted by Figure 8, the UPC defines two zones forbidden to operations. The UPC cannot be embedded in the model because of its high complexity. Thus, one must ensure the optimisation model does not consider those zones as feasible. The complexity lies in the nonlinear variation of the safe operating zone bounds $[\underline{p}^i_{h,t}, \overline{p}^i_{h,t}]$ with the net head. In this thesis, the forbidden zones are modelled by four different methods which differ by their precision and their complexity: the box, the stepwise, the piecewise linear and the conservative piecewise linear approximations.

In a first naive approach, the UPC can be surrounded by a box, i.e., a parallelepipedal rectangle going from the smallest power available $\underline{P}^T_h$ to the largest one $\overline{P}^T_h$. That method has a low-complexity and is easy to implement (Equations (22a) & (22b)). However, vast spaces which are out of the UPC bounds (i.e., are not supposed to be feasible) are considered feasible by this approximation as displayed by Figure 11(a).

$$p^T_{h,t} \geq z^T_{h,t} \cdot \underline{P}^T_h + \sum_{r\in R^-} res^T_{h,r} \qquad \forall h,t \in \{H,T\} \tag{22a}$$

$$p^T_{h,t} \leq z^T_{h,t} \cdot \overline{P}^T_h - \sum_{r\in R^+} res^T_{h,r} \qquad \forall h,t \in \{H,T\} \tag{22b}$$

The second method is the stepwise approximation (Figure 11(b)) where the bounds $\underline{p}^T_{h,t}$ and $\overline{p}^T_{h,t}$ are set constant over each net head subinterval $n \in N$ to $\underline{F}^T_{h,n}$ and $\overline{F}^T_{h,n}$. A M-penalty ($M > \overline{F}^T_{h,n=N}$) is introduced so that the constraints (23a)-(23b) are deactivated if the $d_{h,t,n}^{PHES}$ binary variable is not equal to 1. The lower bound (23a) is enforced only if the unit is operating in turbine mode ($z_{h,t}^{T}=1$) allowing for $p_{h,t}^{T}$ to be worth 0 otherwise. This approximation is very conservative which means that no infeasible zone is supposed to be included in the feasible set approximation. However, being too conservative leads to discard feasible areas from the approximations. Decisions taken in those discarded area might have improved the final result. Therefore, a trade-off is needed between conservatism and risk. Interestingly, though this approximation is conservative, it still includes a very small part of infeasible region. This is due to the continuous space for establishing the bounds and the coarsely discrete space of the UPC scatter dataset causing rounding errors.

$$p_{h,t}^{T}-\sum_{r\in R^{-}} res_{h,r}^{T} \geq \underline{F}_{h,n}^{T}-\left(1-z_{h,t}^{T}\right)\cdot M -(1-d_{h,t,n}^{PHES})\cdot M \qquad \forall h,t,n\in\{H,T,N\} \tag{23a}$$

$$p_{h,t}^{T}+\sum_{r\in R^{+}} res_{h,r}^{T} \leq \overline{F}_{h,n}^{T}+(1-d_{h,t,n}^{PHES})\cdot M \qquad \forall h,t,n\in\{H,T,N\} \tag{23b}$$

A more accurate fit of the bounds can be obtained at the expense of the complexity by using the piecewise linear approximation (see Figure 11(c)). In the stepwise method, the operating zone in each interval is approximated by a rectangle which leads to high conservativeness. The method can be improved by using trapezoids. They will fit more closely the bounds. Despite being more precise, the accuracy of this second method is, akin to the stepwise one, linked to the discretization error over the head. The two sides of one trapezoid which fit the bounds are the linear monotonic functions $\underline{f}_{h,n}^{T}=\underline{A}_{h,n}^{T}h_{h,t}^{net}+\underline{B}_{h,n}^{T}$ and $\overline{f}_{h,n}^{T}=\overline{A}_{h,n}^{T}\cdot h_{h,t}^{net}+\overline{B}_{h,n}^{T}$ which leads to Constraints (24a) and (24b) instead of (23a) and (23b).

$$p_{h,t}^{T}-\sum_{r\in R^{-}} res_{h,r}^{T} \geq \underline{A}_{h,n}^{T}\cdot h_{h,t}^{net}+\underline{B}_{h,n}^{T}-\left(1-z_{h,t}^{T}\right)\cdot M -(1-d_{h,t,n}^{PHES})\cdot M \qquad \forall h,t,n\in\{H,T,N\} \tag{24a}$$

$$p_{h,t}^{T}+\sum_{r\in R^{+}} res_{h,r}^{T} \leq \overline{A}_{h,n}^{T}\cdot h_{h,t}^{net}+\overline{B}_{h,n}^{T}+(1-d_{h,t,n}^{PHES})\cdot M \qquad \forall h,t,n\in\{H,T,N\} \tag{24b}$$

In spite of a better fit, this piecewise approximation presents a massive drawback, it is not conservative enough and tends to consider feasible zones which are not (see Figure 11(c)), especially when the curvature radius is small. To solve this issue, we have developed a fourth approximation where we add/subtract a safety margin s_h to the bounds (see Equations (25a) & (25b)) to ensure the resulting feasible set is conservative as depicted by Figure 11(d).

$$p_{h,t}^{T} - \sum_{r\in R^{-}} res_{h,r}^{T} \geq \underline{A}_{h,n}^{T} \cdot h_{h,t}^{net} + \underline{B}_{h,n}^{T} + s_h - \left(1 - z_{h,t}^{T}\right) \cdot M - \left(1 - d_{h,t,n}^{PHES}\right) \cdot M \qquad \forall h,t,n \in \{H,T,N\} \tag{25a}$$

$$p_{h,t}^{T} + \sum_{r\in R^{+}} res_{h,r}^{T} \leq \overline{A}_{h,n}^{T} \cdot h_{h,t}^{net} + \overline{B}_{h,n}^{T} - s_h + \left(1 - d_{h,t,n}^{PHES}\right) \cdot M \qquad \forall h,t,n \in \{H,T,N\} \tag{25b}$$

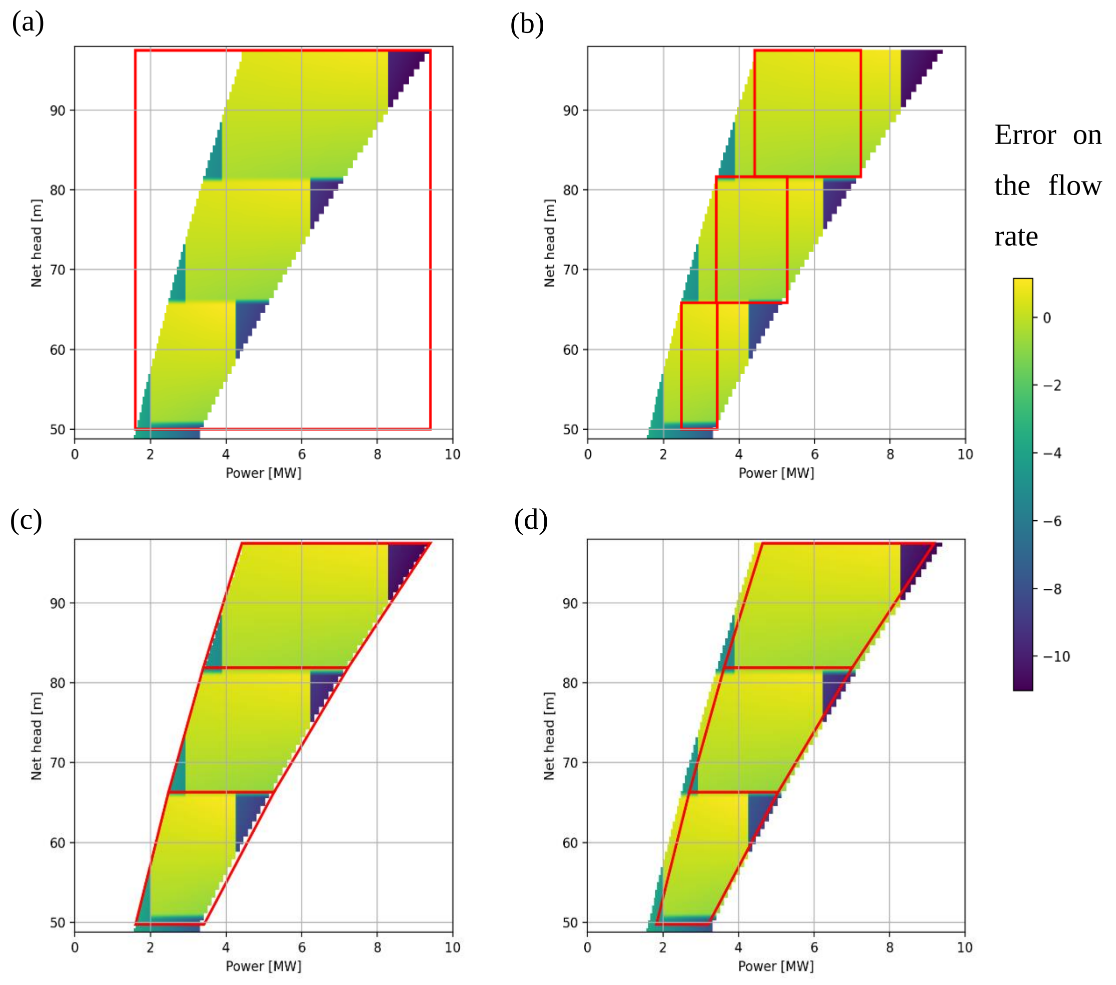


Figure 11: Illustration of the methods for approximating the bounds of the turbine UPC considering three subintervals of head; (a) is the surrounding rectangle; (b) is the stepwise approximation; (c) is the piecewise approximation; (d) is the conservative piecewise approximation (click on each plot to obtain a gif with the number of subintervals ranging from one to seven).

Figure 11 reveals another important characteristic of the optimization problem, the quality of the fit between the approximation and the genuine UPC. It stands out clearly that the planes associated with the piecewise approximation of the UPC exist only over an interval of head and power forcing the operating point to be on one of these planes. In addition, the operating point must be within the bound approximation (red plain lines). Therefore, the feasible set becomes the intersection of those two spaces. Visually, it means that the blue spaces are not part of the final feasible set, even if they are within the red quadrilaterals. It is important to notice that the error computed on the bluish spaces has no physical meaning since the piecewise linear approximation does not exist on those spaces. To get the given errors, the value of the piecewise approximation was set to zero. By doing so, the physical curve and its limits (blue and yellowish spaces) stands out more clearly. However, from the optimization problem point of view, the bluish spaces are white.

2.2 Conclusion

Optimization is a very powerful tool since it ensures any decision-making process is optimal with respect to the problem defined. The main hurdle lies in the definition of this problem. Indeed, the solution is guaranteed to be the global optimum only if the problem is convex which leads to many limitations on its definition. Often, approximations such as the ones presented above are needed to set the problem in a convex way, but they introduce inaccuracies which can lead to wrong decisions. In the application of the PHES, the operator incurs a financial loss because it cannot uphold his commitments.

Furthermore, defining the problem to keep it convex increases the complexity for the implementer. In this application, the two UPCs need to be modelled using a 3-D piecewise linear approximation and their feasible operating zone must be bounded. The breaking points between the subintervals have been chosen uniformly distributed.

Naturally, by choosing the breaking points smartly, a more accurate fit of the curves can be expected without increasing the complexity of the problem (i.e., the number of variables). However, a naïve approach would be for the implementer to try many point combinations and retain the optimal one. A smarter approach is to leverage the power of machine learning techniques for optimizing the position of the breaking points. It is what is realised in the following chapter with the help of deep fully connected NNs.

3 Machine Learning

This chapter is divided into three main sections. It starts by a brief introduction to univariate multilinear regressions. Afterwards, the key elements of the neural networks (NNs) and their working process are presented. Deep neural networks are introduced and discussed. Lastly, different methods for embedding NNs into model-based optimization are reviewed. Only the elements relevant to the understanding of this work are explained for the sake of brevity.

3.1 Linear regression

A linear regression is a linear model for establishing the *best* relationship between response variables given some explanatory variables. The *best* must be understood as the relation with the least error given a model (Figure 12). If there is one single response variable, the problem is said univariate, multivariate in the contrary. Similarly, if there is one single explanatory variable, the problem is said simple, multiple in the contrary. In this work, the focus is on univariate multiple linear regressions.

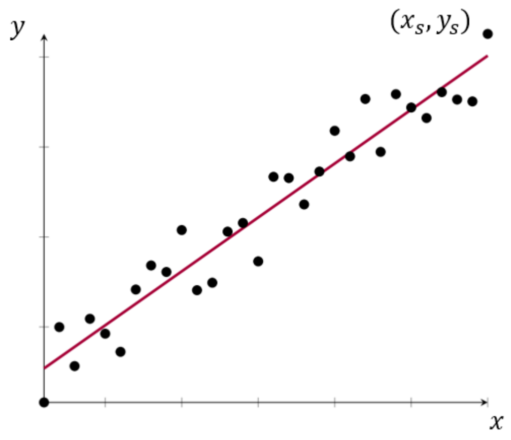


Figure 12: Univariate simple linear regression using a least squared error minimization [24].

Regression is a set of machine learning techniques where the parameters of a given function model are tuned to minimize a loss function over a set of samples. Given a set of samples, sometimes called observations, $S = \{(\bar{x}_1, y_1), \dots, (\bar{x}_S, y_S)\}$, the multiple linear regression model is

$$y_s = a_1 \cdot x_{s,1} + a_2 \cdot x_{s,2} + \cdots + a_M \cdot x_{s,M} + \varepsilon_s \qquad \forall s \in \{S\} \tag{26}$$

where ε_i is the ith error and the $\bar{x}_i$'s are vectors of size M. In matrix notation, the S equations are summarized as

$$Y = XA + \mathrm{E} \tag{27}$$

where Y is a vector of size $S \times 1$, X is the matrix of the explanatory variables of size $S \times M$, A is the vector of the coefficients of size $M \times 1$, and E is the vector of the errors of size $S \times 1$.

Without considering the vector of errors, the system of equations at Equation (27) is solvable, only and only if M equals to S considering that none of the S equations (Equation (26)) can be obtained as a linear combination of the other $S - 1$ equations.

However, in linear regression, it is conventional to have $M \ll S$. Therefore, the goal is not to solve the system of equations but to find the parameters a_m which minimise a function of the error ε_s. A very common function to minimise is the squared error (Equations (28)) which leads to find the least squared error regression.

$$\min_A \sum_{s=1}^{S} {\varepsilon_s}^2 = \min_A ||Y - XA||^2 \tag{28}$$

where $||.||$ indicates the Euclidian distance.

By imposing the derivative of $||Y - XA||^2$ to 0, one can find the system of equations which yields the a_s parameters without using an optimization linear solver (Equation (29)) [24], [25].

$$X^T XA = X^T Y \tag{29}$$

3.2 Neural Networks

Linear regressions, per their nature, are limited to the representation of linear dependencies, which may poorly reflect complex (i.e., non-linear) processes. For non-linear relationships, one tool has tremendously developed, the neural networks (NNs). NNs belong to the machine learning (ML) field which is itself a subpart of artificial intelligence (AI). In ML, the system learns autonomously by surveying a large amount of data without being explicitly programmed [26].

As per their name, NNs or rather, artificial neural networks (ANNs) are an assembly of artificial neurons. They are being used increasingly because they have the power to approximate any computable function, including nonlinear functions, to an arbitrary accuracy. Given two vector spaces, NNs are theoretically able to realise the mapping (i.e., for any valid input vector X, the NN is able to compute an output vector Z) between them whatever their dimensions. It means NNs can perform any task a standard digital computer can do [27]. Of course, in practice, physical considerations such as computational time and resources must be taken into account. A definition focusing on NNs' description, can be as follows:

"*A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns.*[28]"

Several elements appearing in the definition are worth being developed into more detail:

1. "*units or nodes*"
2. "*interunit connection strengths, or weights"*
3. "*process of adaptation to, or learning"*

3.2.1 Single Unit

The unit is the core component of a NN. The first unit introduced into the earlier works performed by Frank Rosenblatt in the 50's is named the perceptron or threshold logic unit (TLU). Figure 13 displays a typical structure of a unit in a NN.

A perceptron takes as input a binary vector of size N and output one single binary. Every binary input x_i is weighted by a scalar W_i and all those products are summed together and yield the activation level $\hat{y}$ (Equation (30)). Then, if the activation level $\hat{y}$ is greater than the scalar bias θ, the unit is said to be activated or fired and its output is set to one; otherwise, the output is null (Equation (31))[29], [30].

$$\hat{y} = W_1 \cdot x_1 + W_2 \cdot x_2 + \cdots + W_N \cdot x_N \tag{30}$$

$$y = \begin{cases} 0 \text{ if } \hat{y} \geq \theta \\ 1 \text{ if } \hat{y} < \theta \end{cases} \tag{31}$$

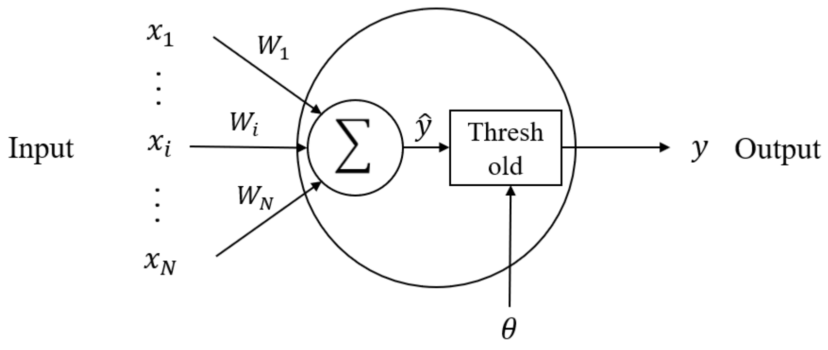


Figure 13: Sketch of a typical perceptron unit, based on [31].

In the more recent developments, the input and output variables are not constrained to be binaries anymore. Indeed, they are now defined as real scalars. Therefore, more flexibility is possible in the choice of the activation function F as its output has not to be binary anymore. Unlike the perceptron, $\hat{y}$ is now obtained by adding a constant b named bias to the sum of the weighted inputs (Equation (32)) as depicted by Figure 14.

$$\hat{y} = W_1 \cdot x_1 + W_2 \cdot x_2 + \cdots + W_N \cdot x_N + b \tag{32}$$

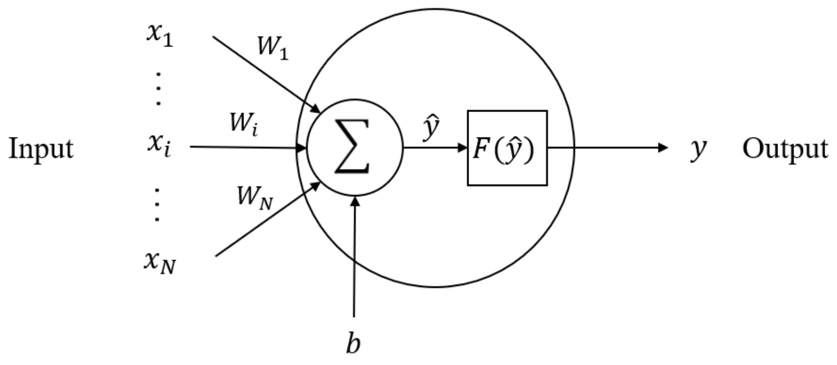


Figure 14: Sketch of a typical unit in a NN where F is the activation function, based on [31].

