UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Power system inertia estimation using machine learning techniques

Diploma Thesis

Themistoklis Proko

Supervisor: Eleftherios Kontis

September 2025

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Power system inertia estimation using machine learning techniques

Diploma Thesis

Themistoklis Proko

Supervisor: Eleftherios Kontis

September 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Προσδιορισμός των επιπέδων αδράνειας συστημάτων ηλεκτρικής ενέργειας με χρήση μεθόδων μηχανικής μάθησης

Διπλωματική Εργασία

Θεμιστοκλής Πρόκο

Επιβλέπων/πουσα: Ελευθέριος Κόντης

Σεπτέμβριος 2025

Approved by the Examination Committee:

Supervisor **Eleftherios Kontis**
Assistant Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Fotios Plessas**
Proffesor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Daskalopulu Aspassia**
Accosiate Proffesor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

Having reached the finish line of my post-graduate studies, I would like to express my deepest gratitude to all those who supported me throughout this beautiful journey. First and foremost, I wish to thank my family from the bottom of my heart. Their unwavering financial and psychological support was a pivotal factor in the completion of my studies.Furthermore, I extend my sincere thanks to my friends and classmates for the unforgettable memories and experiences we shared over the past four years in the city of Volos, as well as for the support they provided me in both good and challenging times.Lastly, I am immensely grateful to my supervisor, Professor Eleftherios Kontis, whose contribution to this final result has been essential. His patience, consistency, and profound insights at every stage of the project were a tremendous source of motivation and inspiration to me.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Themistoklis Proko

Diploma Thesis

Power system inertia estimation using machine learning techniques

Themistoklis Proko

Abstract

As centralized synchronous generators are gradually replaced by converter-based renewable units, the system's rotational inertia has declined, reducing its ability to counter frequency disturbances. This thesis develops and evaluates several machine-learning and deep-learning models capable of real-time estimation of regional inertia constants by optimally fusing wide-area frequency measurements with tie-line active-power flows. Frequency-response models of a multi-area system are built in MATLAB/Simulink to generate training and validation data under varying inertia levels, generation mixes, and region counts. Implemented in Python, the proposed estimators demonstrate excellent performance, achieving mean absolute percentage errors below 4 % across multiple scenarios of generation source mixtures.Real-time inertia estimates enable network operators to dynamically adjust remedial actions to ensure frequency stability of low-inertia power systems.

Keywords:

Conventional power plants,frequency response,interconnected region,power grid,rotational inertia,tie-line active-power flows,

Διπλωματική Εργασία

Προσδιορισμός των επιπέδων αδράνειας συστημάτων ηλεκτρικής ενέργειας με χρήση μεθόδων μηχανικής μάθησης

Θεμιστοκλής Πρόκο

Περίληψη

Καθώς οι συμβατικές μονάδες παραγωγής ισχυύος αποτελούμενες απο σύγχρονες γεννήτριες αντικαθίστανται σταδιακά από Ανανεώσιμες Πηγές Ενέργειας (ΑΠΕ) διασυνδεδεμένες μέσω μετατροπέων ισχύος, η αδράνεια του Συστήματος Ηλεκτρικής Ενέργειας (ΣΗΕ) ελαττώνεται σημαντικά, μειώνοτας την ικανότητά του δικτύου να ανταποκρίνεται σε διαταραχές συχνότητας. Στην παρούσα διπλωματική εργασία αναπτύσσονται και αξιολογούνται μοντέλα μηχανικής και βαθιάς μάθησης για την εκτίμηση σε πραγματικό χρόνο των σταθερών αδράνειας ανά περιοχή του διασυνδεδεμένου ΣΗΕ, μέσω συνδυασμού των μετρήσεων συχνότητας ανά περιοχή και ενεργού ισχύος στις διασυνδετικές γραμμές του ΣΗΕ. Για τη δημιουργία δεδομένων εκπαίδευσης και επικύρωσης χρησιμοποιούνται μοντέλα απόκρισης συχνότητας τόσο δύο όσο και τριών περιοχών αντίστοιχα, υλοποιημένα σε MATLAB/Simulink, με μεταβαλλόμενα επίπεδα αδράνειας και μίγματα παραγωγής. Τα μοντέλα πρόβλεψης που αναπτύχθηκαν σε Python, προσδιορίζουν με εξαιρετική ακρίβεια τις τιμές αδρανειών με τα περισσότερα να παρουσιάζουν μέσο απόλυτο ποσοστιαίο σφάλμα κάτω από 4 %. Οι εκτιμήσεις των επιπέδων αδράνειας σε πραγματικό χρόνο επιτρέπουν στους διαχειριστές των συστημάτων μεταφοράς την έγκαιρη εφαρμογή διορθωτικών ενεργειών,όπως για παράδειγμα την άμεση έντάξη σύγχρονων γεννητριών σε περιπτώσεις χαμηλής αδράνειας ώστε να βελτιωθεί η ευστάθεια του συστήματος.

Λέξεις-κλειδιά:

Αδράνεια,απόκριση συχνότητας,μεταβολή ισχύος διασυνδέσεων,περιοχές διασυνδεδεμένου ΣΗΕ,συμβατικές πήγες ενεργείας,ΣΗΕ

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Although the idea of generating electricity exclusively from inexhaustible sources with a negligible carbon footprint seems impressive and ambitious, the truth is that the extensive integration of renewable energy sources has introduced numerous new challenges related to the stability and reliable operation of modern power systems.One of the most significant impacts of this energy transition lies in the reduction of inertia levels in modern power systems. Indeed, the rotating masses located in the conventional generators of traditional power plants (such as in steam and gas turbines) provide natural inertia to the grid through their kinetic energy. As these conventional units are gradually decommissioned, their place is taken by Renewable Energy Source (RES) units, which are connected to the grid via power converters and, therefore, do not directly contribute to inertia due to the absence of a mechanical coupling with the grid.