The output of the unit is obtained by passing $\hat{y}$ in the activation function. In this work, two activation functions are used but many other exist (ReLU6, sigmoid, softmax, softplus…). Firstly, the rectified linear unit (ReLU) function shown by Figure 15 (a). The ReLU can be mathematically defined following either Equations (33a) or (33b). The leaky rectified linear unit (Leaky ReLU) is a modified ReLU where the left hand-side of the function is sloped upward by a coefficient α as can be seen on Figure 15 (b). This leads to Equation (34a). Akin to the ReLU, the Leaky ReLU can be reformulated in a second way (Equation (34b)) if α is smaller than 1.

$$y = \begin{cases} \hat{y} \text{ if } \hat{y} \geq 0 \\ 0 \text{ if } \hat{y} < 0 \end{cases} \tag{33a}$$

$$y = \max(0, \hat{y}) \tag{33b}$$

$$y = \begin{cases} \hat{y} \text{ if } \hat{y} \geq 0 \\ \alpha \cdot \hat{y} \text{ if } \hat{y} < 0 \end{cases} \tag{34a}$$

$$y = \max(\alpha \cdot \hat{y}, \hat{y}) \qquad \forall \alpha \in \;]-\infty, 1[\tag{34b}$$

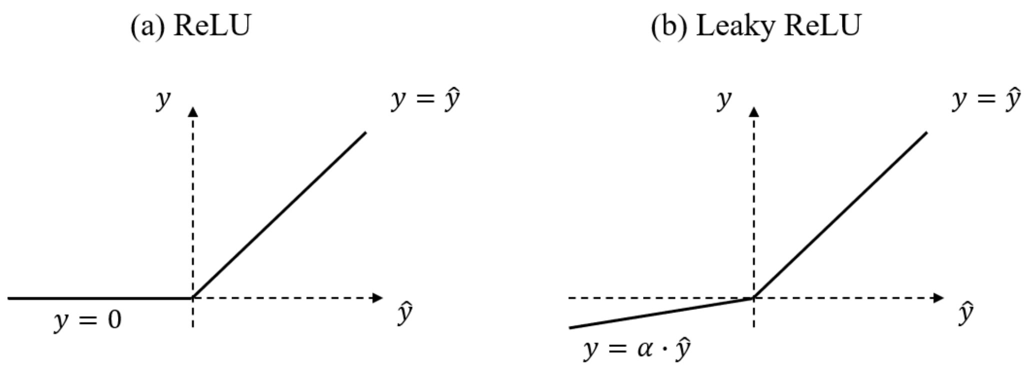


Figure 15: Rectified Linear Unit (ReLU) and Leaky ReLU activation functions.

3.2.2 Networks

Using the unit described above as the core element, one can start building a NN. First, the unit can be duplicated numerous times to form a layer of a given dimension. Once the layers are formed, they can be connected together to produce a neural network.

In conventional feedforward NNs, one can distinguish three main types of layers depicted by Figure 16. Firstly, the input layer represents the size of the vector containing the inputs. Interestingly, no operation on the data is performed at this level (do not be tricked by the visual representation). The last layer is composed of units as seen previously in equal number to the dimension of the output vector. A NN is at least made of the input layer and the output layer. But nothing prevents to add more layers in between the two aforementioned ones. These extra layers are named deep layers. The NN is now named deep NN. A deep NN allows us to perform deep learning which consists in realizing machine learning using a deep NN [32].

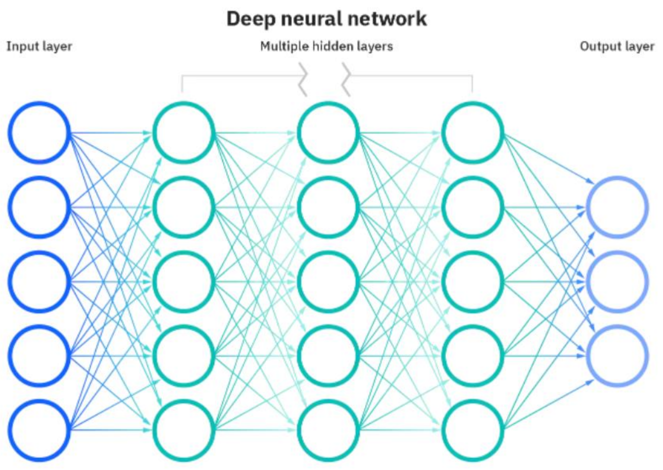


Figure 16: Deep fully connected neural network [33].

In conventional feedforward NNs, the information is propagated to the next layer only, meaning that a layer gets its input from the preceding layer and transmits the output to the following layer. Let us also note that the units of a same layer are not connected together and thus, do not exchange information [34].

In this work, fully connected NNs are used. It describes the way the neurons (or units) are connected together. A neuron in a given layer is connected to every single neuron in the following layer. To each of these connections is associated a weight while a bias is associated to each neuron. The number of layers, the number of units in each layer and the way they are interconnected defines the architecture (and the modelling power) of the NN.

The existence of a link between two neurons does not signify this connection is active. Indeed, the weight associated to the connection can be set to 0 which is equivalent to having no link. This property, named sparsity, is purposely looked for because it increases the speed of the NN when it must predict an output [35].

In conclusion, even with this brief introduction to the neural network units and the possible ways to organise them, one can comprehend the massive number of combinations which is available. Therefore, for every problem, an architecture can be wisely picked to achieve satisfying performances.

3.2.3 Linear Algebra for NNs

In subsection 3.2.1 Single Unit, the formulas used for one NN are presented. Afterwards, in 3.2.2 Networks, the organization of fully connected NNs is developed. In order to be more efficient, the computations within a NN use linear algebra. Hence, the equations using matrixes and vectors are given. They allow to compute the results per layer instead of per neuron.

In a fully connected NN containing L layers, the input of a layer l containing $n^{(l)}$ neurons can be obtained by applying Equation (35). In this equation, $\hat{y}^{(l)}$ is a vector of size $n^{(l)}$ representing the input values of the layer l. $b^{(l)}$, the vector containing the biases, has the same dimension. Then, $y^{(l-1)}$ is a vector of size $n^{(l-1)}$ representing the output values of the layer $l-1$. Finally, $W^{(l)}$ is the matrix containing the weights of the bounds between the layer $l-1$ and l. It has a size of $n^{(l)} \times n^{(l-1)}$.

$$\hat{y}^{(l)} = W^{(l)}y^{(l-1)} + b^{(l)} \qquad \forall l = 1, 2, \dots, L \qquad (35)$$

As all the neurons of a same layer have the activation function F, the output of the layer l can be obtained following Equation (36). In Equation (36), F must be replaced by the expression of the activation function, for example, Equation (34b) if it is a Leaky ReLU.

$$y^{(l)} = F(\hat{y}^{(l)}) \qquad \forall l = 1, 2, \dots, L \qquad (36)$$

By applying Equations (35) and (36) to each layer, one can easily compute the output of a fully connected NN based on its inputs and its parameters. It is also worth noticing that the non-linearity of the NN can only come from the activation F. Would that function be linear, the NN would be linear since Equation (35) is a linear system of equations. In that case, a deep NN is useless because, whatever the number of layers, one would obtain a linear system of equations which could be simplified into Equation (35).

3.2.4 Training of NN

According to the Universal Approximation Theorem (UAT), a NN is able to establish a mapping between its input space X and its output space Z [36]. In order to make this mapping as accurate as possible (i.e., ensure that the estimation output by the NN is closed to the real value), the parameters of the NN must be optimized accordingly. Those parameters are the weights and biases hereabove.

There exist three main ways to train a NN (i.e., tune the NN' parameters to obtain an accurate mapping), namely: the supervised learning, the unsupervised learning and the reinforcement learning (see Figure 17). In supervised learning, the dataset is said to be labelled. It means the output value $z \in Z$ for a given input $x \in X$ is known. In unsupervised learning, there is no information about z. Finally, in reinforcement learning, the agent (which can be a NN) learns by trial and error. It acts and observe the reward yields by that action. For the sake of exhaustivity, let us mention semi-supervised learning where some of the data are labelled, and other not [37], [38].

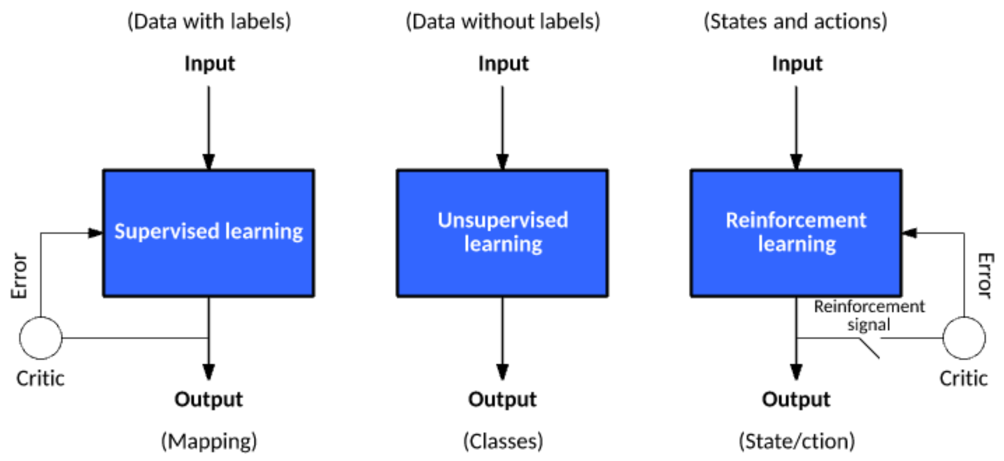


Figure 17: Sketch of the three main learning processes for machine learning [39].

Gradient Descent

To optimize the value of the parameters, the main technique is named Gradient Descent (GD). As explained in Appendix A. Vocabulary and formalism, an objective function to optimize must be given to the solver. This function is named loss function when speaking about NNs training. The goal is to find the weights and biases (i.e., the NN parameters) values which minimize the loss function. However, since NNs are nonlinear and non-convex, there is no warranty of finding the global optimum (see Convex vs. non-convex). What the solver will yield is a set of weights and biases outputting *good enough* results w.r.t. the loss function selected [40].

To understand the definition of the training problem as an optimization problem, one may start by considering one single training sample. For a target output z, and a NN output $\hat{z}$, the loss function is defined as $L(z, \hat{z})$. A very commonly used loss function is the squared error loss defined by

$$L(z, \hat{z}) = \frac{1}{2}(z - \hat{z})^2 \tag{37}$$

Now, the output $\hat{z}$ is the output of the NN during training. Therefore, by representing the NN by the function $g: \mathbb{R}^m \times \mathbb{R}^t \longrightarrow \mathbb{R}^n; (x, \theta) \longmapsto g(x, \theta)$, the per training sample loss becomes $L\big(z, \hat{g}(x, \theta)\big)$. The function $\hat{g}(x, \theta)$ executes some NN upon an input x with the weights and biases gathered into θ, $g(x, \theta)$ is the actual relationship the NN must fit. The total loss $Loss$ is obtained by averaging the loss over the whole training set χ for of all the pairs (x, z) (Equation (38)).

$$Loss(\theta, \chi) = \frac{1}{|\chi|} \sum_{\forall (x,z) \in \chi} L\big(z, \hat{g}(x, \theta)\big) \tag{38}$$

The training process of the NN can be summarized in Equation (39) where $Loss(\theta, \chi)$ is a nonlinear function to be minimized by finding the appropriate NN parameters θ.

$$\min_{\theta} Loss(\theta, \chi) \tag{39}$$

The problem can be solved using an off-the-shelf nonlinear solver such as Adam [41]. Those nonlinear solvers rely on the gradient descent algorithm. The gradient of a function is the vector of the partial derivatives of this function w.r.t. its input variables [42]. In the case of the loss function, the variables are the NN parameters belonging to θ.

Gradient vector - Be a function $f: E \subset \mathbb{R}^n \longrightarrow \mathbb{R}$, $a \in \mathrm{Int}(E)$ and f differentiable in a. The gradient vector of f in a is the vector made of the partial derivatives of f in a. It is usually noted $\nabla f(a)$.

$$\nabla f(a) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(a) \\ \cdots \\ \frac{\partial f}{\partial x_n}(a) \end{pmatrix}$$

The gradient of f in a sample point a ($\nabla f(a)$) indicates the uphill direction. Consequently, the downhill direction is obtained by moving towards $-\nabla f(a)$. The direction is known but one must determine the distance by which to move [40]. Of course, this distance depends on the magnitude of the gradient vector. Nevertheless, it is better to tune it to avoid overshooting the minimum by moving too much. To that end, a parameter ϵ named learning rate multiplies the gradient vector which leads to Equation (40) where θ_j is a parameter of the NN, either a weight or a bias. In practice, ϵ is not kept constant. Instead, it gradually decreases [34].

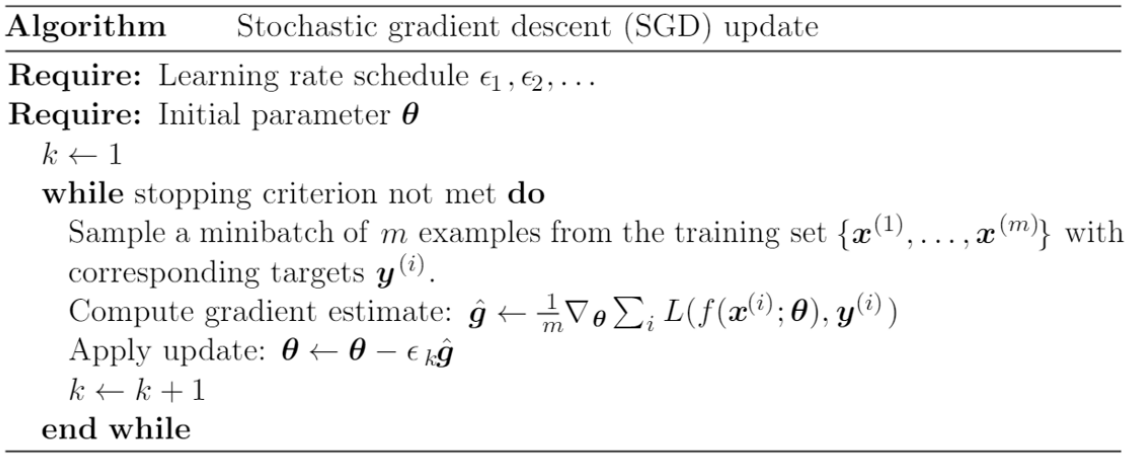
$$\theta_j^{(i)} = \theta_j^{(i-1)} - \epsilon \cdot \frac{\partial Loss}{\partial \theta_j} \qquad \forall \theta_j \in \theta \tag{40}$$

Stochastic Gradient Descent (SGD)

Computing the gradient requires a measure of the loss between the current output of the NN being trained and the target output. In the gradient descent algorithm, the loss is computed over the full training set. This operation is extremely resource-intensive because

ML techniques needs massive datasets to reach great accuracy. In order to alleviate the gradient vector computation, one can estimate it by sampling m pairs (x, z) from the dataset χ. This subset of χ is named minibatch. A minibatch is usually made of a power of 2 number of elements to allow parallel computing. The size of the minibatch must be chosen carefully [34], [40]. If the minibatch is too small, the estimation of the gradient is extremely noisy degrading the performances. On the other hand, if the minibatch is too big, the accuracy gained is not worth the computational time spent on it.

The procedure for updating NN parameters following the SGD method is described by Figure 18. It is the same as the Gradient Descent, only it is performed on a minibatch and not the full dataset.



```
Algorithm    Stochastic gradient descent (SGD) update
Require: Learning rate schedule $\epsilon_1, \epsilon_2, \dots$
Require: Initial parameter $\boldsymbol{\theta}$
  $k \leftarrow 1$
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ with
    corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \hat{\boldsymbol{g}}$
    $k \leftarrow k + 1$
  end while
```

Figure 18: Stochastic gradient descent update.

Backward Propagation Algorithm

In the previous subsection, the SGD algorithm has been explained. However, one may wonder how the partial derivative $\frac{\partial Loss}{\partial \theta_j}$ can be computed for every parameter belonging to the NN. The algorithm which allows to calculate the gradient vector is named Backward Propagation Algorithm (BPA) and relies heavily on the chain rule. In order to explain clearly the BPA and to fix properly the concepts seen previously about NN, an example is presented based on [43].

The example is a four-layer fully connected NN with an input dimension of four, an output dimension of one while the deep layers contain two neurons each. The deep neurons have the ReLU activation function while the output neuron has the identity $(y = \hat{y})$ activation function. Remember the first layer is not made of units per say, it is just a visual representation of the input size.

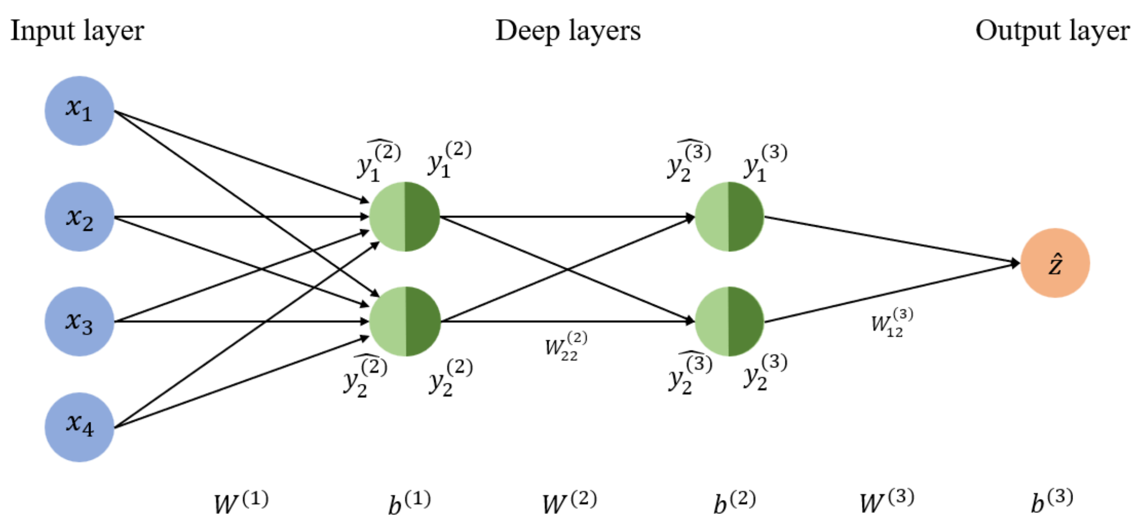


Figure 19: NN example for performing the backward propagation algorithm.

The goal of the example is to determine $\frac{\partial Loss}{\partial W_{22}^{(2)}}$ by applying the BPA. Firstly, since the derivative of the expectation of a variable t is the expectation of the derivative of this same variable (Equation (41a)) and $Loss$ is the expectation of the $L^{(i)}$ one can write Equation (41b) for a minibatch of size m.

$$[\mathrm{E}(t)]' = E[(t)'] \tag{41a}$$

$$\frac{\partial Loss}{\partial W_{22}^{(2)}} = \frac{\partial E\left(L^{(i)}\right)}{\partial W_{22}^{(2)}} = \frac{\partial \frac{1}{m} \cdot \sum_{i=1}^{m} L^{(i)}}{\partial W_{22}^{(2)}} = \frac{1}{m} \cdot \sum_{i=1}^{m} \frac{\partial L^{(i)}}{\partial W_{22}^{(2)(i)}} \tag{41b}$$

By using the chain rule, one can write[3]

$$\frac{\partial L}{\partial W_{22}^{(2)}} = \frac{\partial L}{\partial \hat{z}} \cdot \frac{\partial \hat{z}}{\partial y_2^{(3)}} \cdot \frac{\partial y_2^{(3)}}{\partial \widehat{y_2^{(3)}}} \cdot \frac{\partial \widehat{y_2^{(3)}}}{\partial W_{22}^{(2)}} \tag{42}$$

Carrying out an analysis of the right-hand side part of Equation (42) is very enriching. The first factor can be determined once the formula used to calculate the loss is known. In this example, the very common mean squared error loss is considered (Equation (37)). The derivative is given by Equation (43) where z the target value.

$$\frac{\partial L}{\partial \hat{z}} = \frac{1}{2} \cdot 2 \cdot (z - \hat{z}) \cdot (-1) = \hat{z} - z \tag{43}$$

[3] The index (i) is voluntarily dropped for the sake of clarity.

The second term is the partial derivative of the NN output w.r.t. $y_2^{(3)}$. Remember, there is no difference between the input and output of the last neuron because its activation function is the identity function. The second factor can be calculated very easily by remembering Equation (32) which applied to our specific case is

$$\hat{z} = W_{11}^{(3)} \cdot y_1^{(3)} + W_{12}^{(3)} \cdot y_2^{(3)} + b_1^{(4)} \tag{44a}$$

Therefore, the result of the second derivative of the right-hand side of Equation (42) is $W_{12}^{(3)}$.