1.1 Thesis Objection

This thesis aims to develop machine and deep learning models for the most accurate estimation possible of the total system inertia per area in an interconnected power system. Knowledge of the available inertia in real-time enables the system operator to make timely and radical decisions, by dispatching the necessary generation units to guarantee the system's frequency stability and effectively prevent undesirable situations, such as widespread brownouts or system-wide blackouts.

The contribution of this work lies in its utilization of modern artificial intelligence techniques to predict a critical, yet difficult-to-measure, physical quantity like inertia. It provides

a decision-support tool for network operators, especially in environments with high penetration of Renewable Energy Sources (RES), where traditional physical inertia is significantly reduced. Consequently, this work contributes both to enhancing the stability of modern power systems and to facilitating a smooth transition towards a more sustainable and ”smart” energy future.

1.1.1 Contribution

The contribution of this thesis is summarized as follows:

1. A study of various generator unit combinations was conducted on a standard two-area interconnected power system, implemented in the `MATLAB/Simulink` environment.

2. For each unique combination of inertia values per area (H_1, H_2), the corresponding system configuration was simulated in the `Simulink` environment. Each simulation recorded the dynamic frequency responses of the two areas, as well as the dynamic response of the power change in the tie-line. In total, 1000 simulations were executed, each with a different pair of (H_1, H_2) values. During these simulations, a probing signal (a small disturbance) was applied via a battery energy storage system. The probing signal was modeled in combination with the existing, randomly fluctuating power demand per second. 80% of the generated data was used for model training, while the remaining 20% was utilized for result validation.

3. A total of five machine learning algorithms were applied to estimate the total inertia in each generation scenario, with the goal of comparatively evaluating their ability to accurately approximate the power system’s actual inertia values. The algorithms used were: Random Forest, Multiple Linear Regression, XGBoost, Support Vector Regression, and a Multilayer Perceptron.

4. The performance of each algorithm was evaluated based on its prediction error. It was found that the MLP algorithm provided particularly high accuracy, with a mean absolute percentage error of less than 4% in estimating the inertia of both areas in all examined scenarios of combinations of the different power plants.

5. The same methodology was also applied to a more extensive scenario with three interconnected areas and multiple generation units per area, which increased the demands

for data processing and model generalization control. The performance of the models was significantly reduced, with the sole exception of the MLP model, which continued to determine the three output values (H_1, H_2, H_3) with impressive accuracy, with the error limited to just 4% per output.

1.2 Thesis Organization

Chapter 2 analyzes the concept and significance of inertia in power systems. In Chapter 3, the reader can study the operational principle of each algorithm used for the task of the inertia estimation in this project. Chapter 4 focuses on the modeling and description of the employed System Frequency Response (SFR) models, which are adopted in this thesis to study the frequency response and dymanic behavior of interconnected power systems. Chapter 5 elaborates on the proposed methodology. Emphasis is placed on the collection and preprocessing of data generated from the simulations of the frequency response models in Simulink. Chapter 6 presents an interactive and comparative evaluation of all the inertia estimation methods. Chapter 7 extends the analysis to the more complex scenario of a system with three interconnected areas. Finally, Chapter 8 presents the conclusions of the study and suggests directions for future research on related problems.

Chapter 2

The Concept of Inertia in Power Systems

2.1 Introduction

The concept of inertia is known in physics as the tendency of bodies to resist any change in their kinetic state. In the case of synchronous generators, inertia reflects the tendency of their rotating masses to maintain their existing kinetic state and continue their rotational motion. Knowledge of inertia in a power system is crucial for its smooth and reliable operation. The higher the level of inertia, the smaller the frequency deviation in the event of disturbances in the generation-load balance [1].

2.2 Definition of Inertia

Inertia in power systems is related to the kinetic energy stored in the rotating masses of synchronous generators, namely in the turbines and rotors of the generating units. This physical property provides the grid with immediate, automatic, and passive resistance to abrupt frequency changes when sudden imbalances between generated and consumed power occur (e.g., due to the loss of a generating unit or a sudden load change). Simply put, the more inertia a system possesses, the slower and more smoothly the frequency changes, providing critical time for the activation of control mechanisms, such as the primary frequency response. In contrast, in low-inertia systems, the frequency can change much more rapidly (high Rate of Change of Frequency - RoCoF), leading to the activation of protection systems, such as under-frequency load shedding (UFLS), or even to blackouts [2].

As traditional units with synchronous generators are replaced by photovoltaics and wind

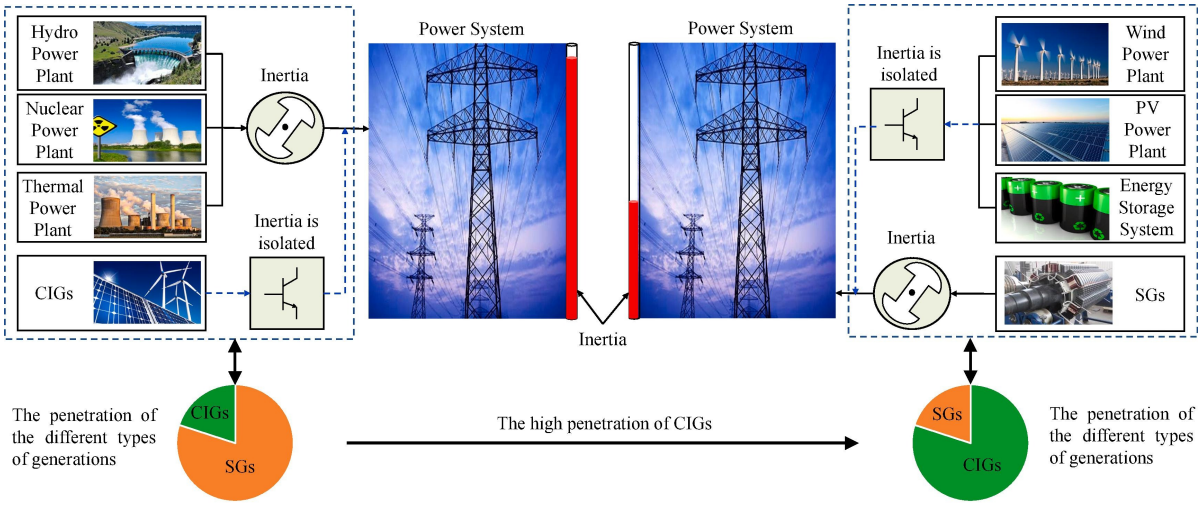

Figure 2.1: The reduction of inertia in a power system with high penetration of power electronics [1].

turbines connected through power converters (which do not provide natural inertia), the total inertia of the grid becomes dynamic and variable (Figure 2.1). This makes the real-time estimation of inertia (online inertia estimation) essential, in order to have an immediate picture of the system's state and to make timely decisions for enhancing stability.

Every conventional energy source with a rotating mass operates in synchronism with the other generators in the grid and contributes to the total rotational inertia value of each area, as shown in Figure 2.2.

The inertia constant of each individual synchronous generator with a rotating mass is described by the following formula [1]:

$$H_{\text{SG}} = \frac{E_k}{S_B} = \frac{\frac{1}{2} \cdot J \cdot \omega_{m0}^2}{S_B} \tag{2.1}$$

Where:

- E_k is the kinetic energy of the rotor (Joules),
- J is the moment of inertia of the rotor (kg·m^2),
- ω_{m0} is the rated angular velocity (rad/s),
- S_B is the rated apparent power of the generator (VA).

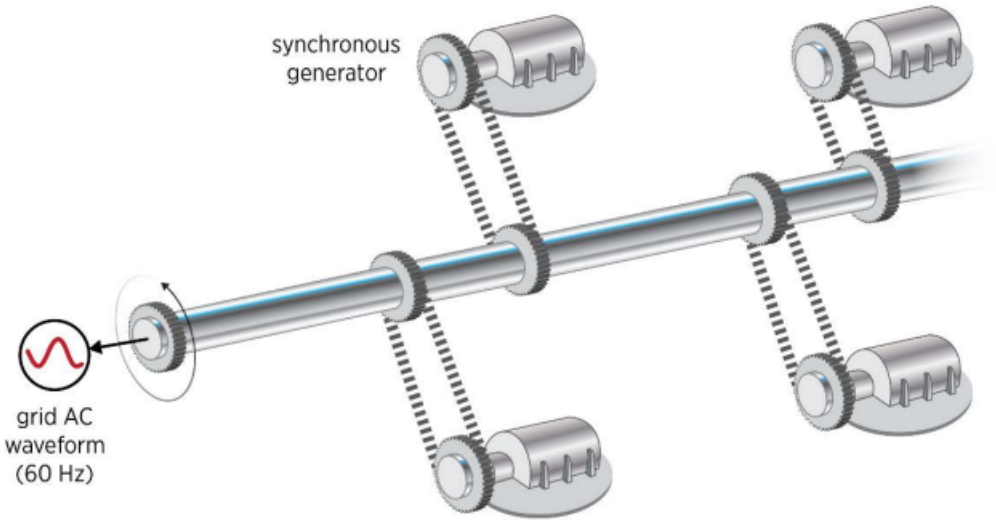

Figure 2.2: Generators working synchronized on the power grid [2].

2.3 System Inertia

In an interconnected power system, each area consists of a set of synchronous generating units, which possess rotating masses and thus contribute to the system's inertia. The total inertia of an area depends on the contribution of all conventional units participating in generation.

Specifically, when n synchronous generators operate within the same area, the total inertia H_{total} is calculated as the weighted average of their inertia constants based on their rated power [1]:

$$H_{\text{total}} = \frac{\sum_{i=1}^{n} H_i \cdot S_{G,i}}{\sum_{i=1}^{n} S_{G,i}} \tag{2.2}$$

Where:

- H_i is the inertia constant of the i-th generator (in seconds),
- $S_{G,i}$ is the rated apparent power of the generator (in MVA or p.u.).

The above formula demonstrates that the inertia of an area depends on the generating units that are connected to the power system at any given time. Thus, the total inertia varies dynamically according to the generation mix.

During periods of high demand, the activation of many conventional generating units can enhance inertia levels and, consequently, the frequency stability of the system. On the other hand, during low-load hours, the total inertia may be significantly reduced, especially when the demand is primarily covered by Renewable Energy Source (RES) units, which do not provide natural inertia. The estimation of inertia per area thus becomes particularly important in modern power systems, as many areas are dominated (or have a significant share) by RES units during specific hours of the day, significantly impacting the dynamic behavior of the grid's frequency.