The third term of the equation is the partial derivative of the activation function of the neuron. Indeed, it is the activation function which establishes the relation between $\widehat{y_i^{(l)}}$ and $y_i^{(l)}$. For a ReLU, the result of the derivative can either be 0 or 1 depending on the sign of the neuron input (i.e., the sign of $\widehat{y_2^{(3)}}$ for the sample (i) considered).

Finally, the last factor can be calculated by using once more Equation (32) which applied to our specific case is

$$\widehat{y_2^{(3)}} = W_{21}^{(2)} \cdot y_1^{(2)} + W_{22}^{(2)} \cdot y_2^{(2)} + b_2^{(3)} \tag{44b}$$

Therefore, the result of the last derivative of Equation (42) is $y_2^{(2)}$ and it can now be rewritten as

$$\frac{\partial L}{\partial W_{22}^{(2)}} = (\hat{z} - z) \cdot W_{12}^{(3)} \cdot \text{sign}(\widehat{y_2^{(3)}}\,) \cdot y_2^{(2)} \tag{45}$$

The m results obtained by applying Equation (45) to each sample (i) of the minibatch are averaged. Then the parameter $W_{22}^{(2)}$ can be updated following the SGD algorithm, more precisely Equation (40).

ReLU and SGD

Because the ReLU function is piecewise linear, it brings advantages and challenges, especially when applying the SGD to a NN with neurons whose activation function is a ReLU.

Firstly, the derivative of the ReLU function is straightforward to compute. It is the equivalent of a sign function as explained in the previous section. This allows to quicken the BPA.

Secondly, the ReLU is not differentiable in 0 because it is a discontinuity point. Nevertheless, the value of the derivative at this point can be imposed, for example to 1 to keep the continuity with the derivative of the positive part of the function. It means that 0 is considered as a positive number by the function $\text{sign}(.)$ used to represent the derivative of the activation function.

Furthermore, because the negative side of the ReLU is a constant (worth 0), the gradient on that side is null. Therefore, if the input of the ReLU $\hat{y}$ is negative, $\text{sign}(\hat{y})$, the derivative of the ReLU, is null. It leads Equation (45) to be null, as well. It means that, if all the inputs of a neuron (whose activation function is the ReLU) over a minibatch are negative, it will prevent the gradient from flowing during the backpropagation causing the parameters not to update. If the parameters of a neuron get stuck to values which cause the neuron inputs to be negative, not only will they not update but also the neuron will only output 0. This neuron is so-called dead because it does not learn anymore (since its parameters are stuck) and produces only null outputs.

This syndrome of dead neurons is, if dompted properly, a great opportunity to increase the sparsity (i.e., the number of inactive connections between neurons because the weights associated to them is close to 0) of the NN. Good practices to avoid the dead neuron syndrome are among others to initialize the weights and the biases to positive values, and make sure to pick a learning rate small enough. Another solution is to use the Leaky ReLU instead of the ReLU since the negative side of the function still present a gradient of value α as visible in Figure (14) [40], [44].

3.3 Embedding Neural Network into Optimization

In the previous section, NNs are briefly presented. The mathematical relationships to obtain the input of one single unit and compute its output are given. In this section, one will learn how to translate the non-linear mapping yielded by a NN into piecewise-linear equations. Those equations are constraints which can be easily embedded within an optimisation problem.

Firstly, one must remember that the equality constraints must be linear in order to have a convex optimization problem. Therefore, it comes handy that Equation (35) linking the output of the layer $l - 1$ to the input of the layer l is linear. No work is necessary here and the equation can be included as such within the optimization formulation.

The arduous part lies in Equation (36) and the translation into equations of the activation function. As previously explained, the use of a deep NN (DNN) makes sense if and only if the activation function is nonlinear. Otherwise, the use of a shallow NN (i.e., a NN with no hidden layer) yields the same results as the DNN. In this work, we study two activation functions: the ReLU and the Leaky ReLU. Both are piecewise linear functions which allow to translate them into piecewise linear equations without introducing any approximation to the original functions.

3.3.1 ReLU

Formulation 1

The ReLU function can be expressed in two different ways following Equations (33a) and (33b). For each of these equations, a formulation is possible. Equation (33a) is already a piecewise expression. The introduction of a binary variable b is necessary to track on which segment of the function the operation takes place. A simple way to express Equation (33a) is therefore as follows

$$x_1 \leq 0 \text{ and } x_2 \geq 0 \qquad \text{(46a) - (46b)}$$

$$b \in \{0, 1\} \qquad \text{(46c)}$$

$$x_1 \geq \hat{y} \cdot (1 - b) \qquad \text{(46d)}$$

$$x_2 \leq \hat{y} \cdot b \qquad \text{(46e)}$$

$$y = x_2 \qquad \text{(46f)}$$

$$\hat{y} = x_1 + x_2 \qquad \text{(46g)}$$

where $\hat{y}$ is the input of the activation function and y, its output.

In the above formulation, if b is worth 1 then x_1 is also null by Equations (46a) and (46d). This triggers $\hat{y} = x_2$ by Equation (46g) and through Equation (46f), $y = \hat{y}$. In the case of $b = 0$, x_2 is null by Equations (46b) and (46e). Equation (46f) imposes $y = 0$.

The current formulation presents two major drawbacks. Firstly, it is not a linear formulation since there are some products between the decisions variables e.g., $\hat{y} \cdot b$ in Equation (46e). Secondly, the formulation works for any input value of $\hat{y}$ which makes it not tight. Increasing the tightness of the formulation reduces the computational time needed by the optimization solver. A smarter formulation is obtained by introducing lower and upper bound of $\hat{y}$ as follows

$$x_1 \leq 0 \text{ and } x_2 \geq 0 \qquad \text{(47a) - (47b)}$$

$$b \in \{0, 1\} \qquad \text{(47c)}$$

$$x_1 \geq \underline{\hat{y}} \cdot (1 - b) \qquad \text{(47d)}$$

$$x_2 \leq \overline{\hat{y}} \cdot b \tag{47e}$$

$$y = x_2 \tag{47f}$$

$$\hat{y} = x_1 + x_2 \tag{47g}$$

where $\underline{\hat{y}}$ is the lowest bound of $\hat{y}$ (i.e., the minimal value that can take $\hat{y}$) and $\overline{\hat{y}}$, its highest bound (i.e., the maximal value that can take $\hat{y}$).

This new formulation works similarly to the previous one, but the bilinear products have disappeared since $\underline{\hat{y}}$ and $\overline{\hat{y}}$ are constant values and the formulation is tighter since $\hat{y}$ can go from $\underline{\hat{y}}$ to $\overline{\hat{y}}$ instead of $-\infty$ to $+\infty$. The values of the bounds can be obtained and saved during the testing of the neural network.

Formulation 2

A similar reasoning can be applied to the second definition of the ReLU function (Equation (33b)). The following formulation can be developed

$$y \geq 0 \tag{48a}$$

$$b \in \{0, 1\} \tag{48b}$$

$$y \leq \hat{y} - \underline{\hat{y}} \cdot (1 - b) \tag{48c}$$

$$y \geq \hat{y} \tag{48d}$$

$$y \leq \overline{\hat{y}} \cdot b \tag{48e}$$

In this formulation, when b is null, y is also null by Equations (48a) and (48e). And Equation (48d) makes sure that $\hat{y}$ is negative. Therefore, it yields an output of zero for a negative input which matches the left part of the ReLU. If $b = 1$, then y must be greater than or equal to $\hat{y}$ (Equation (48d)) and smaller than or equal to $\hat{y}$ by Equation (48c). Consequently, $y = \hat{y}$ which is the expected result. This formulation was proposed in [45].

3.3.2 Leaky ReLU

Formulation 1

Likewise, the Leaky ReLU has two formulations (Equations (34a) and (34b)). The first formulation is the piecewise formulation, and its below translation is an extension of Equations (47a) - (47g) to consider the slope α in the negative part of the function. One might notice that, by setting $\alpha = 0$, Equations become similar to Equations (47a) - (47g) of the ReLU.

$$x_1 \leq 0 \text{ and } x_2 \geq 0 \tag{49a) - (49b}$$

$$b \in \{0,1\} \tag{49c}$$

$$x_1 \geq \underline{\hat{y}} \cdot (1-b) \tag{49d}$$

$$x_2 \leq \overline{\hat{y}} \cdot b \tag{49e}$$

$$y = \alpha \cdot x_1 + x_2 \tag{49f}$$

$$\hat{y} = x_1 + x_2 \tag{49g}$$

For $b = 1$, the interpretation of this formulation is the same as Equations (47a) - (47g). For $b = 0$, Equations (49b) and (49e) impose $x_2 = 0$. Equation (49g) sets $\hat{y} = x_1$ while Equation (49f) ensures that $y = \alpha \cdot x_1$. Therefore, $y = \alpha \cdot \hat{y}$ which is the desired output for a negative $\hat{y}$ (enforced by Equations (49a) and (49d)).

Formulation 2

This second translation of the Leaky ReLU into equations relies on its second formulation (Equation (34b)) and is an extension of Equations (48a) - (48e) for the ReLU.

$$y \geq \alpha \cdot \hat{y} \tag{50a}$$

$$b \in \{0,1\} \tag{50b}$$

$$y \leq \hat{y} - \underline{\hat{y}} \cdot (1-b) \tag{50c}$$

$$y \geq \hat{y} \tag{50d}$$

$$y \leq \overline{\hat{y}} \cdot b + \alpha \cdot \hat{y} \tag{50e}$$

If b is null, then Equations (50a) and (50e) impose $y = \alpha \cdot \hat{y}$. By introducing this last equation into Equation (50d), one gets $\alpha \cdot \hat{y} \geq \hat{y}$, $\alpha \in\ ]-\infty, 1[$ which implies $\hat{y}$ is negative. In the case $b = 1$, Equations (50c) and (50d) set $y = \hat{y}$. Equation (50a) becomes $\hat{y} \geq \alpha \cdot \hat{y}$ which is verified for any positive $\hat{y}$ if $\alpha \in\ ]-\infty, 1[$.

3.4 Conclusion

In this chapter, some basic concepts of machine learning have been presented. Firstly, the mathematics behind the linear regression methods have been explained. Afterwards, the fundamentals of NNs were introduced describing what a neuron is, its functioning, its interactions with the surrounding neurons and the training process of NNs. Finally, two reformulations for the ReLU and Leaky ReLU activation functions have been given.

4 Study Case

In the previous chapters, one has learnt about the fundamental concepts of optimization applied to PHES and machine learning. Moreover, the way to link both together has been explained. It is now time to gather all the knowledge accumulated over the past chapters and apply it to a study case. In this chapter, we realise some machine learning informed optimisation to operate optimally the UPHES site of Maizeret.

Three different methods to approximate the UPCs are studied. Firstly, we review the optimization method presented in Methodology which is the starting point of this work. Then, we approximate the two UPCs by a data-driven plane computed following a linear regression. At last, we train some NNs to fit the UPCs and we translate them into the optimization problem. For the three approximations of the UPCs, we investigate the impact of various approximations of the UPC bounds. We stress the importance for the reader to understand the difference between the approximation of the UPC itself and the approximation of its bounds. The latter defines the feasible operating domain for the UPC but not the value of the UPC! To avoid confusion, the UPC bound approximation is referred as bound approximation in the rest of the chapter.

4.1 The Pumped-Hydro Energy Storage (PHES) Unit of Maizeret

The study case of this Master Thesis is a hypothetical UPHES unit in a Belgian decommissioned quarry in Maizeret. It is considered that the operator of the unit can take part into the electricity markets in (i) bidding electricity on the day-ahead market, (ii) offering reserves to the TSO. The technical features of the site are available in Figure 20.

The UPCs for this hypothetical unit are presented in Figure 7. They have been obtained through the SmartWater project [46] which investigated the feasibility to rehabilitate some old quarries as UPHES.

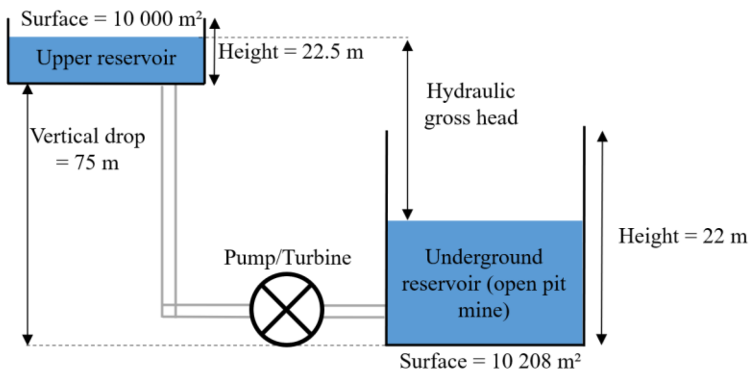


Figure 20: Schematic of the imaginary UPHES plant in Maizeret [47].

4.2 Optimisation Model Improvements and Discussions

Before including any machine learning techniques in the starting optimization model which was presented in 2.1.3 Methodology, the influence of some important parameters such as the number of head and power subintervals is assessed. We also looked into the impact of the four different methods to approximate the bounds of the UPCs: the box (or rectangle), the stepwise, the piecewise linear and the conservative piecewise linear approximations (see Figure 11). The model has been implemented using the language Julia JuMP and Gurobi as solver.

In this section, the two UPCs displayed in Figure 7 are approximated by some planes with a 1D sloped as shown in Figure 10. The accuracy of the bound and the UPC approximations are linked because there is one plane fitting the UPC for each subinterval of head and power. Therefore, for H_{sub} head subintervals and P_{sub} power subintervals, there are $H_{sub} \cdot P_{sub}$ planes (see Figure 9).

For all the models, the horizon of the simulation is one day with a time step of one hour. At the beginning of the day, both basins are considered to be filled with 112,500 m^3 of water (such that half of the energy is readily available at the start of the scheduling horizon). In order to ensure the good operation of the unit for the following day, the volume of water available in the upper reservoir at the end of the simulation is set at 80,000 m^3. The optimization yields, for every time step, the participation into both electricity markets and the operating point for the hydraulic machine.

4.2.1 Example

In Table 1 is displayed an example of the results output by the solver for a piecewise plane approximation of the UPCs and piecewise linear approximation of the bounds (without safety margin) for 3 subintervals of head and 1 subinterval of power. In the second part of the table, one can see that the expected profit at the end of the day is about 1930 € among which 750 € come from the participation to the reserves.

The first part of the table contains the hourly decisions. *T* is the time step columns. *Price DAM* stands for the hourly price on the day-ahead market while *En. DAM* is the energy bid by the UPHES owner on the DAM. It is negative if the plant owner buys energy, and positive if the plant generates. *P turb.* And *P pump* are the power in turbine mode and pump mode, respectively. *Q turb.* and *Q pump* are the equivalent but for the water flow rate. Lasty, *H low*, *H up* and *H net* are the head of the lower and the upper basins and the net head, respectively.

When surveying the hourly decisions, one can notice how the unit tends to pump when the price of electricity is low and to turbine when it is high. Each of these power positions is associated to a water flow rate being transferred from one basin to the other. This leads to an update in the head levels which define the range of power accessible for the current time step. Obviously, the unit is either pumping or turbining. Therefore, when one mode is in operation, the other is off.

Table 1: Results output by the optimisation with the piecewise plane approximation of the UPC and piecewise linear approximation of the bounds (without safety margin) for 3 subintervals of head and 1 subinterval of power: (a) the DAM bid and the operating point for each hour; (b) the ancillary market participation (i.e., reserve participation) and the financial expected results.

T [h]	Price DAM [€/MWh]	En. DAM [MWh]	P turb. [MW]	P pump [MW]	Q turb. [m^3/s]	Q pump [m^3/s]	H low [m]	H up [m]	H net [m]
0	68.3	3.61	3.61	0.00	5.25	0.00	12.87	9.36	71.5
1	63.7	3.40	3.40	0.00	4.94	0.00	14.61	7.58	68.0
2	58.1	-5.59	0.00	5.59	0.00	8.11	11.75	10.50	73.7
3	56.0	-6.49	0.00	6.49	0.00	9.53	8.39	13.93	80.5
4	55.9	-7.40	0.00	7.40	0.00	8.85	5.27	17.12	86.8
5	60.8	-8.47	0.00	8.47	0.00	10.24	1.66	20.80	94.1
6	70.5	4.75	4.75	0.00	5.69	0.00	3.67	18.76	90.1
7	90.9	4.50	4.50	0.00	5.40	0.00	5.57	16.81	86.2
8	93.0	5.36	5.36	0.00	6.42	0.00	7.84	14.50	81.7
9	93.5	3.97	3.97	0.00	5.77	0.00	9.87	12.43	77.6
10	101.9	4.88	4.88	0.00	7.09	0.00	12.37	9.87	72.5
11	98.7	3.46	3.46	0.00	5.03	0.00	14.14	8.06	68.9
12	89.5	-5.74	0.00	5.74	0.00	8.35	11.20	11.07	74.9
13	86.6	-6.49	0.00	6.49	0.00	9.53	7.84	14.50	81.7

14	90.6	-7.59	0.00	7.59	0.00	9.10	4.63	17.78	88.1
15	88.5	4.39	4.39	0.00	5.25	0.00	6.48	15.89	84.4
16	87.6	-8.06	0.00	8.06	0.00	9.70	3.06	19.38	91.3
17	97.5	4.91	4.91	0.00	5.88	0.00	5.13	17.26	87.1
18	112.0	6.40	6.40	0.00	7.66	0.00	7.84	14.50	81.7
19	112.4	5.66	5.66	0.00	8.23	0.00	10.74	11.54	75.8
20	98.1	4.39	4.39	0.00	6.39	0.00	12.99	9.24	71.2
21	86.5	3.39	3.39	0.00	4.92	0.00	14.73	7.47	67.7
22	83.5	-5.55	0.00	5.55	0.00	8.06	11.89	10.37	73.5
23	79.3	4.52	4.52	0.00	6.57	0.00	14.20	8.00	68.8

	Price [€/MW]	**Capacity [MW]**
Upward aFRR:	10	0
Upward aFRR:	20	0.68
Upward mFRR:	5	0
Downward FCR:	10	0
Downward aFRR:	20	0.88
Downward mFRR:	5	0
Profit for availability (capacity) of upward reserves [€]:	326.83	
Profit for availability (capacity) of downward reserves [€]:	424.25	
Profit realized with arbitrage in the day-ahead market [€]:	1668.98	
Operational costs for utilization of the PSH stations [€]:	-490.15	
Total profit (objective value) [€]:	1929.91	

The optimization algorithm expects a profit of roughly 1930 € at the end of the day if the unit owner follows its decisions. However, those decisions are based on two approximations, one on the UPCs and one on the operating bounds of these UPCs. What is the feasibility of those decisions? To answer this question, an ex-post profit (i.e., profit which would be actually obtained by the owner) is computed using a simulator.

4.2.2 Simulator

For each set of parameters fed to the optimization problem, the optimization solver outputs the best solution (if one exists). However, this solution is based on some approximations (approximation on the UPCs and on theirs bounds) which introduce some errors. Those errors can mislead the PHES operator in its decisions as he will not be able to implement them in real-life. Therefore, he exposes himself to some penalties from the market because he cannot comply with his commitments. The deviation from some goals must also be penalized.

The simulator is not an optimization problem. It takes the decision output from the optimization problem and ensure their feasibility by checking the real values on the actual UPCs and the real bounds rather than some approximation. It requires far less resources than an optimization process, which allows to reduce drastically the time granularity of the procedure. Whereas the optimization time step is one hour, the simulator time step is set to 1 minute. Therefore, for every minute of the time horizon, the simulator updates the head and ensures the decisions taken by the optimization solver are feasible. Despite this thinner granularity, the simulator requires less than 30 seconds to output the results.

The simulator we have implemented using Julia is initialized at the same values of heads and water volumes as the optimization. The reserve participation is retrieved once at the beginning. With the exact net head and the reserve participation, the real range of power available can be found. In this range of power, it selects the power operating point available in the UPC databases the closest to the optimization power operating point. Having a power and a net head, the simulator gets the actual water flow rate from the UPC. If the volumes constraints are not met at the end of the time step due to an unrealistic water flow rate, the simulator finds the closest water flow rate which fulfill the constraints. Finally, it updates the volumes and the heads, and starts to compute a new time step. The proceeding either for the turbine or the pump mode is summarized in Figure 21.