2.4 The Role of Inertia in a Power System

The inertia levels of an area play a crucial role in the safe and reliable operation of a power system. During an imbalance between generation and demand, fluctuations in the system frequency occur. According to the security regulations of European Transmission System Operators [7], the frequency must be maintained within the range of 49.5 Hz to 50.5 Hz. If, for any reason, the frequency exceeds these limits, the impacts on the power grid can be severe or even catastrophic [2].

When a sudden disturbance in power demand or generation manifests, a series of control mechanisms are activated to gradually restore the frequency to its nominal value of 50 Hz, as it is indicated in Figure 2.3, which depicts a scenario of a step increase in power demand.

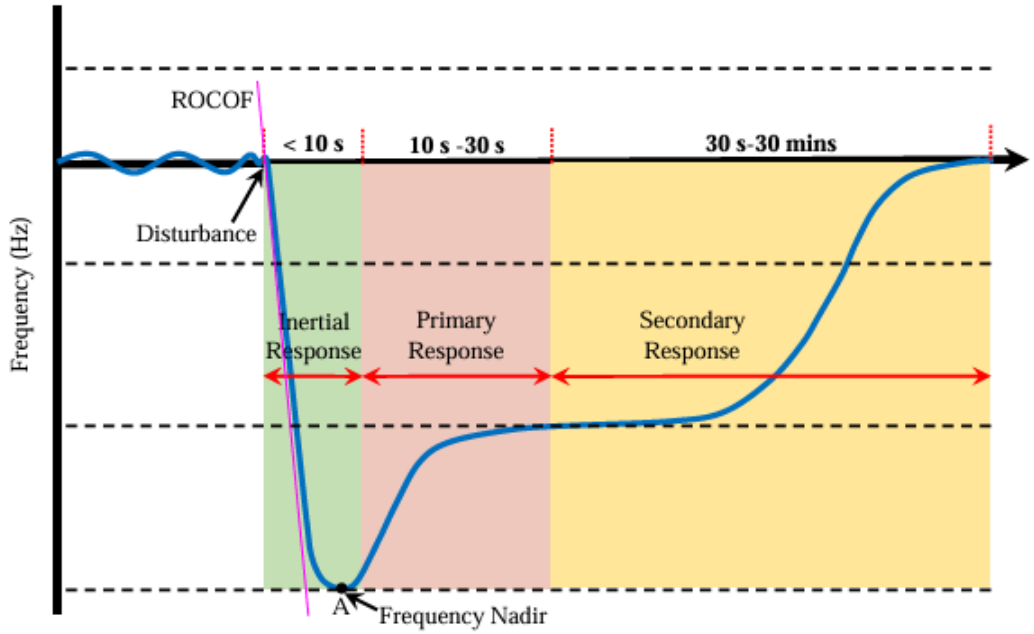

Figure 2.3: Frequency response at various time instances during a step increase in power demand [1].

As shown in Figure 2.3, immediately following the disturbance, the inertial response takes place. During this period, part of the kinetic energy of the rotating masses is converted into electrical power, temporarily contributing to covering the power deficit and reducing the frequency nadir, as well as the Rate of Change of Frequency (RoCoF). Similarly, in the case of excess generated power, the surplus energy is converted into kinetic energy, increasing the angular velocity of the machine rotors.

A few seconds after the disturbance, the primary frequency response is activated, typically lasting from 10 to 30 s. In this phase, the governor of the synchronous generators takes action by adjusting the mechanical input power to compensate for the difference between generation and demand. Specifically, in the event of a power deficit, the governor increases the opening of the steam (or fuel) inlet valve, while in the case of excess power, it decreases it, thereby accordingly altering the turbine's torque. In this way, a new temporary balance between generation and consumption is achieved, and the frequency stabilizes at a new value closer to the nominal one [4].

Finally, at a later time after the activation of the primary frequency control, the secondary frequency control is also activated. During this phase, using Proportional-Integral-Derivative (PID) controllers, appropriate generation signals are sent to the units so that the frequency returns to its nominal value [4].

The contribution of the inertial response depends on the number and size of the synchronous generators connected to the power grid at any given time. When a disturbance occurs (e.g., a sudden load change or loss of a generating unit), a torque imbalance is created on the rotor of each generator, leading to its acceleration or deceleration. As a result, the angular velocity and, consequently, the system frequency change [2].

The dynamic behaviour of each synchronous generator in a power system consisting of N_{SG} rotating machines can be described by the swing equation [1]:

$$2H_{SG,i} \cdot \frac{d\Delta\omega_{g,i}}{dt} = \Delta P_{m,i} - \Delta P_{e,i} - D_i \cdot \Delta\omega_{g,i} \tag{2.3}$$

Where:

- $H_{SG,i}$ is the inertia constant of generator i (in seconds),
- $\Delta\omega_{g,i}$ is the change in angular velocity of generator i (rad/s),
- D_i is the damping coefficient of generator i (in p.u.),
- $\Delta P_{m,i}$ is the change in mechanical input power of generator i (in p.u.),
- $\Delta P_{e,i}$ is the change in electrical output power of generator i (in p.u.).

Based on the type of synchronous generator , the value of the inertia constant ranges between 1.75 s and 10 s, as shown in Table A.1.