In the simulator, we penalize three types of constraint disrespects. Firstly, the simulator ensures the operator can deliver the downward and the upward reserve at any moment. If it cannot meet this requirement, every MWh overcommitted is penalized at 500 €/MWh. Typically, this situation occurs for an outer approximation of power bounds, i.e., when the bounds on the power are approximated larger than what they really are. Therefore, the optimization problem considers a wider range of power than the real one and may overbook its reserve capacities as shown by Figure 22.

It is worth considering which power decision the simulator should take when a reserve overbooking occurs. Any power decision taken within the "overcommitment" range minimize the amount of reserve commitment which is not fulfilled. Therefore, the operator's most strategic choice is to take, within that range, the power decision the closest to its DAM bid which is what the simulator does.

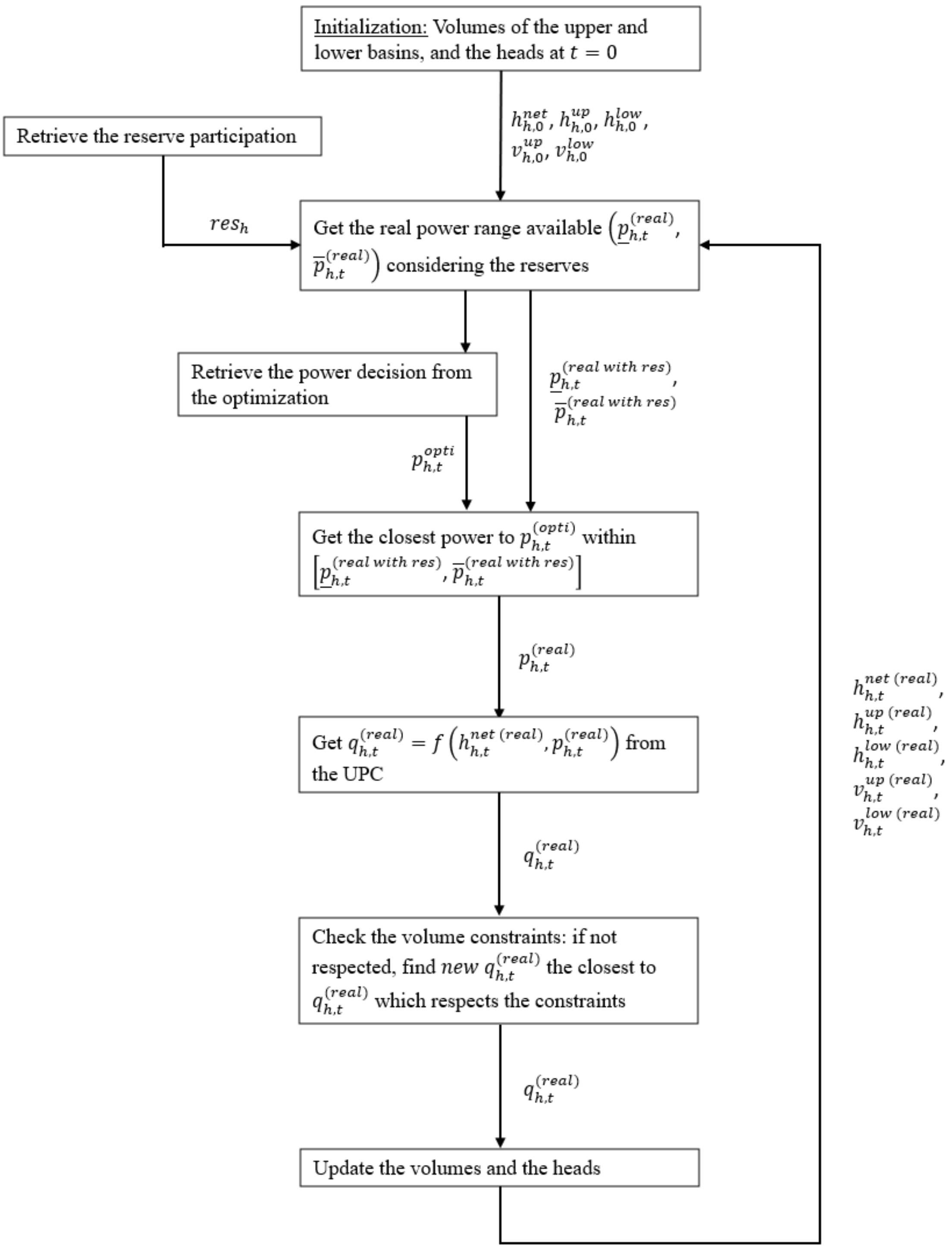
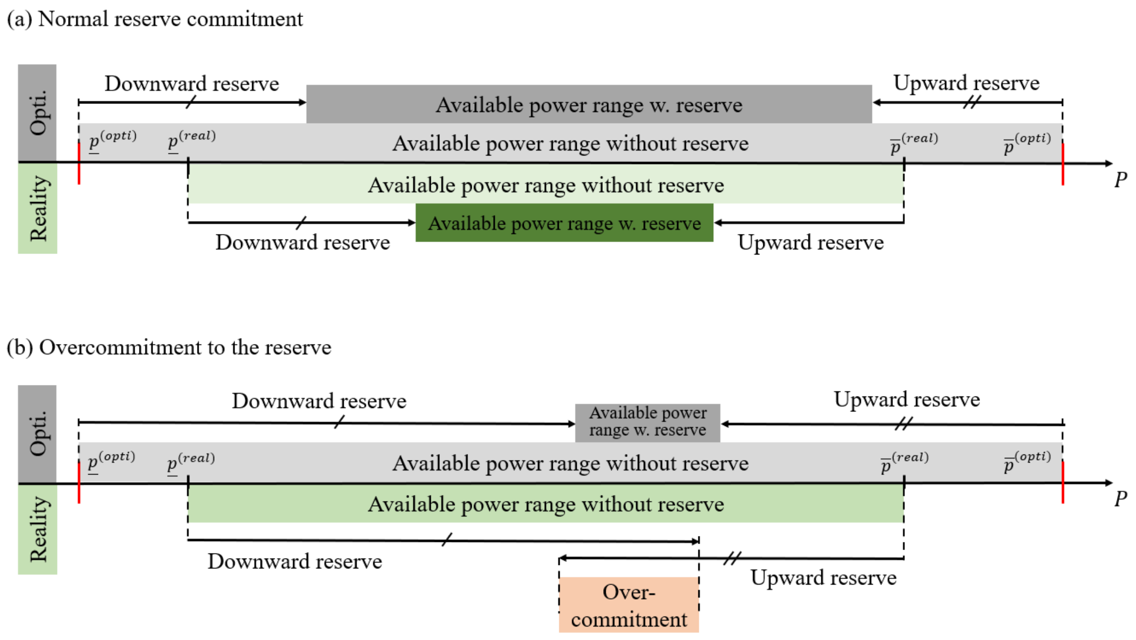

Initialization: Volumes of the upper and lower basins, and the heads at $t = 0$
$h_{h,0}^{net}, h_{h,0}^{up}, h_{h,0}^{low}, v_{h,0}^{up}, v_{h,0}^{low}$
Retrieve the reserve participation
res_h
Get the real power range available $\left(\underline{p}_{h,t}^{(real)}, \overline{p}_{h,t}^{(real)}\right)$ considering the reserves
Retrieve the power decision from the optimization
$\underline{p}_{h,t}^{(real\ with\ res)}, \overline{p}_{h,t}^{(real\ with\ res)}$
$p_{h,t}^{opti}$
Get the closest power to $p_{h,t}^{(opti)}$ within $\left[\underline{p}_{h,t}^{(real\ with\ res)}, \overline{p}_{h,t}^{(real\ with\ res)}\right]$
$p_{h,t}^{(real)}$
Get $q_{h,t}^{(real)} = f\left(h_{h,t}^{net\ (real)}, p_{h,t}^{(real)}\right)$ from the UPC
$h_{h,t}^{net\ (real)}, h_{h,t}^{up\ (real)}, h_{h,t}^{low\ (real)}, v_{h,t}^{up\ (real)}, v_{h,t}^{low\ (real)}$
$q_{h,t}^{(real)}$
Check the volume constraints: if not respected, find $new\ q_{h,t}^{(real)}$ the closest to $q_{h,t}^{(real)}$ which respects the constraints
$q_{h,t}^{(real)}$
Update the volumes and the heads

Figure 21: Algorithm of the simulator.



(a) Normal reserve commitment
Opti.
Reality
Downward reserve
Available power range w. reserve
Upward reserve
$\underline{p}^{(opti)}$
$\underline{p}^{(real)}$
Available power range without reserve
$\overline{p}^{(real)}$
$\overline{p}^{(opti)}$
P
Available power range without reserve
Available power range w. reserve
Downward reserve
Upward reserve
(b) Overcommitment to the reserve
Downward reserve
Available power range w. reserve
Upward reserve
Available power range without reserve
Available power range without reserve
Downward reserve
Upward reserve
Over-commitment

Figure 22: The figure displays how approximating the power bounds loosely and overestimating the power range can easily lead to a reserve overcommitment in real-life (situation (b)) which is heavily penalized.

Secondly, if the actor cannot fulfill his commitment on the day-ahead market (DAM), we consider he has to buy the electricity on the imbalance settlement at a price of 250 €/MWh, which is a realistic imbalance price for Belgium at the current period [48]. This price is assumed constant over time. It must be paid by the operator for every MWh that cannot be delivered (in turbine mode) and every extra MWh consumed (in pump mode). On the opposite, if the operator generates more power than he bided (turbine mode), no money is earned for the extra production. If less power is consumed by the pump than the bid on the DAM, the full amount of electricity bid must still be paid.

Thirdly, if the amount of water in the upper reservoir is not above the set threshold (80,000 m²), every MWh of energy not available to be turbined the following day costs 100 €/MWh. The conversion between water volume and MWh available is done using Equation (49) which is based on Equation (1b) for computing the gravitational potential energy assuming a turbine efficiency η of 90%.

$$E_{loss} = \eta \cdot \rho \cdot g \cdot V \cdot h \tag{49}$$

To sum up, we have carefully designed the simulator to mimic the behavior of an UPHES plant operator in real-life. Due to the approximations in the optimization process, this behavior deviates from the optimization decisions. The penalties incurred to the operator because of these necessary deviations are computed and recorded.

4.2.3 Example (following)

Thanks to the simulator, one can deepen the analysis of the example which started in section 4.2.1 Example. The power bounds (and the corresponding water flow rate bounds) can be found back. Figure 23 illustrates the evolution of the net head, the power and the water flow rate following the optimization decisions as well as the actual power and water flow feasible intervals. It stands out clearly that, when the participation to the reserves is considered (i.e., dark grey interval), there is not always a feasible range of power for the pump mode. However, for the turbine mode, the situation never occurs. This is an expected result as for a given net head, the power range available for the pump mode is always smaller than the one for the turbine mode (see Figure 8).

A similar figure can be computed by plotting the decisions corrected by the simulator instead of the ones from the optimization. The results are displayed by Figure 24. One should notice how the simulator adapts the operating point of the turbine to stay within the actual available power range considering the reserve but always as close as possible from the decision of the optimization. Another remarkable feature is the net head difference at the end of the day between the optimization and the simulator. Despite a similar starting level, there is a difference of roughly 5 m. Figure 25 teaches us how the errors over the net head estimation accumulate over the day.

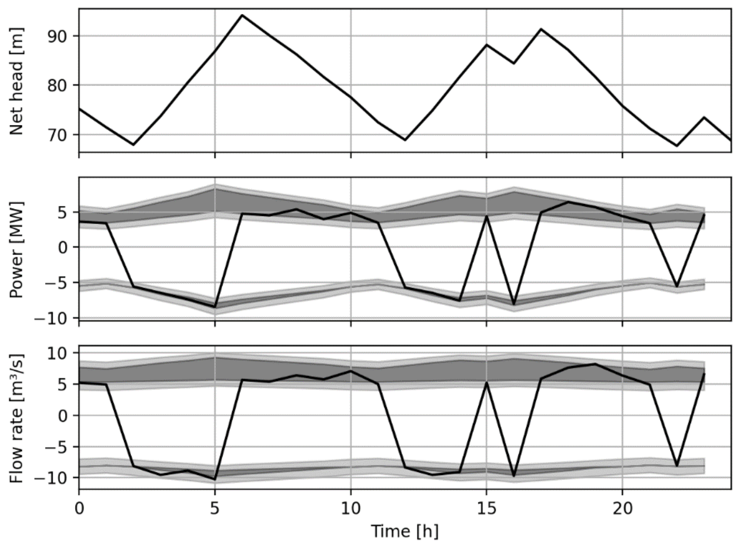


Figure 23: Plot of the net head, the power and the water flow rate evolution over one day following the *optimization decisions*. The actual power and water flow feasible intervals are represented, in light grey and darker grey whether the reserves are considered, or not.

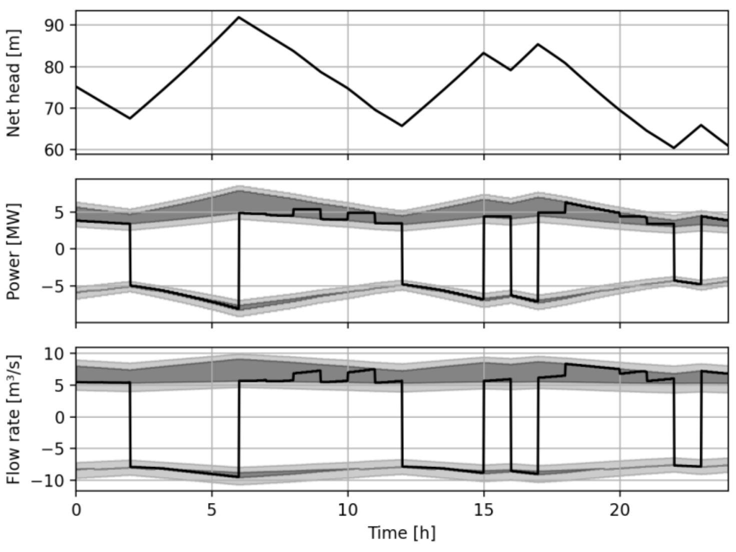


Figure 24: Plot of the net head, the power and the water flow rate evolution over one day following the *simulator decisions*. The actual power and water flow feasible intervals are represented, in light grey and darker grey whether the reserves are considered, or not.

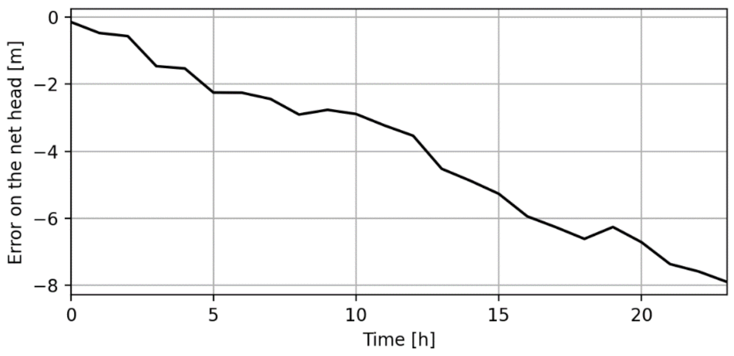


Figure 25: Evolution of the difference between the net head approximated by the optimization and the actual one (i.e., the net head error) for the example.

As a result of the simulation, the ex-post profit of the example can be obtained along with the underpinning penalties. In the case of this example, the three types of penalty are present (Table 2). There has been an overbooking of the reserve leading to a penalty of about 350 €. This is caused solely by a loose, i.e., non-conservative, approximation of the UPC bounds. This overbooking likely triggered an incapacity to fulfill the DAM bid due to very restraint power ranges available, resulting in almost 350 additional euros of penalty. Finally, 880 € of penalty are incurred due to a lack of water in the upper reservoir at the end of the day which represents 40 000 m^3 of water. Those are the consequence of a bad approximation of the UPCs and its bounds.

Table 2: Profits and penalties yielded by the simulator for the study example.

Profit energy sold [€]:	1669.0
Penalty on the reserve [€]:	349.2
Profit reserve [€]:	751.1
Settlement penalties [€]:	344.0
Operational cost [€]:	456.8
Volume left cost [€]:	-880.5
Ex-post profit [€]:	389.6

This ex-post profit of 389.6 € is the real profit the operator of the plant would have obtained if it had taken part into the DAM and the reserve market according to the optimization decisions and adjusted, in real-time, its operating point as done by the simulator. It is important to note that the operator was expecting a profit of 1930 € by following the optimization decisions. **This clearly demonstrates (i) the need to evaluate the performance of an optimization problem using the ex-post profit (i.e., the profit yielded by the simulator) and not the expected profit (i.e., the profit yielded by the optimization solver), and (ii) the need to improve the mathematical representation of the UPHES physical constraints.**

4.2.4 Parametric analysis

For this first sensitivity study, we realised a parametric sweep over the number of subintervals for the head and the power. Both those numbers have been assigned the same integer value ranging from one to seven. Each of these seven cases have been run for the four bounds approximations presented in Technical constraints and displayed by Figure 11. The results are presented in Table 3.

Table 3: Sensitivity study of the UPC piecewise approximation depending on the number of subintervals and the UPC bound approximation method.

n° subintervals (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q[4] [m^3/s]	Ex-post [€]
1 x 1	rectangle	822	1.86	2763.5	1.97	-24158.7
	stepwise	822	0.01	0.0	0.00	0.1
	piecewise	822	0.93	1630.3	0.66	-626.3
	cons. pcw	822	1.51	1228.4	0.28	681.7
2 x 2	rectangle	1158	181	2405.3	1.72	-15024.4

[4] MAE stands for Mean Absolute Error. The MAE on Q is defined as the mean over the errors between the flow rate estimated by the optimization and the real one applied by the simulator.

	stepwise	1158	0.04	360.7	0.04	360.8
	piecewise	1158	64.6	1738.9	0.56	980.1
	cons. pcw	1158	18.3	1599.6	0.27	1586.5
3 x 3	rectangle	1686	801	2530.4	1.08	-16777.1
	stepwise	1686	0.12	559.5	0.01	559.6
	piecewise	1686	369	1930.6	0.53	389.7
	cons. pcw	1686	228	1763.0	0.35	1246.1
4 x 4	rectangle	2406	7143	2489.5	0.89	-20679.2
	stepwise	2406	24.8	1571.1	0.19	1573.7
	piecewise	2406	2922	1787.9	0.33	1619.9
	cons. pcw	2406	2001	1668.1	0.30	1667.0
5 x 5	rectangle	3318	6063	2560.5	1.70	-18811.7
	stepwise	3318	711	1567.7	0.14	1567.5
	piecewise	3318	7460	1760.1	0.31	1530.9
	cons. pcw	3318	7129	1656.2	0.25	1451.2
6 x 6	rectangle	4422	11828	2462.1	1.06	-15935.6
	stepwise	4422	2980	1530.3	0.10	1529.2
	piecewise	4422	70533	1799.1	0.43	721.7
	cons. pcw	4422	18855	1679.3	0.28	1308.3
7 x 7	rectangle	5718	147242	2486.2	1.29	-15475.4
	stepwise	5718	10729	1631.5	0.11	1630.9
	piecewise	5718	197004	1728.8	0.28	1518.0
	cons. pcw	5718	184400	1648.3	0.20	1649.4

Firstly, the number of variables grows quickly with the number of subintervals going from 822 variables for one subinterval of head and power to 5718 for seven subintervals of each. The solving time follows naturally the same trend going from an average of 1 s to more than 37 h for one subinterval to seven, respectively. This shows an exponential evolution as clearly visible on Figure 26. Furthermore, **the stepwise approximation is by far the least time-consuming** while the normal and conservative piecewise approximations tend to be the most time-consuming methods. At this stage, it should be noted that **a computation time of maximum 1 hours (or 3600s) is deemed as reasonable for the day-ahead scheduling problem.**

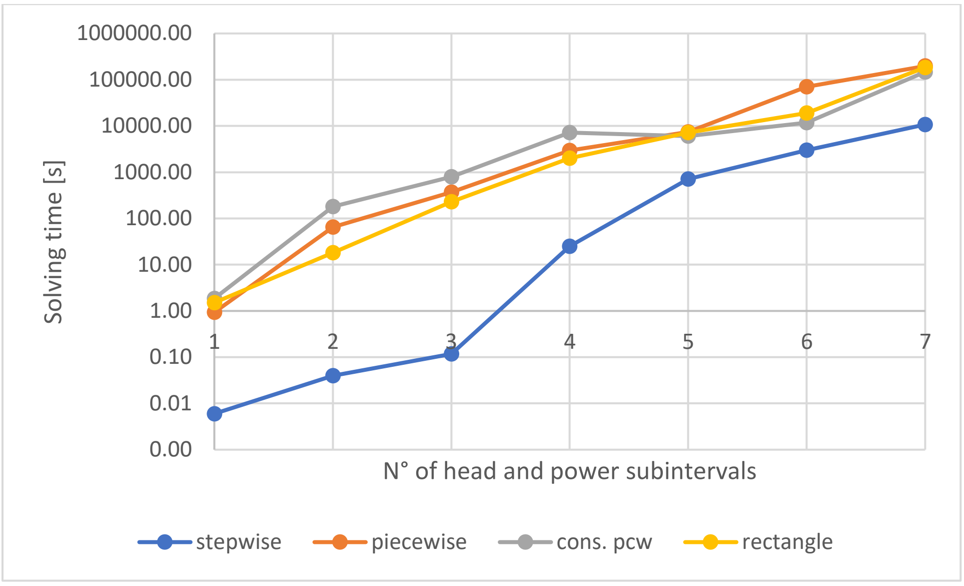


Figure 26: Evolution of the solving time depending on the number of subintervals and the UPC bound approximation method.

When surveying the expected profit (at the end of the optimization), one can see that the four methods level off after four subintervals. Overall, the method promising by far the highest profits is the rectangle method followed by the normal and the conservative piecewise methods and lastly, the stepwise method (see Figure 27). The steep rise in expected profit for the stepwise method from three to four subintervals occurs because there is virtually no feasible set in the pump UPC, thus preventing any pumping.

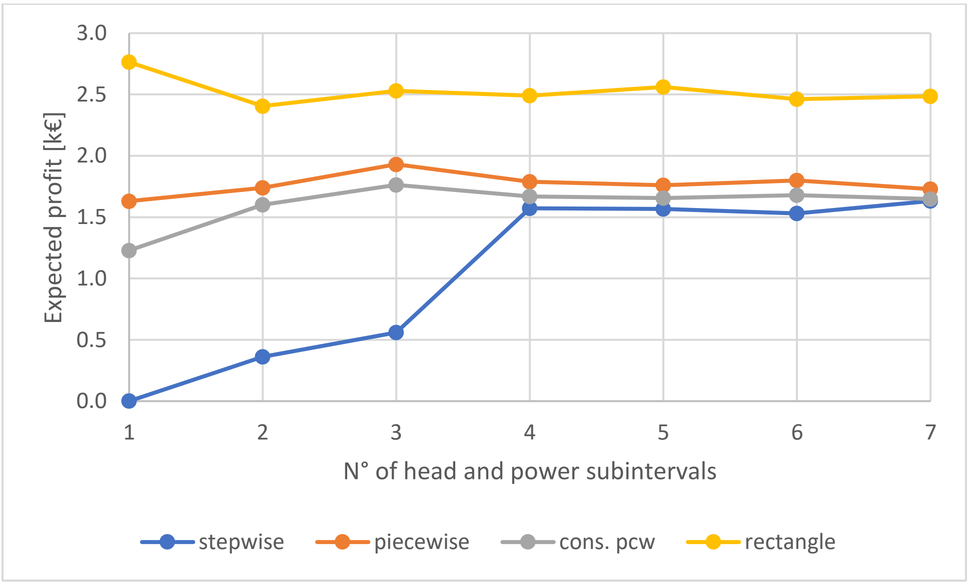


Figure 27: Evolution of the expected profit depending on the number of subintervals and the UPC bound approximation method.

Lastly, the ex-post profit can be analysed (Figure 28). **The method with the smallest ex-post profit is the rectangle method, which suffer more than 15000 € in penalties on average**. If the rectangle method was able to reach larger expected profits, it is because of the infeasible decisions it was taking. These infeasible decisions are then heavily penalized by the simulator which mimics the real-life penalties. **Despite the very poor performances shown by this technique, it is still widely used in the literature for its simplicity [49]–[51].**

On the contrary, the stepwise method with the smallest expected profit does not suffer from any major penalty, making it the best method for five and six subintervals and being therefore the most reliable one.

The piecewise linear method can sometimes be incurred high penalties due to a reserve overcommitment. This is what happens for one, two, three and six subintervals. This overcommitment usually leads to a very narrow range of power in which the unit, especially the pump mode, can operate. It prevents the owner to meet the DAM bids and to have enough water in the upper reservoir at the end of the simulation. In other words, the overcommitment on the reserve constraints heavily the available power range, triggering a disrespect of the DAM bids and the water volume in the upper reservoir at the end of the day. Altogether, they cause heavy penalties.

The conservative piecewise method is more reliable than its conventional counterpart. The only penalty from which it may sometimes suffer is a lack of water in the upper basin at the last time step due to a bad approximation of the water flow rates.

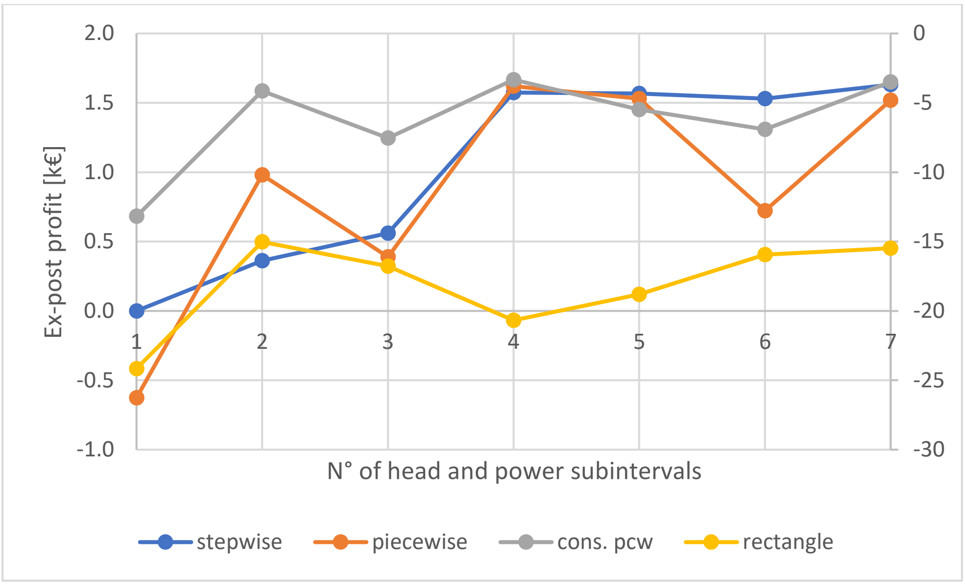


Figure 28: Evolution of the ex-post profit depending on the number of subintervals and the UPC bound approximation method (the rectangle values are on the right axis).

In the above results, the number of head subintervals is set to be equal to the number of power subintervals. However, it was noticed in Figure 10 that the power subintervals seem to have a very little impact. To demonstrate this intuition, we have decoupled the number of subintervals and set only one power subinterval for all the cases. It is important to notice that this changes only the approximation of the UPCs and not their bounds. The approximation of the bounds is linked to the head subintervals only. The results of this new sensitivity study are displayed in Table 4.

Table 4: Sensitivity study of the UPC piecewise approximation depending on the number of head subintervals (*number of power subinterval set to one*) and the UPC bound approximation method.

n° subin-tervals (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	rectangle	822	2.00	2763.5	1.97	-24158.7
	stepwise	822	0.01	0.0	0.00	0.1
	piecewise	822	1.04	1630.3	0.66	-626.3
	cons. pcw	822	1.51	1228.4	0.28	681.7
2 x 1	rectangle	966	68.1	2408.8	1.72	-15069.8
	stepwise	966	0.05	360.7	0.04	360.8
	piecewise	966	20.6	1737.1	0.56	967.2
	cons. pcw	966	18.9	1598.2	0.28	1585.9
3 x 1	rectangle	1110	187.4	2523.7	1.08	-16757.6
	stepwise	1110	0.57	1417.1	0.14	1418.8
	piecewise	1110	76.7	1929.9	0.52	389.6

	cons. pcw	1110	109.9	1761.3	0.35	1246.0
4 x 1	rectangle	1254	524.5	2494.3	0.86	-20680.2
	stepwise	1254	9.3	1576.1	0.19	1577.3
	piecewise	1254	526.9	1789.4	0.34	1612.1
	cons. pcw	1254	1331.8	1673.3	0.30	1673.0
5 x 1	rectangle	1398	889.8	2557.5	1.70	-18682.9
	stepwise	1398	133.6	1572.1	0.14	1572.4
	piecewise	1398	1791.4	1763.9	0.32	1514.8
	cons. pcw	1398	1492.5	1660.8	0.34	1217.3
6 x 1	rectangle	1542	3737.2	2455.6	1.04	-15742.6
	stepwise	1542	1114.3	1537.8	0.10	1535.4
	piecewise	1542	2929.2	1804.7	0.44	695.3
	cons. pcw	1542	1690.3	1684.9	0.30	1266.2
7 x 1	rectangle	1686	3335.5	2486.1	1.29	-15475.9
	stepwise	1686	634.7	1636.7	0.11	1636.2
	piecewise	1686	6945.5	1728.8	0.30	1431.8
	cons. pcw	1686	5317.5	1645.9	0.21	1641.5

By comparing Table 4 and Table 3, one can see that the results match quite well over-all. The average expected profit stands at 1695.3 € for Table 3 and 1728.3 € for Table 4 with an average delta of 34 €. The ex-post profit is also slightly higher in the second case, going from -2954.0 € to -2926.4 €. The average delta is 61 € in this case. Surprisingly, the ex-post profit is better when there is only one subinterval of power. This is due to one single major discrepancy occurring for three subintervals of head and the stepwise approximation where the ex-post profit jumps from 559.6 € to 1418.8 €. If this discrepancy and the rectangle approximations are discarded the ex-post profits averages become 1119.3 € and 1098.9 € for Table 3 and Table 4, respectively. **Affording this ex-post profit loss of around 2% allows to divide the solving time by 20** (from an average of 6.7 h to 0.33 h) over the whole tables and by 5 (from an average of 0.49 h to 0.10 h) when stopping at five subintervals.

In the following sections, the rectangle approximation is discarded since its inability to properly model the problem has been demonstrated. Moreover, the number of head subintervals is limited to five because no important evolution is noticeable by going further and the solving time shoots up.

4.3 Linear Regression

In this section, we perform a first machine learning informed optimization by replacing the piecewise approximation of the UPCs by the least squared error planes. This method was presented in 3.1 Linear regression. The techniques to approximate the bounds do not change. The best plane obtained for the pump and the turbine UPCs, obtained with the package Scikit-learn in Python, are depicted by Figure 29. Those planes, unlike the piece-wise linear planes are not defined on intervals of head and power. They extend to the infinity. Therefore, the feasible set on the curve is bounded by the approximation method only. Figure 30 displays those observations and the error between the genuine turbine UPC and its approximation.

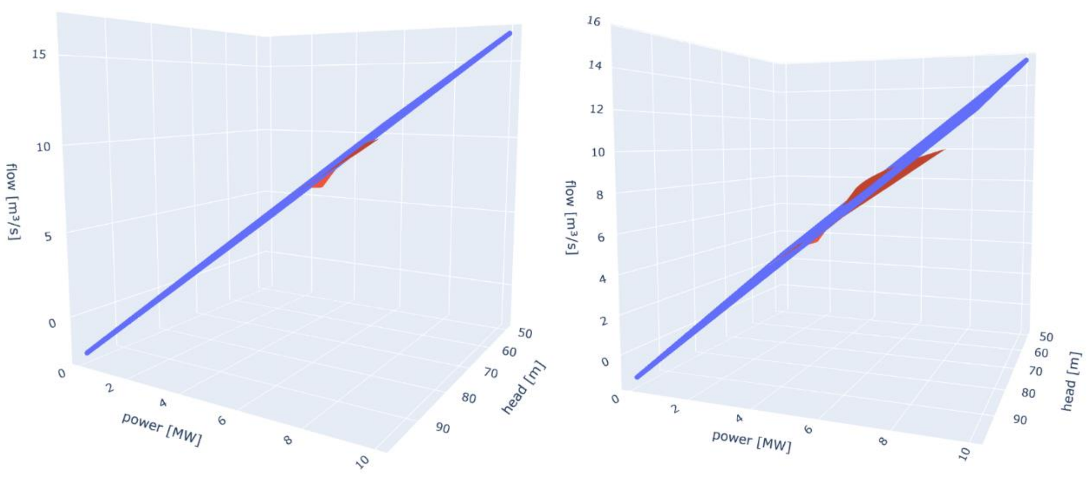


Figure 29: Least squared error planes (blue) compared to the original UPC (red) for the pump (left) and the turbine (right). (click on the figures and download for an interactive view)

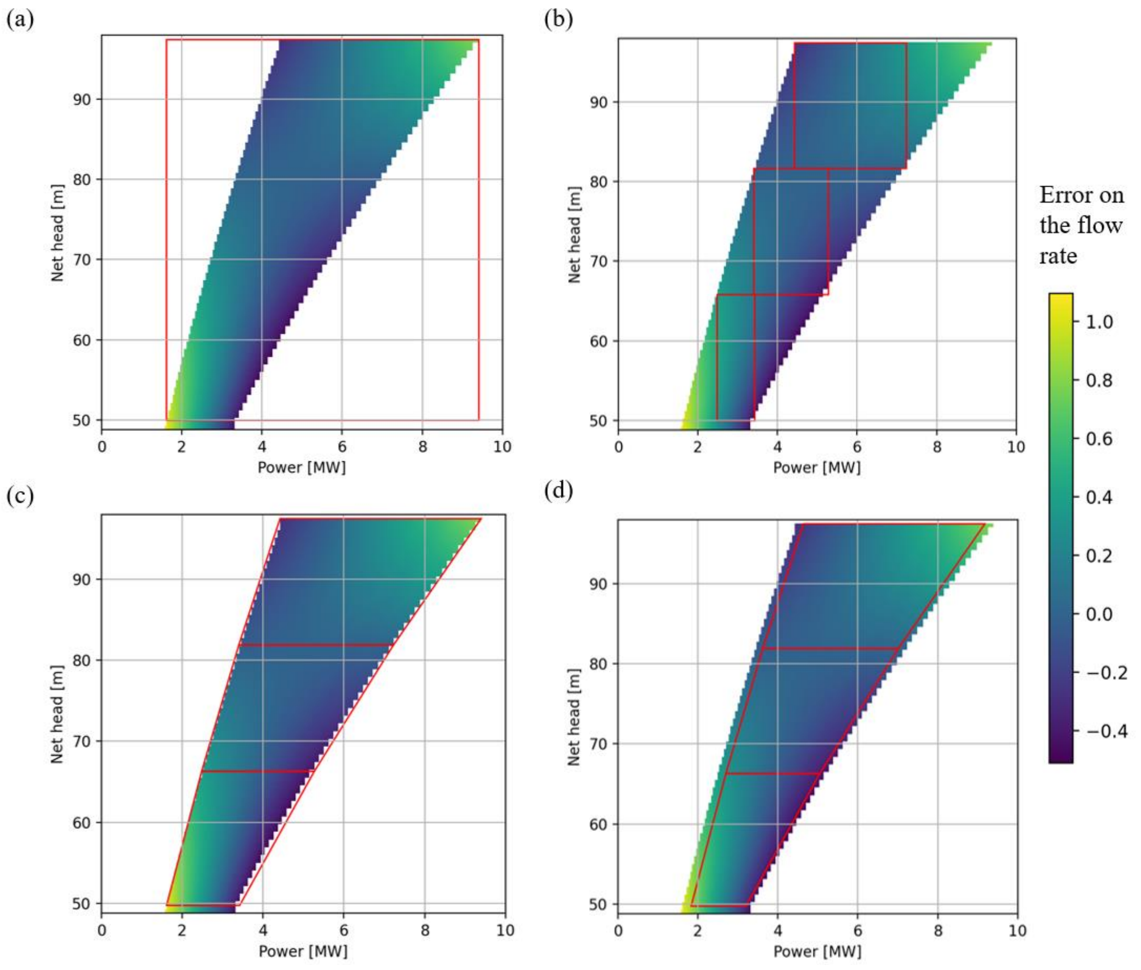


Figure 30: Illustration of the methods for approximating the bounds of the turbine UPC considering three subintervals of head; (a) is the surrounding rectangle; (b) is the stepwise approximation; (c) is the piecewise approximation; (d) is the conservative piecewise approximation. The plot of the error is based on the linear regression approximation.

The linear regression process provides the coefficients of the planes. Consequently, Equations (16) become can be replaced by Equations (50a) & (50b) and the eleven Equations (17a)-(19) can be deleted. However, Equations (50a) & (50b) cannot be integrated as such in the problem. Indeed, when the unit is in turbine mode, Constraint (50a) must be unbinding (i.e., the constraint must not be enforced) and the other way around when the unit is in pump mode, Constraint (50b) must be unbinding. To reach this goal, the Big-M relaxation [52], [53] is used, and the two constraints are replaced by Equations (51a)-(51d). Thanks to the use of the binary variables $z_{h,t}^{P}$ and $z_{h,t}^{T}$ introduced previously which indicate the mode in which the unit operates, the respect of the UPC associated to one mode is imposed if the unit operates in that mode only.

$$q_{h,t}^{P} = -0.106 \cdot h_{h,t}^{net} + 1.38 \cdot p_{h,t}^{P} + 8.05 \qquad \forall h,t \in \{H,T\} \qquad (50a)$$

$$q_{h,t}^{T} = -0.076 \cdot h_{h,t}^{net} + 1.31 \cdot p_{h,t}^{T} + 6.15 \qquad \forall h,t \in \{H,T\} \qquad (50b)$$

$$q_{h,t}^{P} \leq -0.106 \cdot h_{h,t}^{net} + 1.38 \cdot p_{h,t}^{P} + 8.05 + (1 - z_{h,t}^{P}) \cdot \overline{Q}_{h}^{P} \qquad \forall h,t \in \{H,T\} \qquad (51a)$$

$$q_{h,t}^{P} \geq -0.106 \cdot h_{h,t}^{net} + 1.38 \cdot p_{h,t}^{P} + 8.05 + (1 - z_{h,t}^{P}) \cdot \underline{Q}_{h}^{P} \qquad \forall h,t \in \{H,T\} \qquad (51b)$$

$$q_{h,t}^{T} \leq -0.076 \cdot h_{h,t}^{net} + 1.31 \cdot p_{h,t}^{T} + 6.15 + (1 - z_{h,t}^{T}) \cdot \overline{Q}_{h}^{T} \qquad \forall h,t \in \{H,T\} \qquad (51c)$$

$$q_{h,t}^{T} \geq -0.076 \cdot h_{h,t}^{net} + 1.31 \cdot p_{h,t}^{T} + 6.15 + (1 - z_{h,t}^{T}) \cdot \underline{Q}_{h}^{T} \qquad \forall h,t \in \{H,T\} \qquad (51d)$$

As visible in Table 5 summarizing the results of the sensitivity study for the linear regression approximation of the UPCs, using one single plane per UPC allows to decrease the number of variables. However, **the solving time increases from 368 s to 433 s with respect to the same cases with a piecewise UPC approximation (Table 4). The average ex-post profit increases as well, going from 1013 € to 1177 € which represents an increase by more than 16%.** Both the expected and the ex-post profits level off even quicker than before since the last significative evolution when going from three to four head subintervals is the stepwise approximation.