Table 2.1: Typical inertia constant values for various types of synchronous generators [4]

Generator Type	**Inertia Constant (s)**
Hydro Generator	1.75 – 4.75
Nuclear Generator	4
Thermal Generator (Steam Turbine)	2 – 10

In a power system consisting of multiple synchronous generators, a common rotational frequency can be defined, known as the **centre of inertia frequency** (ω_{sys}). This quantity corresponds to the weighted average angular velocity of all the system's generators, where the weight of each generator is given by its inertia constant. In this way, all units can be aggregated to an equivalent single-machine model, making the dynamic analysis of the system more efficient [1].

This conclusion above is of pivotal importance for the present work, as the development of the linear Simulink models in the following sections, which will simulate the generators' dynamic behavior, will be based on this specific consideration; all generators of a common type are grouped together and represented as a single model. With this approach, the effective management of the entire network's behaviour becomes possible, since instead of handling hundreds of individual generators, the analysis is limited to 3 or 4 groups (families) of generators per area (see Figure 2.4).

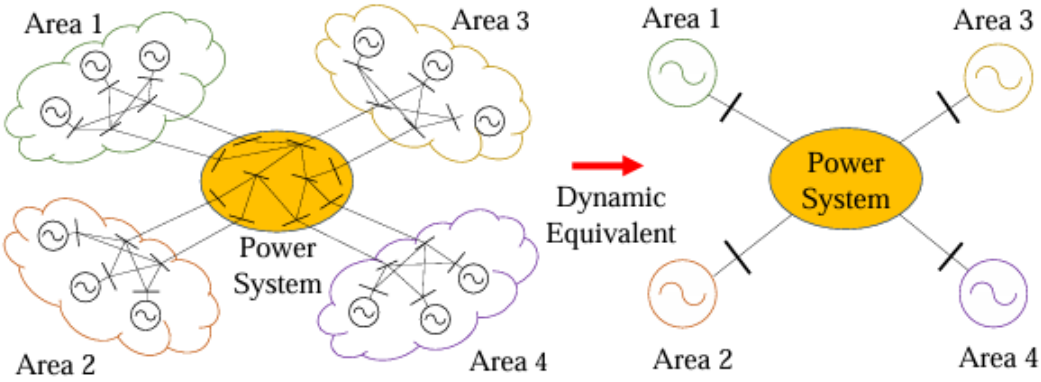

Figure 2.4: Dynamic equivalent representation of a multi-area power system [1].

The swing equation is reformulated for this equivalent system, taking into account the total system inertia H_{sys}, the mechanical and electrical power deviation ($\Delta P_m, \Delta P_e$), and the system damping coefficient D_{sys}. Its final form is shown in the following equation [1]:

$$2H_{\text{sys}} \cdot \frac{d\Delta\omega_{\text{sys}}}{dt} = \Delta P_m - \Delta P_e - D_{\text{sys}} \cdot \Delta\omega_{\text{sys}} \tag{2.4}$$

Generally speaking, the total inertia constant of a system tends to increase as the rated power of its constituent generators is getting larger (equation 2.2), which is however,not necessarily true for an individual generator.

For an individual unit, according to equation (2.1), an increase in rated power typically implies an increase in the moment of inertia J, due to the larger dimensions and mass of the rotor. However, because the increase in rated power (S_n) appears in the denominator of the fraction, the relationship between the rated power and the inertia constant of an individual generator is not directly proportional and is determined by the design of each machine, making it difficult to predict and compare across different units.The larger the inertia of a unit, the greater its ability to resist abrupt power changes. Additionally, it can supply or absorb

larger amounts of kinetic energy for a longer duration [2].

The inertia constant is measured in seconds (s) and indicates the time period for which the generator is capable of supplying its rated power to the grid, using exclusively its stored kinetic energy [2].

The kinetic energy of the rotor essentially functions as a reserve power reservoir: it is released to the system during a power deficit and absorbed during a power surplus. The greater the inertia of a generator, the larger the ”capacity” of this reservoir [**?**].

Chapter 3

Machine Learning Algorithms

3.1 Introduction

Inertia estimation is a particularly complex and time-consuming process, especially when attempting its analytical calculation through Equations 2.1 , 2.2, 2.3 and 2.4. This is due to the fact that the power grid comprises thousands of synchronous generators, which may differ significantly from each other in terms of their construction characteristics and power generation capability. Furthermore, the frequent lack of critical technical data and information regarding the system's generators makes it practically impossible to determine inertia in real-time using strictly mathematical models [8].

For this reason, the prevailing approach in practice has become the estimation of inertia through the analysis of the dynamic behaviour of the system taking measures such as the frequency and interconnection power . In this context, the contribution of machine learning and deep learning algorithms is crucial, as they enable the estimation of the grid's equivalent inertia with high precision [9, 10].

The scientific literature presents a plethora of algorithms for this problem. The choice of an appropriate algorithm depends on the nature and volume of the available data used for inertia estimation [10, 11, 12, 13, 14].

In the present thesis, we estimated inertia values using the following algorithms: Multi-Linear Regression (MLR), Random Forest, XGBoost, Support Vector Regression (SVR), and Multilayer Perceptron (MLP).In the current chapter the methods for data organization and management will be firstly analysed before we provide a detailed description of the operation and fundamental characteristics of the aforementioned algorithms.