Table 5: Sensitivity study of the UPC linear regression approximation depending on the number of head subintervals (number of power subinterval set to one) and the UPC bound approximation method.

n° subintervals (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m³/s]	Ex-post [€]
1 x 1	stepwise	726	0.01	0.0	0.000	0.1
	piecewise	726	237	1715.1	0.253	637.1
	cons. pcw	726	11.0	1573.0	0.051	1561.9
2 x 1	stepwise	774	0.30	360.7	0.009	360.8
	piecewise	774	122	1693.9	0.206	1084.8
	cons. pcw	774	112	1582.8	0.064	1547.4
3 x 1	stepwise	822	6.80	1407.4	0.059	1409.1
	piecewise	822	589	1692.6	0.200	1150.6
	cons. pcw	822	329	1585.6	0.067	1545.8
4 x 1	stepwise	870	41.6	1502.7	0.158	1507.6
	piecewise	870	1310	1690.5	0.208	1115.0
	cons. pcw	870	428	1588.9	0.068	1541.8
5 x 1	stepwise	918	406	1513.4	0.154	1517.1
	piecewise	918	1564	1690.5	0.204	1133.1
	cons. pcw	918	1332	1590.0	0.070	1540.6

4.4 Neural Network

In this last part, we approximate the UPCs using NNs. Leveraging the modelling power of NNs allows to obtain very good fits of the curves with great ease. One independent NN is trained for each of both UPCs using Keras in Python. The complexity of the fit (and its quality) can very easily be tailored by adjusting the number of neurons and layers. It is also possible to change the activation function. Two of them are studied here, the ReLU and the Leaky ReLU. Finally, we survey the impact of the sparsity on the NN performance and time for solving the optimization.

It is important to notice that **the method developed to translate a NN into an optimization problem is not limited to the application presented and can be applied to a wide range of problems.**

The NNs used have an input layer dimension of two and an output layer dimension of one. The two input variables are the net head and power whereas the output variable is the water flow rate. The activation function of the output layer is always the identity function. In between the input and the output layers are introduced some layers with the tested activation function (ReLU or Leaky ReLU).

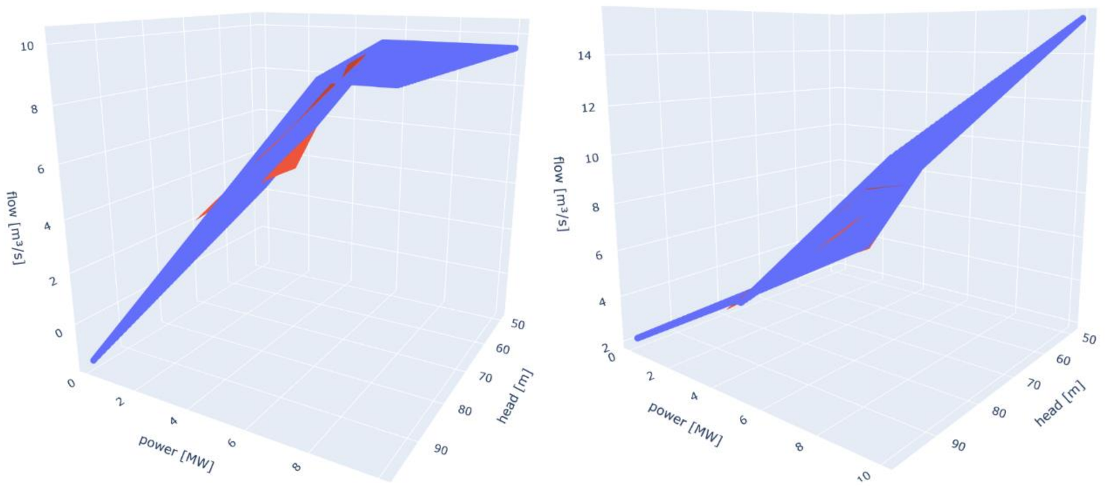


Figure 31: Neural network fit (blue) compared to the original turbine UPC (red) for one hidden layer with one neuron (left) and one hidden layer with two neurons (right) both cases with a ReLU activation function. (click on the figures for an interactive view)

Similarly to the linear regression planes, the curves equations expressed by the NN must be made unbinding when the unit operates in the other mode. Consequently, the Big-M relaxation is used on the output of both NNs (one for the turbine UPC and one for the pump UPC). However, it is not enough. Indeed, as seen in Embedding Neural Network into Optimization, in order to achieve a tight reformulation of the NN, the activation function of every neuron is translated into equations over a domain going from $\underline{\hat{y}}$ to $\overline{\hat{y}}$. Therefore, the power input variable must also be unbinding. For instance, if the unit is operating in turbine mode, the pump power variable $p_{h,t}^{P}$ is set to 0 which is not part of the training database. If $p_{h,t}^{P}$ is directly defined as the input of the NN, it triggers a risk of over constraining the problem because some intermediate variables of the NN will not respect the domain of existence $\underline{\hat{y}}$ to $\overline{\hat{y}}$ defined over the activation function. The current situation is depicted by Figure 32.

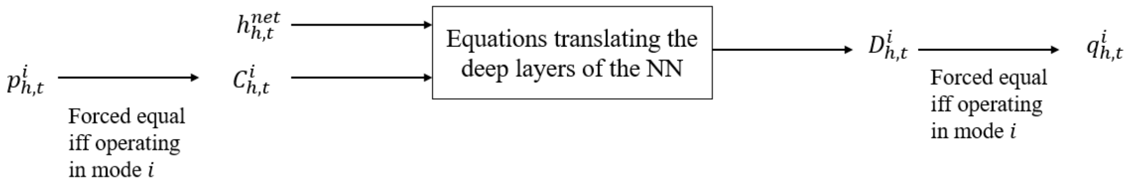

$h_{h,t}^{net}$
$p_{h,t}^{i}$
$C_{h,t}^{i}$
Forced equal iff operating in mode i
Equations translating the deep layers of the NN
$D_{h,t}^{i}$
$q_{h,t}^{i}$
Forced equal iff operating in mode i

Figure 32: Intermediate variables to make the NN modelling UPC of mode M unbinding when not operating in mode M.

Both variables C_h^i and D_h^i are to be constrained using the Big-M method. For C_h^i, a first approach would be to set the bounds to $\underline{P}_h^i$ and $\overline{P}_h^i$thereby ensuring that, for any net head $h_{h,t}^{net}$, the input variable C_h^i takes a value which has been seen by the NN during training. However, there exists a way to improve the tightness by setting the bounds $\underline{C}_h^i$ and $\overline{C}_h^i$ to the maximal actual power for the smallest allowed head and the minimal actual power for the highest allowed head (see Figure 33 and Equations (52a) & (52b)), respectively. Clearly, the red lines defined a much tighter interval than the black ones. Interestingly, if there was an overlap between the range of power matching the smallest head and the one matching the highest head, then $\underline{C}_h^M$ can be set equal to $\overline{C}_h^M$ and both can be assigned any power value within the overlap. This has shown very promising reductions of the computational time. Unfortunately, it depends on the physics of the problem.

$$\underline{C}_h^M = \max_p UPC_h^M(\underline{h}_h^{net}) \qquad \forall h \in \{H\} \qquad (52a)$$

$$\overline{C}_h^M = \min_p UPC_h^M(\overline{h}_h^{net}) \qquad \forall h \in \{H\} \qquad (52b)$$

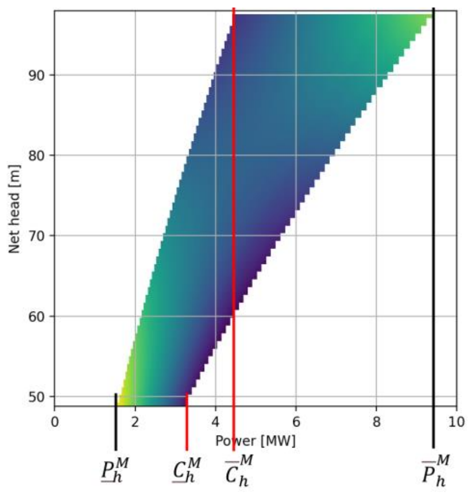


Figure 33: Smart definition of the bounds on A.

Similarly, the bounds $\underline{D}_h^M$ and $\overline{D}_h^M$ on D_h^M can be set to $\underline{Q}_h^M$ and $\overline{Q}_h^M$ but it is tighter to set them to the minimum and maximum water flow rate over the power range $\left[\underline{C}_h^M, \overline{C}_h^M\right]$ (Equations (53a) & (53b)). The final conditions on C_h^M and D_h^M are Equations (54a)-(54d).

$$\underline{D}_h^M = \min_q UPC_h^M(p) \qquad \forall h, p_h^M \in \left\{H, \left[\underline{C}_h^M, \overline{C}_h^M\right]\right\} \tag{53a}$$

$$\overline{D}_h^M = \max_q UPC_h^M(p) \qquad \forall h, p_h^M \in \left\{H, \left[\underline{C}_h^M, \overline{C}_h^M\right]\right\} \tag{53b}$$

$$C_{h,t}^M \leq p_{h,t}^M + (1 - z_{h,t}^M) \cdot \overline{C}_h^M \qquad \forall h, t \in \{H, T\} \tag{54a}$$

$$C_{h,t}^M \geq p_{h,t}^M + (1 - z_{h,t}^M) \cdot \underline{C}_h^M \qquad \forall h, t \in \{H, T\} \tag{54b}$$

$$D_{h,t}^M \leq q_{h,t}^M + (1 - z_{h,t}^M) \cdot \overline{D}_h^M \qquad \forall h, t \in \{H, T\} \tag{54c}$$

$$D_{h,t}^M \geq q_{h,t}^M + (1 - z_{h,t}^M) \cdot \underline{D}_h^M \qquad \forall h, t \in \{H, T\} \tag{54d}$$

Table 6 summarizes the results for all the NNs tested where “Average” is the average results for the three bounds approximations (stepwise, piecewise and conservative piecewise) performed over a number of head subinterval ranging from one to four, and “Best” highlight the best ex-post profit obtained. Firstly, the two ReLU reformulations are investigated to determine the quickest and ensure that the same results are obtained.

For a NN with no sparsity imposed, one deep layer made of one neuron having a ReLU activation function and using the piecewise reformulation (Formulation 1), the average ex-post profit stands at 1214 €. This is, for the similar cases, already much better than the linear approximation which stands at 1121 € (for an average solving time higher by 40%). The profits stay the same from one reformulation to the other. This is mainly due to one point, the stepwise approximation for two head subintervals.

On average, the solving time for a solver accuracy of 20% stands at 48 minutes against 78 minutes for the second reformulation (Formulation 2) despite a higher number of variables. It is reduced to 3m11s and 3m23s, respectively, for an accuracy of 50%. The results are available in Appendix,Table 7 & Table 8. Reducing the solver accuracy reduces drastically the computation time but impact only slightly the quality of the results [54]. Consequently, the solver accuracy is fixed to 50% for all the cases and the Formulation 1. Adding a second neuron to the layer increases the computation time to 2h12 but also the ex-post profits to 1249€ on average with a peak at 1624€ which is the best result obtained with the NN approximation.

If the final sparsity of the NN is imposed to 50% and the piecewise reformulation used, the solving time can be brought down to 53 seconds for one single neuron. Of course, as an extra condition is imposed on the NN, the quality of the fit is degraded. This is visible in the ex-post profit which drops to 1159 €. For the detailed results, please consult Appendix, Table 9. The average ex-post profit can be improved to 1266€ by considering two neurons instead of one at the expense of the computational time which is multiplied by nine standing at about 8 minutes (Table 11).

Finally, it seems that splitting a same number of neurons amongst more layers helps reducing the solving time (see ReLU 1 x 4 vs. ReLU 2 x 2 cases). Moreover, the quickest reformulation for one case is not the same for the other one. This means that the performances of the reformulations are case-dependent.

Changing the activation function for a Leaky ReLU allows to reach, for one layer with one neuron, an average profit of 1281€ which is better than the ReLU equivalent architecture. However, the solving time is 15 times greater. Increasing the number of neurons to two, without sparsity, leads to a solving time of almost 12h which is too much for a day-ahead scheduling of the unit. Indeed, the DAM closes at 12 am.

Imposing 50% of sparsity diminishes slightly the average ex-post profit for one neuron to 1278€ but divide the computational time by ten, bringing it slightly over five minutes. Doubling the number of neurons to two also doubles the time. This last configuration of one layer of two neurons, with a sparsity of 50%, output the best average results with an ex-post profit at 1310€.

Overall, the average ex-post profits of the NN are always higher than those of the previous approximations.

Table 6: Summary of the performances of several NN architectures compared to the piecewise linear and linear regression approximations. The data are valid for four head subintervals and three bound approximations (stepwise, piecewise and conservative piecewise).

				Average[5]		Best[6]	
	Activation function	**Sparse**	**Archit.**	**Time [s]**	**Ex-post [€]**	**Time [s]**	**Ex-post [€]**
Piecewise linear $(N = M)$[7]	Ø	Ø	Ø	469	836.6	2001	1667.0
Piecewise linear $(M = 1)$[8]	Ø	Ø	Ø	175	907.2	1673.3	1673.0
Linear regression	Ø	Ø	Ø	266	1121.8	11	1561.9
NN	ReLU (1)[9]	No	1 x 1	191	1214.0	112	1560.6
	ReLU (2)	No	1 x 1	203	1214.0	11	1560.6
	ReLU (1)	Yes	1 x 1	53	1159.1	87	1560.6
	ReLU (1)	No	1 x 2	7918	1248.9	691	1624.2
	ReLU (1)	Yes	1 x 2	366	1265.6	75	1574.3
	ReLU (1)	Yes	1 x 4	1186	1272.1	2187	1539.6
	ReLU (2)	Yes	1 x 4	803	1272.1	2960	1539.6
	ReLU (1)	Yes	2 x 2	272	1170.7	46	1555.2
	ReLU (2)	Yes	2 x 2	592	1170.7	92	1555.2
	Leaky ReLU (1)	No	1 x 1	3069	1280.7	12581	1574.0
	Leaky ReLU (1)	Yes	1 x 1	313	1278.0	1036	1570.0
	Leaky ReLU (1)	No	1 x 2	43013	1156.1	65678	1511.1
	Leaky ReLU (1)	Yes	1 x 2	668	1310.0	3429	1607.6
	Leaky ReLU (1)	Yes	2 x 2	15480	1304.1	4771	1564.9

4.5 Conclusion

The state-of-the-art approximation has been able to provide the best ex-post profit (1673€) in 22 minutes for a conservative piecewise approximation of the bounds, four head and one power subintervals.

[5] "Average" is the average performances over the 12 cases (three bounds approximation over a number of head subinterval ranging from one to four).
[6] "Best" highlights the best ex-post profit obtained
[7] The number of head subintervals N equals to the number of power subintervals M (Table 3).
[8] The number of head subintervals N varies while the number of power subintervals M is set to one (Table 4).
[9] The number in between parenthesis refers to the reformulation number.

The linear approach based on a multi linear regression has the advantage to be much easier to implement for a similar computational time. Nevertheless, the maximum ex-post profit reached falls at 1562€ in barely 11 seconds. One single plane can be so competitive due to the weak non-linearities of the UPCs.

The NN approximation has reached a maximum ex-post profit of 1624€ for computational time of 11m30s. Still, an ex-post profit of 1561€ can be achieved in 11 seconds. **The main drawback of the NN approximation is the quickly exploding solving time.** Hence, larger NNs fitting better the UPCs could not be tested. **Nevertheless, the study has clearly highlighted the interest of sparsity which significantly reduces the computational time. The ReLU has a shorter solving time than the Leaky ReLU. For a given number of neurons, splitting them between several layers of neurons seems beneficial for the solving time.**

For this study case, which is made of weakly non-linear curves, we would recommend sticking with the state-of-the-art piecewise linear method as very good results are obtained for one subinterval of power which reduces drastically the solving time. **Overall, the best ex-post profits are reached for the conservative piecewise linear approximation of the bounds** whichever UPC approximation is used (except for the Leaky ReLU NN approximations which perform better in non-conservative piecewise). The linear regression approach is simple, reliable and yields fairly good decisions very quickly for the proposed case study. Lastly, **the NNs have demonstrated the best average ex-post profits. They outperform the other method for a loose approximation of the bounds (i.e., a low number of head subintervals).** This method should be used if the feasible operating bounds of the machine or the quality of the bound approximation are not well-known.

5 Conclusion and perspectives

The scientific consensus is established, the anthropic greenhouse gas emissions must be reduced to avoid a devastator global warming. Fortunately, it is ongoing albeit not quickly enough. Renewable energies, due to their intermittency and uncertainty, have disrupted the historical functioning of the electricity network. **At the moment, the key to boost the renewable energy resource penetration and, thus, the energy transition lies in cheap, efficient and reliable energy storage systems.** These systems have the ability to bring the much-needed flexibility to the network, easing the strains created by the renewables technologies.

Among the numerous energy storage systems, only one is able to store energy at large scale, the pumped-hydro energy storage. PHES units can be installed where two basins can be fitted at different heights. It has not always to be on the ground level. Indeed, one basin can be at the ground level while the second one, the lower basin, is at the bottom of an ancient quarry or mine. This massively reduces the usually important CAPEX of the PHES and rehabilitates these abandoned sites. Regions with an important coal extraction history are very propitious which is the case of, for instance, Wallonia.

Despite their key role to play to achieve sustainability, PHES are very difficult to model because of the complexity of their efficiency curves. These UPCs are three-dimensional, non-linear, neither convex, nor concave curves which makes them impossible to include into a convex optimization problem. Complex methods to approximate the curve have been developed but the current state-of-the-art, when confronted to a realistic simulator, shows poor results as demonstrated in this work. This prevents the operators of the units to take effective decisions on the energy markets, and thus, to support properly the network due to the high penalty risk.

In this thesis, for the first time to the author’s knowledge, a neural network implemented as a regressor has been embedded to perform a machine learning informed optimization. This brand-new approach has been applied over the UPCs of an imaginary underground PHES unit based on the site of a former Belgian quarry in Maizeret. Those curves are weakly non-linear which hinders the benefits of the NN approximation. However, the feasibility of the method for obtaining satisfying results has been demonstrated.

For the first time too, the impacts of the sparsity and of the activation function of the NN over the performances of the final optimization problem are studied. **The results demonstrate that sparsity quicken the resolution of the optimization while the ReLU activation function, unlike the Leaky ReLU, allows to keep the computation time under control.**

The approximation of the UPC by a piecewise linear, a linear regression, and a NN method has been thoroughly investigated and the impact of the UPC bounds approximation has been studied. **Clearly, the current state-of-the-art which relies mainly on the box bound approximation is catastrophic.** The stepwise and piecewise bound approximations, which exist also in the literature perform better. However, **the best ex-post profits have been achieved for a conservative piecewise approximation developed in this work** (except for the Leaky ReLU NN approximations which perform better in non-conservative piecewise). Nonetheless, the stepwise approximation is associated to the lowest computation times.

Overall, thanks to the review performed and the new methods employed, the operators of a PHES unit can now confidently commit their units in a profitable way and thereby bringing the crucial flexibility to support the electricity network. This will help increasing the penetration of renewable resources (by better mitigating its intrinsic uncertainty) and make the energy sector more sustainable.

Many future works can be undertaken based on this master thesis. Firstly, there are other types of turbines with UPCs presenting higher non-linearities. For those cases, the NN approximation will likely be more advantageous. Next, other reformulation of the ReLU and Leaky ReLU functions exist in the literature, which can be investigated. They could allow to reduce the computation time, the main hurdle of the NN approximation. Afterwards, NNs could be used to model the bounds of the UPCs instead of the current approximations. Moreover, the basins should be chosen with a more realistic shape and thus, more complex. Once again, the resulting non-linear relationship between the head and the volume can be fitted with the help of a NN. Trying to use other activation functions and architectures might also bring some improvements. Lastly, adapting the method to leverage its power within a stochastic optimization problem should likely be enriching.

Bibliography

[1] A. Helseth, S. Jaehnert, and A. L. Diniz, ‘Convex Relaxations of the Short-Term Hydrothermal Scheduling Problem’, *TPWRS*, vol. 36, no. 4, pp. 3293–3304, 2021, doi: 10.1109/TPWRS.2020.3047346.

[2] A. Hamann, G. Hug, and S. Rosinski, ‘Real-Time Optimization of the Mid-Columbia Hydropower System’, *TPWRS*, vol. 32, no. 1, pp. 157–165, 2017, doi: 10.1109/TPWRS.2016.2550490.