3.2 Data Organization and Preprocessing

It is deemed essential that, before we proceed to an extensive description of complex and often intricate algorithms, the reader has a deep understanding of the fundamental concepts and basic operating principle of a machine learning algorithm. Generally speaking, all machine learning and deep learning techniques are inextricably linked to core fields of mathematics, such as linear algebra and numerical analysis, which focus, among other things, on solving systems of equations by leveraging the properties of matrices. The contribution of these mathematical tools to scientific fields related to data analysis, such as speech processing, computer vision, and the development of predictive models, is immense. For this exact reason it is particularly important to have a comprehensive overview of how data is converted to matrix form, as well as the type of processing it undergoes before being used as inputs to a model that will find the output values.

3.2.1 Input Data Structure

The output of a prediction model is generated each time as the result of a set of factors, each contributing to a different degree in determining the desired final value. In the literature, these factors are referred to as features, and they constitute the initial form of the input that is fed into the model.

Most datasets published by companies, government agencies, sports clubs, or health organizations can be represented as a matrix X of dimensions $a \times b$. In this form, each column $(x_1, x_2, x_3, \ldots, x_b)$ corresponds to a specific characteristic quantity (e.g., inflation rate, temperature, standard deviation, wind speed,humidity indicator etc.), while each row contains a complete record of all the features required to determine a corresponding output.

The degree of influence on the final output value varies from feature to feature. Some features are particularly critical for the model's success, while others may have a negligible or even minimal contribution. For example, in a classification model aimed at predicting whether a company will go bankrupt, a feature such as the company's annual revenue is likely to play a decisive role in the prediction. In contrast, other features, such as the number of employees or the date of establishment, may be of insignificant importance [15].

3.2.2 Performance Factors in Regression and Classification Problems

There are two major factors that ensure high prediction accuracy in regression and classification problems: the selection of an appropriate model and the correct preprocessing of the data.

Regarding the selection of the appropriate model, it must be capable of understanding the nature and behaviour of the input data in relation to the output, in order to achieve a proper generalization of the problem. In the majority of applications, it is common practice to apply multiple machine learning algorithms to estimate the target value, as the suitability of each method depends heavily on the structure, distribution, and complexity of the data. Consequently, it is not feasible to know from the beginning which algorithm will perform best or understand the behaviour of the data most effectively ([16]).

Most prediction models are pre-defined and ready for the analyst to use via available Python libraries, with the primary exception being neural networks. In the latter case, customized design of the network's architecture is often required, such as selecting the number and type of layers (e.g., dense layers), the number of neurons, the activation functions, and other parameters.

The second, and potentially more important, component for a model's success is the correct preprocessing of the data. Preprocessing constitutes a series of procedures applied to the initial data in the form of matrix X, before they are fed into the selected models. The goal of these procedures is the transformation and cleaning of the existing dataset to make it suitable for training and prediction.

3.2.3 Data Preprocessing Steps

The first step in data preprocessing is checking for the presence of missing values. If the percentage of missing values in a column is exceptionally high, that specific column may be removed entirely from the matrix X, as its contribution to the prediction can be considered negligible. Otherwise, the missing values are usually replaced with statistical values such as the mean or median of the corresponding column, depending on the nature of the distribution [15].

This is followed by the detection and handling of outliers: values that deviate significantly from the rest of the dataset. Their presence can negatively impact the model's performance, especially in algorithms based on distances or normality assumptions. Common detection

methods include the Z-score and IQR (Interquartile Range) analysis. Depending on the case, outliers are either modified (clipping), discarded, or replaced with statistically acceptable values [15].

Another extremely important procedure applied to data during preprocessing is the conversion of columns containing categorical data such as a product's country of origin (e.g., Greece, Germany, Italy) into numerical values. This is necessary because the majority of machine learning algorithms are unable to handle this type of data directly. To address this issue, techniques like one-hot encoding are used, whereby a new binary column is created for each distinct category, with values of 0 or 1 indicating the absence or presence of that specific category, respectively. In the example above, the `country` column would be replaced by three new columns: `country_Greece`, `country_Germany`, and `country_Italy` [15].

A critical stage of preprocessing is the scaling of numerical features. This process ensures that the data in each column are within a specified range, often between $[-1, 1]$ or $[0, 1]$, without distorting their relative ordering. The most widespread technique is standard scaling (or standardization), where the data are transformed to have a mean of zero and a standard deviation of one. This procedure is essential for the proper functioning of most machine learning and deep learning algorithms [15].

The final stage of preprocessing could be none other than feature selection. At this stage, the already modified dataset, as shaped by the previous steps, is reduced in size and complexity by discarding features (columns) whose contribution to the model's output is limited or negligible. This process is important for improving model efficiency, reducing computational cost, and avoiding overfitting. A multitude of techniques exist for evaluating the importance of each feature relative to the predicted output, such as methods based on statistical measures (ANOVA, χ^2), methods embedded within models (feature importances from Random Forests, Lasso), or iterative selection methodologies (Recursive Feature Elimination). However, a detailed presentation of these techniques falls outside the scope of this thesis. The key steps of data preprocessing are illustrated in Figure 3.1.

It is important to clarify, however, that the order and number of these specific preprocessing procedures vary from problem to problem. Some datasets require extremely thorough and careful preprocessing, while in others, certain steps may be omitted entirely or applied in a different order. Ultimately, in the world of data, no process is entirely standardized but always depends on the nature of the data itself [17].