[3] Cheng-Hung Chen, Nanming Chen, and P. B. Luh, ‘Head Dependence of Pump-Storage-Unit Model Applied to Generation Scheduling’, *TPWRS*, vol. 32, no. 4, pp. 2869–2877, 2017, doi: 10.1109/TPWRS.2016.2629093.

[4] S. Wang, J. Liu, H. Chen, R. Bo, and Y. Chen, ‘Modeling State Transition and Head-Dependent Efficiency Curve for Pumped Storage Hydro in Look-Ahead Dispatch’, *TPWRS*, vol. 36, no. 6, pp. 5396–5407, 2021, doi: 10.1109/TPWRS.2021.3084909.

[5] J.-F. Toubeau, Z. De Greve, P. Goderniaux, F. Vallee, and K. Bruninx, ‘Chance-Constrained Scheduling of Underground Pumped Hydro Energy Storage in Presence of Model Uncertainties’, *TSTE*, vol. 11, no. 3, pp. 1516–1527, 2020, doi: 10.1109/TSTE.2019.2929687.

[6] R. Lindsey, ‘Climate Change: Atmospheric Carbon Dioxide’, *National Oceanic and Atmospheric Administration Climate.gov*, Aug. 14, 202AD.

[7] V. Masson-Delmotte, P. Zhai, A. Pirani, and S. L. Connors, ‘IPCC, 2021: Summary for Policymakers. . In: Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change’, Cambridge University Press, 2021.

[8] ‘What is the Kyoto Protocol? | UNFCCC’. https://unfccc.int/kyoto_protocol (accessed Mar. 14, 2022).

[9] ‘Greenhouse gas emissions by country and sector (infographic) | News | European Parliament’, Mar. 07, 2018. https://www.europarl.europa.eu/news/en/headlines/society/20180301STO98928/greenhouse-gas-emissions-by-country-and-sector-infographic (accessed Jun. 07, 2022).

[10] H. Ritchie and M. Roser, ‘Energy’, *Our World in Data*, Nov. 2020, Accessed: Mar. 14, 2022. [Online]. Available: https://ourworldindata.org/renewable-energy

[11] ‘Renewable Energy Statistics 2021’, IRENA, 2021.

[12] J. Bottieau, ‘Machine Learning for Risk-Aware Decision-Making in Short-Term Electricity Markets’, UMons, 2022.

[13] F. Vallée, ‘Techno-economic analysis of power systems’, presented at the Course 1, Mons, Belgium, Feb. 03, 2021.

[14] ‘The current electricity market design in Europe’. KU Leuven Energy Institute, Jan. 2015. Accessed: Apr. 14, 2022. [Online]. Available: https://set.kuleuven.be/ei/factsheets

[15] ‘EPEX SPOT – Europex’. https://www.europex.org/members/epex-spot/ (accessed Jun. 16, 2022).

[16] S. Rehman, L. M. Al-Hadhrami, and Md. M. Alam, ‘Pumped hydro energy storage system: A technological review’, *RENEW SUST ENERG REV*, vol. 44, pp. 586–598, 2015, doi: 10.1016/j.rser.2014.12.040.

[17] ‘How Pumped Storage Hydropower Works’, *Energy.gov*. https://www.energy.gov/eere/water/how-pumped-storage-hydropower-works (accessed Mar. 10, 2022).

[18] J. P. Deane, B. P. Ó Gallachóir, and E. J. McKeogh, ‘Techno-economic review of existing and new pumped hydro energy storage plant’, *Renewable and Sustainable Energy Reviews*, vol. 14, no. 4, pp. 1293–1302, May 2010, doi: 10.1016/j.rser.2009.11.015.

[19] ‘Centrale d’accumulation par pompage de Coo’, *ENGIE*. https://corporate.engie.be/fr/energy/hydraulique/centrale-daccumulation-par-pompage-de-coo (accessed Mar. 11, 2022).

[20] J. Vanvilthoven, ‘Introducing Reinforcement Learning for the day-ahead scheduling of pump-hydro stations’, UMons, 2020.

[21] G. Coussement, ‘Méchanique des Fluides’, presented at the Lecture, FPMs - UMons, Fall 2019.

[22] G. Ardizzon, G. Cavazzini, and G. Pavesi, ‘A new generation of small hydro and pumped-hydro power plants: Advances and future challenges’, *RENEW SUST ENERG REV*, vol. 31, pp. 746–761, 2014, doi: 10.1016/j.rser.2013.12.043.

[23] B. Tong, Q. Zhai, and X. Guan, ‘An MILP Based Formulation for Short-Term Hydro Generation Scheduling With Analysis of the Linearization Effects on Solution Feasibility’, *TPWRS,* vol. 28, no. 4, pp. 3588–3599, 2013, doi: 10.1109/TPWRS.2013.2274286.

[24] A. Vandaele, ‘Analyse numérique : Approximation et polynômes orthogonaux’, presented at the Séance 4, Mons, Belgium, 2021.

[25] D. J. Olive, *Linear Regression by David J. Olive.*, 1st ed. 2017. Cham : Springer International Publishing : Imprint: Springer, 2017.

[26] ‘What is machine learning?’, *Google Cloud*. https://cloud.google.com/learn/what-is-machine-learning (accessed May 07, 2022).

[27] L. Sutton, ‘Introduction To Neural Networks’, presented at the Satellite Image Classification & Change Detection, Portland State University, 2012.

[28] K. Gurney, ‘Introduction to Neural Networks’. Taylor & Francis, Oxford.

[29] A. Blais and D. Mertz, ‘An introduction to neural networks’, *IBM Developer*, Aug. 20, 2018. https://developer.ibm.com/articles/l-neural/ (accessed May 05, 2022).

[30] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015. Accessed: May 06, 2022. [Online]. Available: http://neuralnetworksanddeeplearning.com

[31] M. Mehrzadi, Y. Terriche, C.-L. Su, M. Bin, J. C. Vasquez, and J. Guerrero, ‘Review of Dynamic Positioning Control in Maritime Microgrid Systems’, Jun. 2020.

[32] ‘What is Deep Learning?’, Mar. 30, 2022. https://www.ibm.com/cloud/learn/deep-learning (accessed May 05, 2022).

[33] ‘What are Neural Networks?’, *IBM*, Aug. 03, 2021. https://www.ibm.com/cloud/learn/neural-networks (accessed May 05, 2022).

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. Accessed: May 06, 2022. [Online]. Available: https://www.deeplearningbook.org/

[35] L. Timpl, R. Entezari, H. Sedghi, B. Neyshabur, and O. Saukh, ‘Understanding the effect of sparsity on neural networks’ robustness: Sparsity in Neural Networks<br/>Advancing Understanding and Practice’, Jul. 2021.

[36] S. Sonoda and N. Murata, ‘Neural network with unbounded activation functions is universal approximator’, *Applied and Computational Harmonic Analysis*, vol. 43, no. 2, pp. 233–268, Sep. 2017, doi: 10.1016/j.acha.2015.12.005.

[37] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: A Bradford Book, 2018.

[38] ‘Supervised vs. Unsupervised Learning: What’s the Difference?’, Mar. 12, 2021. https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning (accessed May 09, 2022).

[39] ‘Supervised, Unsupervised, & Reinforcement Learning’. https://machine-learning.paperspace.com/wiki/supervised-unsupervised-and-reinforcement-learning (accessed May 09, 2022).

[40] L. White, R. Togneri, W. Liu, and M. Bennamoun, ‘Introduction to Neural Networks for Machine Learning’, in *Neural Representations of Natural Language*, L. White, R. Togneri, W. Liu, and M. Bennamoun, Eds. Singapore: Springer, 2019, pp. 1–21. doi: 10.1007/978-981-13-0062-2_1.

[41] D. P. Kingma and J. Ba, ‘Adam: A Method for Stochastic Optimization’, 2014.

[42] A. Vandaele, *Mathématique 1: Calcul Différentiel et Intégral 1*, Mutuelle d’édition de la FPM’s. 2021.

[43] S. Kostadinov, ‘Understanding Backpropagation Algorithm’, *Medium*, Aug. 12, 2019. https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd (accessed May 11, 2022).

[44] K. Leung, ‘The Dying ReLU Problem, Clearly Explained’, *Medium*, Sep. 23, 2021. https://towardsdatascience.com/the-dying-relu-problem-clearly-explained-42d0c54e0d24 (accessed May 11, 2022).

[45] I. Murzakhanov, A. Venzke, G. S. Misyris, and S. Chatzivasileiadis, ‘Neural Networks for Encoding Dynamic Security-Constrained Optimal Power Flow’, *arXiv:2003.07939 [cs, eess, math]*, Oct. 2021, Accessed: Jan. 12, 2022. [Online]. Available: http://arxiv.org/abs/2003.07939

[46] ‘SMARTWATER’, *Multitel*. https://www.multitel.eu/projects/smartwater/ (accessed May 27, 2022).

[47] J.-F. Toubeau, ‘Modeling Nonlinear Effects In The Day-ahead Scheduling Of Pumped Storage Hydro Stations.’

[48] ‘Imbalance prices 15 min’. https://www.elia.be/en/grid-data/balancing/imbalance-prices-15-min (accessed May 25, 2022).

[49] B. Huang, Y. Chen, and R. Baldick, ‘A Configuration Based Pumped Storage Hydro Model in the MISO Day-Ahead Market’, *IEEE Transactions on Power Systems*, vol. 37, no. 1, pp. 132–141, Jan. 2022, doi: 10.1109/TPWRS.2021.3097270.

[50] H. -Y. Su *et al.*, ‘Developing an Optimal Scheduling of Taiwan Power System With Highly Penetrated Renewable Energy Resources and Pumped Hydro Storages’, *IEEE Transactions on Industry Applications*, vol. 57, no. 3, pp. 1973–1986, Jun. 2021, doi: 10.1109/TIA.2021.3057300.

[51] H. Alharbi and K. Bhattacharya, ‘Participation of Pumped Hydro Storage in Energy and Performance-Based Regulation Markets’, *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4307–4323, Nov. 2020, doi: 10.1109/TPWRS.2020.2998490.

[52] ‘Big-M and convex hulls’, *YALMIP*, Sep. 17, 2016. https://yalmip.github.io/tutorial/bigmandconvexhulls (accessed Jun. 15, 2022).

[53] F. Trespalacios and I. E. Grossmann, ‘Improved Big-M reformulation for generalized disjunctive programs’, *Computers & Chemical Engineering*, vol. 76, pp. 98–103, May 2015, doi: 10.1016/j.compchemeng.2015.02.013.

[54] Z. Chen, D. Kuhn, and W. Wiesemann, ‘Data-Driven Chance Constrained Programs over Wasserstein Balls’. arXiv, Sep. 01, 2018. Accessed: Jun. 02, 2022. [Online]. Available: http://arxiv.org/abs/1809.00210

[55] Z. De Grève, ‘Optimization Tools for Energy Systems’. Feb. 2021.

[56] J. M. Morales, A. J. Conejo, H. Madsen, P. Pinson, and M. Zugno, *Integrating Renewables in Electricity Markets: Operational Problems*, 2014th ed., vol. 205. Boston, MA: Boston, MA: Springer US. doi: 10.1007/978-1-4614-9411-9.

[57] X. Wu, ‘Lecture 3: Convex Functions’, p. 33, Fall 2019.

Appendix

A. Introduction to optimization

Definition

Optimization "aims at finding the best way to achieve a given objective (or multiple objectives), given a series of constraints" [55]. Mathematical optimization problems make this process quantitative, hence leading the decision-maker to take a numerical well-informed decision. There exists a broad range of optimization problems in terms of structure but also complexity.

Vocabulary and formalism

A generic mathematical optimization problem can be formulated as:

$$\min_{x} f(x) \tag{A - 1a}$$

$$\text{s.t. } h(x) = 0, \tag{A - 1b}$$

$$g(x) \leq 0. \tag{A - 1c}$$

In the above problem, three major elements appear:

1. $x \in \mathbb{R}^n$ is the vector of dimension n containing the *decision variables*. Those variables can be seen as degrees of freedom upon which the decision-maker can act to optimize the objective function. The goal of optimization is to give the decision-maker the value of each decision variable corresponding to the optimal solution. Decision variables must not be confused with parameters or problem data which are exogeneous to the problem upon which the decision-maker cannot act.
2. $f(.) : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the *objective function* of the defined problem. It establishes a relation between the value of the decision variables and the desirability of this solution with respect to the decision-maker. It quantifies how good the current decision is. The objective function can be a minimization or a maximization and represent many different things e.g.: cost to minimize, benefit to maximize, voltage deviations to minimize, etc.

3. $h(.) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ and $g(.) : \mathbb{R}^n \longrightarrow \mathbb{R}^l$ are vector-valued functions with the decision variables as input. They define the m equality *constraints* and the l inequality constraints[10] of the problem, respectively.

Together, $h(.)$ and $g(.)$ define what is named the *feasible set*. The feasible set is the portion of $\mathbb{R}^n$ in which the decision variables are allowed to take their value from in order to consitute a feasible solution (i. e., a solution which satisfies all the constraints enforced by $h(.)$ and $g(.)$). Usually, for real-life problems, a solution outside of the feasible set (i.e., infeasible) leads to a solution that cannot be implemented in reality. Therefore, the objective function must be optimized on the feasible set [56].

Classification

There exist a vast range of optimization problems and many different features and categories. An optimization problem can be:

1. Convex vs. non-convex

In convex problems, three additional conditions must be fulfilled. Firstly, the objective function must be convex. Secondly, the convex set must be convex which means that $h(.)$ are affine functions and $g(.)$ are convex functions.

Convex function - A function $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ is convex if **dom** f is a convex set and

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

Geometrically, the condition can be interpreted as: for any two points A and B belonging to f, all points belonging to the straight-line segment connecting those two points are

Convex set - A set $C \in \mathbb{R}^n$ is convex if

$$\theta x + (1-\theta)y \in C$$

for all points $x, y \in C$ and $\theta \in [0, 1]$.

located above the function [57]. Figure 34 illustrates the condition for a convex and a non-convex function. It is worth noting that a function is concave if its opposite is convex.

[10] The alert reader can note the absence of greater-than-or-equal-to ($\geq$) constraints. That is because they can be transformed into smaller-than-or-equal-to ($\geq$) constraints by multiplying both side of the equation by -1. In addition, strict inequalities (i.e., greater-than ($>$) and smaller-than ($<$)) introduce complexities in problem solving and are therefore avoided.

Geometrically, a set Ω is convex if for any two points A and B belonging to Ω, all points belonging to the straight-line segment connecting those two points are part of Ω [57]. Figure 35 depicts a convex set followed by a non-convex set.

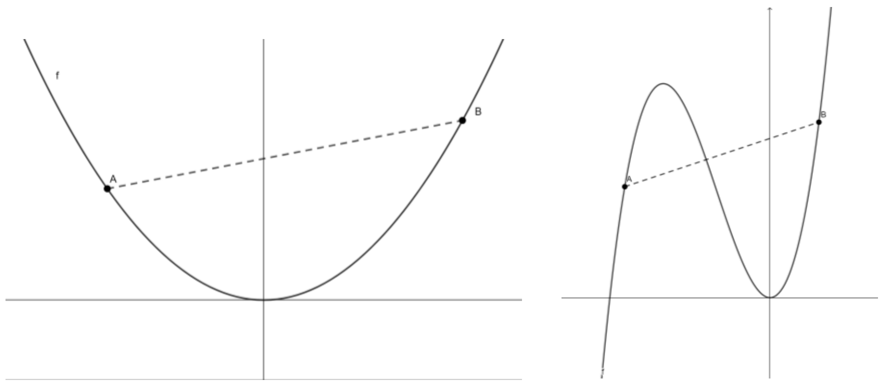


Figure 34: (a) $f = x^2$ is a convex function; (b) $f = x^3 + 3x^2$ is a non-convex function (click on the figure for an interactive view).

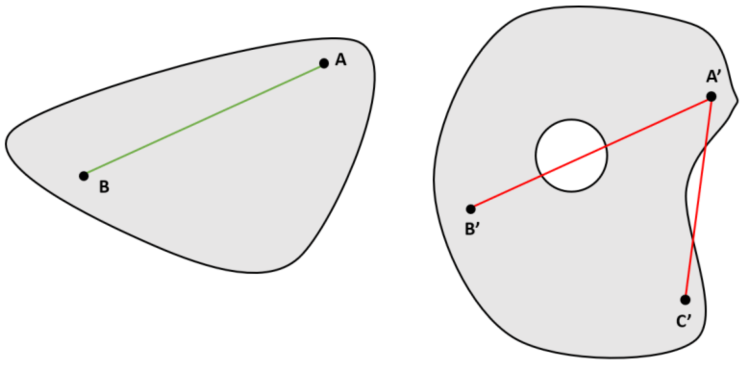


Figure 35: (a) sketch of a convex set; (b) sketch of a non-convex set.

Convexity is an extremely significant property in optimization because it allows the solver (i.e., the algorithm which determines the optimal solution) to find the global optimum of the problem. If the problem is not convex, there is a risk that the solution returned by the solver is a local optimum.

2. *Linear Programming Problem (LPP) vs. Non-Linear Programming Problem (NLPP)*

In LPPs, the constraints $h(.)$ and $g(.)$ as well as the objective function $f(.)$ must be linear while there are no conditions for NLPPs. If a problem is linear then, it belongs to the class of convex problem (Figure 36).

3. *Continuous vs. integer-valued*

In continuous problems, one has $x \in \mathbb{R}^n$. In integer-valued problems, some of the decisions variables may be binary variables (i.e., variables that can be 0 or 1), integer or natural numbers only. If a problem is linear then it is named Mixed Integer Linear Programming Problem (MILPP). Remarkably, a problem is non-convex if any of its decision variables allow values from a discrete set.

4. Deterministic vs. non-deterministic

In deterministic optimization, there is no uncertainty about the parameters and the data used in the optimization model. If a decision-maker wants to take the best decision considering uncertainty, the problem becomes non-deterministic as random variables must be considered. There exist three main subclasses of non-deterministic problems, ranked from the least to the most conservative (i.e., the attitude of hedging against risk): (i) stochastic, (ii) chance-constrained, (iii) robust.

In stochastic optimization, the objective function becomes the expectation of the initial objective function over the probability distributions of all the uncertain parameters ξ (Equations **Error! Reference source not found.** - (A - 2b)). The solution is now the value of the decision variables which optimize the expectation of the objective function considering the uncertainty of the parameters. It is also possible to use scenarios if the probability distributions are unknown. In this case, the solution is the value of the decision variables which optimize the expectation over N considered scenarios (Equations (A - 3a) - (A - 3b)). Interestingly, if an infinity of scenarios is known then, the probability distributions are known, and one falls back on the previous case.

$$\min_{x} \mathbb{E}_{\xi}[f(x), \xi] \tag{A - 2a}$$

$$\text{s. t. } \mathbb{E}_{\xi}[g(x), \xi] \leq 0 \tag{A - 2b}$$

$$\min_{x} \frac{1}{N} \sum_{i=1}^{N} [f(x), \xi_i] \tag{A - 3a}$$

$$\text{s. t.} \frac{1}{N} \sum_{i=1}^{N} [g(x), \xi_i] \leq 0 \tag{A - 3b}$$

In chance-constrained optimization, the constraints are enforced with a given probability $1 - \epsilon$ chosen by the decision-maker (Equations (A - 4a) - (A - 4b)). Therefore, a risk attitude towards the respect of the constraints can be selected by varying the hyperparameter ϵ. Once again, the probability distributions can be approximated by scenarios.

$$\min_{x} \mathbb{E}_{\xi}[f(x), \xi] \tag{A - 4a}$$

$$\text{s.t.}\ \mathbb{P}([g_k(x), \xi] \leq 0\ \forall k\ \in \{1, \dots, K\}) \geq 1 - \epsilon \tag{A - 4b}$$

Finally, the robust optimization consists in optimizing the solution in the worst-case scenario. If the problem is a minimization, then the robust optimization will assign the values which minimize the maximum of the objective function considering the uncertainty set $\mathcal{U}$ for the parameters to the decision variables (Equations (A - 5a) - (A - 5b)). Instead of using probability distributions, the maximum of f can be determined using scenarios.

$$\min_{x} \max_{\xi \in \mathcal{U}} f(x, \xi) \tag{A - 5a}$$

$$\text{s.t.}\ g(x, \xi) \geq 0 \quad \forall\, \xi \in \mathcal{U} \tag{A - 5b}$$

5. *Mono-objective vs. multi-objective*

In mono-objective problems, there is one objective function $f(\,.\,) : \mathbb{R}^n \longrightarrow \mathbb{R}$ whereas in multi-objective problems the output of $f(\,.\,)$ is multi-dimensional ($f(\,.\,) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$). The solution becomes a set named *Pareto front*.

6. *Single agent vs. multi-agent*

In single agent problems, there is one actor. If multiple actors are involved, problems become multi-agent. Each actor has its own set of decisions variables and tries to optimize its own objective function which can be conflictual with others.

7. *Static vs. dynamic*

Dynamic programming problems include sequential decision making, typically, regarding time.

Figure 36 summarizes the various types of mathematical optimization problems and the ease to solve them.

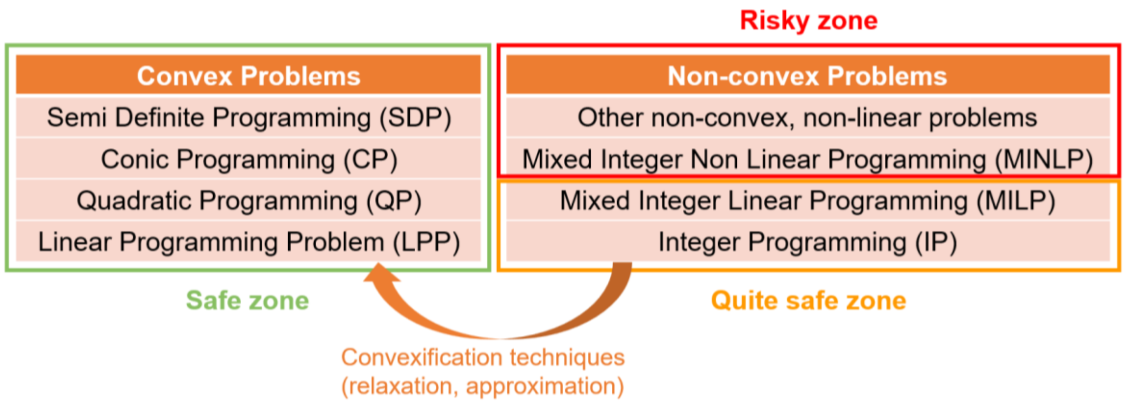

Risky zone
Convex Problems
Semi Definite Programming (SDP)
Conic Programming (CP)
Quadratic Programming (QP)
Linear Programming Problem (LPP)
Non-convex Problems
Other non-convex, non-linear problems
Mixed Integer Non Linear Programming (MINLP)
Mixed Integer Linear Programming (MILP)
Integer Programming (IP)
Safe zone
Quite safe zone
Convexification techniques
(relaxation, approximation)

Figure 36: Summary of the different categories of optimization problems and the ease to solve them [55].

B. Complementary Results: NN Approximation

Table 7: Result for a NN with no sparsity imposed, with one deep layer made of one neuron having a ReLU activation function and using the piecewise reformulation (see section 3.3.1, Formulation 1).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	stepwise	1350	0.021782	0	0	0.083333
1 x 1	piecewise	1350	483.6834	1669.706	0.206468	962.3142
1 x 1	cons. pcw	1350	112.3442	1558.808	0.058917	1560.551
2 x 1	stepwise	1398	0.262678	624.0625	0.01786	622.8684
2 x 1	piecewise	1398	1466.952	1653.221	0.186606	1268.546
2 x 1	cons. pcw	1398	481.5465	1582.103	0.092744	1531.588
3 x 1	stepwise	1446	13.18081	1403.872	0.064428	1404.863
3 x 1	piecewise	1446	7454.505	1652.454	0.183953	1331.052
3 x 1	cons. pcw	1446	2856.528	1586.722	0.097482	1528.568
4 x 1	stepwise	1494	158.3939	1494.539	0.164288	1499.365
4 x 1	piecewise	1494	13418.11	1650.519	0.180254	1333.365
4 x 1	cons. pcw	1494	7187.761	1592.004	0.100245	1524.988

Table 8: Result for a NN with no sparsity imposed, with one deep layer made of one neuron having a ReLU activation function and using the reformulation proposed in [45] (see section 3.3.1, Formulation 2).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	stepwise	1350	0.021782	0	0	0.083333
1 x 1	piecewise	1350	483.6834	1669.706	0.206468	962.3142
1 x 1	cons. pcw	1350	112.3442	1558.808	0.058917	1560.551
2 x 1	stepwise	1398	0.262678	624.0625	0.01786	622.8684
2 x 1	piecewise	1398	1466.952	1653.221	0.186606	1268.546
2 x 1	cons. pcw	1398	481.5465	1582.103	0.092744	1531.588
3 x 1	stepwise	1446	13.18081	1403.872	0.064428	1404.863
3 x 1	piecewise	1446	7454.505	1652.454	0.183953	1331.052
3 x 1	cons. pcw	1446	2856.528	1586.722	0.097482	1528.568
4 x 1	stepwise	1494	158.3939	1494.539	0.164288	1499.365
4 x 1	piecewise	1494	13418.11	1650.519	0.180254	1333.365
4 x 1	cons. pcw	1494	7187.761	1592.004	0.100245	1524.988

Table 9: Result for a NN with a sparsity of 50% imposed, with one deep layer made of one neuron having a ReLU activation function and using the piecewise reformulation (see section 3.3.1, Formulation 1).

n° interval (h*p)	**Approximation**	**n° variables**	**Time [s]**	**Expected [€]**	**MAE on Q [m^3/s]**	**Ex-post [€]**
1 x 1	stepwise	1350	0.021782	0	0	0.083333
1 x 1	piecewise	1350	483.6834	1669.706	0.206468	962.3142
1 x 1	cons. pcw	1350	112.3442	1558.808	0.058917	1560.551
2 x 1	stepwise	1398	0.262678	624.0625	0.01786	622.8684
2 x 1	piecewise	1398	1466.952	1653.221	0.186606	1268.546
2 x 1	cons. pcw	1398	481.5465	1582.103	0.092744	1531.588
3 x 1	stepwise	1446	13.18081	1403.872	0.064428	1404.863
3 x 1	piecewise	1446	7454.505	1652.454	0.183953	1331.052
3 x 1	cons. pcw	1446	2856.528	1586.722	0.097482	1528.568
4 x 1	stepwise	1494	158.3939	1494.539	0.164288	1499.365
4 x 1	piecewise	1494	13418.11	1650.519	0.180254	1333.365
4 x 1	cons. pcw	1494	7187.761	1592.004	0.100245	1524.988

Table 10: Result for a NN with no sparsity imposed, with one deep layer made of two neurons having a ReLU activation function and using the piecewise reformulation (see section 3.3.1, Formulation 1).

n° interval (h*p)	**Approximation**	**n° variables**	**Time [s]**	**Expected [€]**	**MAE on Q [m^3/s]**	**Ex-post [€]**
1 x 1	stepwise	1590	0.167219	0	0	0.083333
1 x 1	piecewise	1590	7671.371	1622.981	0.243728	1284.743
1 x 1	cons. pcw	1590	5693.424	1515.17	0.051548	1515.778
2 x 1	stepwise	1638	0.408024	667.6776	0.008064	652.6582
2 x 1	piecewise	1638	16018.04	1642.49	0.248741	1231.444
2 x 1	cons. pcw	1638	61869.46	1519.855	0.053527	1523.483
3 x 1	stepwise	1686	4.894634	1362.931	0.077217	1364.16
3 x 1	piecewise	1686	690.7917	1638.712	0.218684	1624.207
3 x 1	cons. pcw	1686	800.6902	1520.972	0.053951	1521.226
4 x 1	stepwise	1734	60.4579	1469.64	0.15433	1472.17
4 x 1	piecewise	1734	1032.22	1642.668	0.23462	1277.937
4 x 1	cons. pcw	1734	1169.961	1516.631	0.07784	1518.622

Table 11: Result for a NN with sparsity imposed to 50%, with one deep layer made of two neurons having a ReLU activation function and using the piecewise reformulation (see section 3.3.1, Formulation 1).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	stepwise	1590	0.008003	0	0	0.083333
1 x 1	piecewise	1590	777.926	1732.706	0.242534	998.8659
1 x 1	cons. pcw	1590	124.5264	1549.753	0.056071	1535.705
2 x 1	stepwise	1638	0.078024	626.0452	0.016614	625.2823
2 x 1	piecewise	1638	973.8306	1715.656	0.167269	1489.717
2 x 1	cons. pcw	1638	581.5671	1566.002	0.067903	1523.22
3 x 1	stepwise	1686	6.00123	1398.438	0.069482	1400.226
3 x 1	piecewise	1686	2730.946	1714.803	0.170202	1513.935
3 x 1	cons. pcw	1686	4344.468	1569.22	0.070953	1520.884
4 x 1	stepwise	1734	148.2924	1492.928	0.160152	1497.422
4 x 1	piecewise	1734	17509.57	1713.758	0.175235	1507.249
4 x 1	cons. pcw	1734	35329.43	1574.592	0.167626	1574.271

Table 12: Result for a NN with sparsity imposed to 50%, with one deep layer made of four neurons having a ReLU activation function and using the piecewise reformulation (see section 3.3.1, Formulation 1).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	stepwise	2070	0.043011	0	0	0.083333
1 x 1	piecewise	2070	281.727	1597.954	0.096621	1480.51
1 x 1	cons. pcw	2070	321.3462	1515.345	0.054905	1517.166
2 x 1	stepwise	2118	0.12019	651.4588	0.008137	651.4818
2 x 1	piecewise	2118	642.4211	1591.078	0.09991	1484.39
2 x 1	cons. pcw	2118	627.2589	1526.191	0.047189	1527.174
3 x 1	stepwise	2166	7.226135	1333.53	0.095842	1335.192
3 x 1	piecewise	2166	3849.96	1599.565	0.231196	1235.876
3 x 1	cons. pcw	2166	1587.392	1532.105	0.046271	1531.863
4 x 1	stepwise	2214	86.64341	1467.528	0.175115	1471.659
4 x 1	piecewise	2214	4644.653	1585.744	0.09939	1490.219
4 x 1	cons. pcw	2214	2187.45	1536.968	0.045664	1539.559

Table 13: Result for a NN with sparsity imposed to 50%, with one deep layer made of four neurons having a ReLU activation function and using the reformulation proposed in [45] (see section 3.3.1, Formulation 2).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	stepwise	1590	0.025009	0	0	0.083333
1 x 1	piecewise	1590	222.0115	1597.954	0.096621	1480.51
1 x 1	cons. pcw	1590	108.9848	1515.34	0.0549	1517.173
2 x 1	stepwise	1638	0.123016	651.4588	0.008137	651.4818
2 x 1	piecewise	1638	1290.024	1591.078	0.09991	1484.39
2 x 1	cons. pcw	1638	477.2959	1526.186	0.047187	1527.179
3 x 1	stepwise	1686	9.378134	1333.53	0.095842	1335.192
3 x 1	piecewise	1686	2395.404	1599.565	0.231196	1235.876
3 x 1	cons. pcw	1686	1281.31	1532.098	0.046269	1531.874
4 x 1	stepwise	1734	93.4639	1467.528	0.175115	1471.659
4 x 1	piecewise	1734	803.5562	1585.744	0.09939	1490.219
4 x 1	cons. pcw	1734	2959.654	1536.976	0.045667	1539.568

Table 14: Result for a NN with sparsity imposed to 50%, with two deep layers made of two neurons having a ReLU activation function and using the piecewise reformulation (see section 3.3.1, Formulation 1).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	stepwise	2070	0.009203	0	0	0.083333
1 x 1	piecewise	2070	70.20746	1710.657	0.297965	688.9163
1 x 1	cons. pcw	2070	45.60111	1553.537	0.074005	1555.164
2 x 1	stepwise	2118	0.089211	623.084	0.01786	623.1027
2 x 1	piecewise	2118	156.4544	1687.7	0.218355	1092.409
2 x 1	cons. pcw	2118	113.6439	1581.24	0.078509	1530.399
3 x 1	stepwise	2166	0.633781	1345.957	0.142818	1351.946
3 x 1	piecewise	2166	595.4057	1691.204	0.220384	1139.368
3 x 1	cons. pcw	2166	405.9132	1585.842	0.082292	1527.372
4 x 1	stepwise	2214	8.449522	1482.915	0.159692	1487.337
4 x 1	piecewise	2214	1138.356	1686.132	0.148477	1528.282
4 x 1	cons. pcw	2214	730.988	1591.072	0.08645	1523.766

Table 15: Result for a NN with sparsity imposed to 50%, with two deep layers made of two neurons having a ReLU activation function and using the reformulation proposed in [45] (see section 3.3.1, Formulation 2).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	stepwise	1590	0.009001	0	0	0.083333
1 x 1	piecewise	1590	72.63914	1711.864	0.319729	653.6479
1 x 1	cons. pcw	1590	92.21221	1553.537	0.074005	1555.164
2 x 1	stepwise	1638	0.09017	594.0741	0.01786	594.0927
2 x 1	piecewise	1638	1295.905	1692.66	0.235057	1071.705
2 x 1	cons. pcw	1638	204.8034	1581.24	0.078509	1530.399
3 x 1	stepwise	1686	1.986164	1362.869	0.070359	1364.668
3 x 1	piecewise	1686	679.8754	1691.204	0.220384	1139.368
3 x 1	cons. pcw	1686	519.0123	1585.842	0.082292	1527.372
4 x 1	stepwise	1734	10.90975	1484.34	0.163901	1489.17
4 x 1	piecewise	1734	2136.972	1689.515	0.214786	1145.128
4 x 1	cons. pcw	1734	2093.093	1591.069	0.086445	1523.758

Table 16: Result for a NN with no sparsity imposed, with one deep layer made of one neuron having a Leaky ReLU activation function and using the reformulation proposed in [45] (see section 3.3.2, Formulation 1).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	stepwise	1350	0.02183	0	0	0.083333
1 x 1	piecewise	1350	297.5587	1660.405	0.201523	1165.171
1 x 1	cons. pcw	1350	165.1039	1560.062	0.085197	1559.543
2 x 1	stepwise	1398	0.062786	627.9538	0.015257	627.7896
2 x 1	piecewise	1398	1125.489	1644.548	0.155165	1399.073
2 x 1	cons. pcw	1398	448.973	1568.089	0.120519	1547.978
3 x 1	stepwise	1446	7.653423	1422.681	0.062977	1424.133
3 x 1	piecewise	1446	13434.82	1642.439	0.151104	1474.574
3 x 1	cons. pcw	1446	5008.719	1571.679	0.12089	1542.628
4 x 1	stepwise	1494	24.46171	1507.384	0.150516	1510.971
4 x 1	piecewise	1494	12580.89	1642.717	0.202825	1574.026
4 x 1	cons. pcw	1494	3731.641	1574.119	0.122357	1541.932

Table 17: Result for a NN with sparsity imposed to 50%, with one deep layer made of one neuron having a Leaky ReLU activation function and using the reformulation proposed in [45] (see section 3.3.2, Formulation 1).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	stepwise	1350	0.005987	0	14.93056	0.083333
1 x 1	piecewise	1350	44.38011	1655.73	56.04167	1108.397
1 x 1	cons. pcw	1350	29.64054	1558.629	18.05556	1558.848
2 x 1	stepwise	1398	0.056015	589.2827	6.319444	589.305
2 x 1	piecewise	1398	139.0958	1644.803	53.33333	1401.545
2 x 1	cons. pcw	1398	77.12562	1567.196	47.98611	1544.907
3 x 1	stepwise	1446	1.512867	1422.066	14.86111	1422.925
3 x 1	piecewise	1446	1036.308	1638.513	53.33333	1570.036
3 x 1	cons. pcw	1446	376.4455	1570.706	19.16667	1540.288
4 x 1	stepwise	1494	8.488192	1505.722	22.5	1510.026
4 x 1	piecewise	1494	1025.662	1639.935	56.18056	1550.693
4 x 1	cons. pcw	1494	1017.336	1573.243	23.47222	1538.568

Table 18: Result for a NN with no sparsity imposed, with one deep layer made of two neurons having a Leaky ReLU activation function and using the reformulation proposed in [45] (see section 3.3.2, Formulation 1).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m^3/s]	Ex-post [€]
1 x 1	stepwise	1590	0.461756	0	0	0.083333
1 x 1	piecewise	1590	9046.048	1664.416	0.296526	885.0341
1 x 1	cons. pcw	1590	17442.29	1489.679	0.125758	1492.418
2 x 1	stepwise	1638	0.389067	648.0582	0.005149	648.0764
2 x 1	piecewise	1638	102550.5	1642.017	0.201314	1150.472
2 x 1	cons. pcw	1638	282878.5	1495.429	0.055187	1496.059
3 x 1	stepwise	1686	8.268425	1352.388	0.082043	1353.859
3 x 1	piecewise	1686	2488.399	1669.63	0.198017	1177.024
3 x 1	cons. pcw	1686	16305.37	1497.956	0.054834	1500.638
4 x 1	stepwise	1734	279.1749	1443.731	0.151099	1448.292
4 x 1	piecewise	1734	19473.67	1664.472	0.192161	1210.058
4 x 1	cons. pcw	1734	65678.16	1513.025	0.191963	1511.123

Table 19: Result for a NN with sparsity imposed to 50%, with one deep layer made of two neurons having a Leaky ReLU activation function and using the reformulation proposed in [45] (see section 3.3.2, Formulation 1).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m³/s]	Ex-post [€]
1 x 1	stepwise	1590	0.008003	0	12.63889	0.083333
1 x 1	piecewise	1590	118.5255	1626.901	19.51389	1493.443
1 x 1	cons. pcw	1590	64.73381	1528.237	43.125	1526.904
2 x 1	stepwise	1638	0.207045	360.7114	17.01389	360.7947
2 x 1	piecewise	1638	355.9276	1610.276	65.27778	1603.812
2 x 1	cons. pcw	1638	179.9117	1545.29	16.11111	1542.292
3 x 1	stepwise	1686	2.401039	1404.142	22.70833	1405.477
3 x 1	piecewise	1686	1366.049	1611.195	52.56944	1603.331
3 x 1	cons. pcw	1686	756.3878	1549.384	21.38889	1543.876
4 x 1	stepwise	1734	19.56302	1487.949	19.79167	1489.387
4 x 1	piecewise	1734	3428.97	1612.33	0.234527	1607.589
4 x 1	cons. pcw	1734	1718.831	1552.745	0.098569	1542.593

Table 20: Result for a NN with sparsity imposed to 50%, with two deep layer made of two neurons having a Leaky ReLU activation function and using the reformulation proposed in [45] (see section 3.3.2, Formulation 1).

n° interval (h*p)	Approximation	n° variables	Time [s]	Expected [€]	MAE on Q [m³/s]	Ex-post [€]
1 x 1	stepwise	2070	0.018999	0	0	0.083333
1 x 1	piecewise	2070	8821.959	1583.21	0.073771	1541.774
1 x 1	cons. pcw	2070	6463.381	1491.887	0.0916	1492.901
2 x 1	stepwise	2118	0.308071	620.0431	0.007306	620.058
2 x 1	piecewise	2118	4770.811	1562.59	0.08766	1564.878
2 x 1	cons. pcw	2118	1527.534	1492.579	0.094948	1494.626
3 x 1	stepwise	2166	57.64124	1364.007	0.085238	1364.764
3 x 1	piecewise	2166	524.473	1559.054	0.061085	1521.42
3 x 1	cons. pcw	2166	60525.3	1497.942	0.054423	1500.226
4 x 1	stepwise	2214	1297.132	1481.696	0.152285	1485.42
4 x 1	piecewise	2214	3904.15	1554.711	0.087754	1557.758
4 x 1	cons. pcw	2214	97861.89	1502.246	0.094948	1505.356

C. Script And Code Files

All the script and code files used in this work are available upon request.