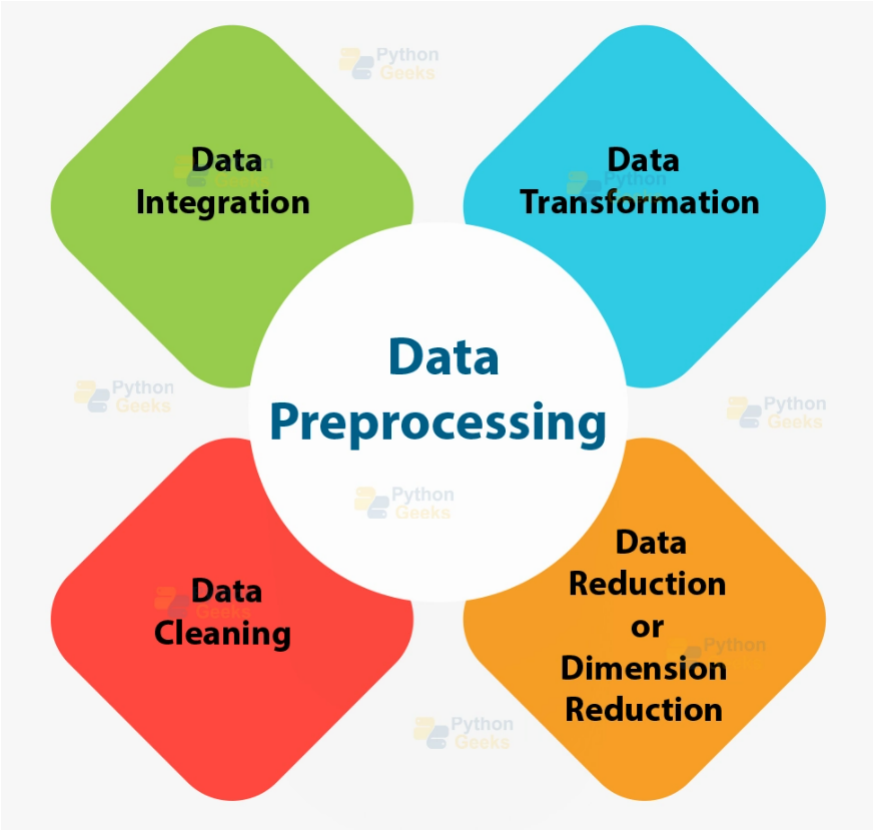

Figure 3.1: Stages of data preprocessing (Data Preprocessing in Machine Learning by PythonGeeks Team)

There are numerous ways to implement and train machine learning models for the estimation of a dependent variable. These methods depend on both the form and volume of the input data, as well as the level of accuracy desired for each application. In critical cases, such as medical diagnosis or predicting failures in industrial systems, exceptionally high accuracy is required. This requirement is usually met by using models of increased complexity, such as neural networks, which, however, demand significant computational power and longer training times. On the other hand, when a problem has a higher error tolerance or resources are limited, it is common practice to sacrifice a degree of accuracy for the benefit of faster training and lower computational demands. In such cases, simpler models like linear regression or decision trees are preferred.

The goal in every case remains the same: the model must be capable of understanding the relationships between the input variables (features) and the output, and during its training, it must construct a predictive function. This function must adequately capture the correlation between the independent and dependent variables so that the model can generalize to unseen input values it has not encountered in the training data [18].

3.2.4 Training and Evaluation Data Split

The initial dataset, after having undergone the preprocessing procedure, is typically divided into two subsets: the training dataset and the testing dataset.

As a general rule, the training set comprises approximately 80% of the original dataset and is used to train the model in order to teach it how to connect the various input features with the corresponding output values. Accordingly, the testing set contains the remaining 20% and is used to evaluate the model's ability to generalize, meaning to correctly predict output values for input data it has not "seen" during training.It is important, however, to clarify that the splitting of samples into training and testing sets is usually done randomly (random split) from the entire dataset, and not sequentially.The splitting does not imply that the first 8 out of 10 samples will be included in the training set and the last 2 in the testing set [15]. Such an approach could introduce temporal or structural bias. A key exception to this rule is time series forecasting, where the temporal sequence of the data is of critical importance, and the prediction of the most recent temporal values is the objective. In this case, the split is made sequentially in time to ensure that the model predicts the future based exclusively on the past.

After the horizontal split of the original dataset into training and testing subsets, a second, this time vertical, split takes place. The goal of this process is to separate the input variables from the output variables in order to distinguish the features from the target values [15].

In this way, we arrive at the creation of four distinct sub-matrices, which are derived from the original data matrix. According to the literature, the matrix of input variables is usually denoted by X, while the matrix of output variables is denoted by y. Consequently, the four subsets created are: $X_{train}, y_{train}, X_{test}, y_{test}$. This separation allows for the independent processing of inputs and outputs and constitutes a standard practice in every supervised learning problem [15].

The idea behind the majority of machine and deep learning algorithms is based on initially adopting a mathematical function with randomly initialized parameters (e.g., weights and biases) and gradually modifying these parameters through an iterative process (epochs), aiming to reduce the error between the actual and predicted values.

Being more accurate, during the first iteration, the model begins with a predictive function that connects the input variables (X_{train}) with the corresponding output values (y_{train}) using randomly chosen parameters. The resulting estimates are recorded in a new prediction vector, usually denoted as y_{pred}, which is updated in each iteration. Due to the random initialization,

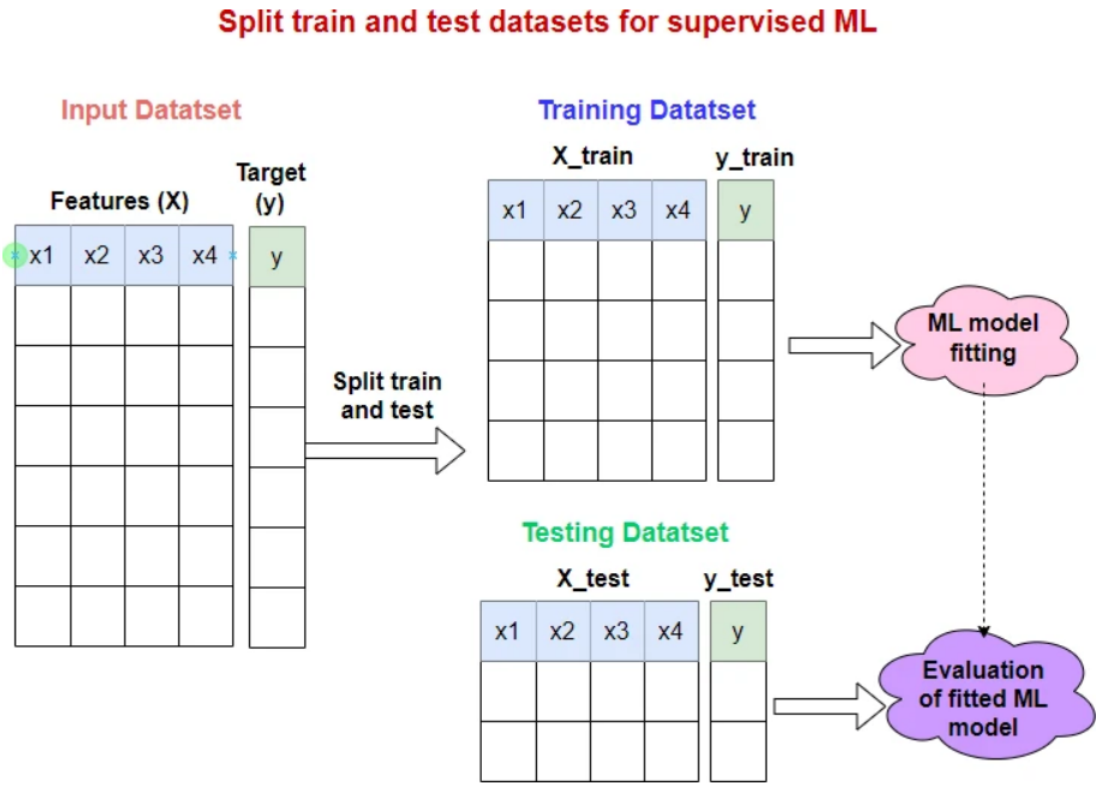

Figure 3.2: Splitting of the original dataset into sub-matrices for training and testing How to Split Data into Train and Test Sets in Python with sklearn

the predicted values in the first iteration deviate significantly from the actual values.

After the completion of each iteration, the model's parameters are updated via optimization algorithms, such as Gradient Descent, with the goal of gradually reducing the error. This process is repeated for several iterations until the error between y_{pred} and y_{train} is deemed sufficiently small. At that point, the model is considered to have learned the input-output relationship and is ready to be applied to new, unseen data whose output values are not known in the beginning [15].

Having now completed the training process of the respective model, the evaluation stage follows, during which it is examined to what extent the model is capable of accurately predicting output values for input data it has never "seen" before. The goal is to confirm that the model has not simply memorized the values of y_{train}, but instead has learned the underlying input-output relationship and can now generalize. It is at this exact stage that the use of the testing dataset, mentioned in the previous section, becomes essential. If the final form of the mathematical function developed during training indeed adequately describes the relationship between the input and output variables, then the model is expected to accurately predict

the output values y_{pred}, using the values of the X_{test} matrix as input [15].

The question that arises at this stage is the following: how do we know if the output values produced by the model are actually correct? The answer lies in the y_{test} vector, which contains the true output values for the testing dataset. Although the model does not have access to y_{test} during training or prediction, these values are known to the user. Thus, by comparing the predicted values y_{pred}, as calculated by the trained model based on X_{test}, with the corresponding true values from y_{test}, we can estimate the accuracy of the predictions. The difference between these two vectors (prediction error) constitutes the fundamental metric for evaluating the model's performance. Everything discussed in this paragraph is summarized in Figure 3.2.

3.2.5 Evaluation Metrics

According to the type of problem (regression or classification), appropriate metrics are used to quantify the error. In most regression problems, as is the case in the present thesis, four such metrics are used. Those are the Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), and the Coefficient of Determination (R^2).

Description of Regression Evaluation Metrics

- **Mean Absolute Error (MAE)** It expresses the average absolute difference between the predicted and the actual values, without considering the sign of the difference. Although it is one of the simplest metrics to interpret, it is one of the most widely used methods for error evaluation in regression problems. The value of this metric is given by the following equation [19]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  where n is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

- **Root Mean Squared Error (RMSE)**

  This metric measures the average deviation between the actual and predicted values. It is calculated as the square root of the average of the squared differences between

corresponding value pairs. Due to the squaring, RMSE gives greater weight to larger errors. The smaller its value, the better the model approximates the actual output points. Its value is given by the following mathematical formulation [20]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where n is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

- **Mean Absolute Percentage Error (MAPE)**

  It expresses the error as a percentage of the actual value [21]:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

  where n is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value. The smaller the MAPE value, the more accurate the model is considered.

  A significant advantage of MAPE is that it provides an intuitive, scale-independent measure of error. Unlike metrics such as MAE or RMSE, whose values can only be judged in the context of the data's range, MAPE gives a direct percentage, making it immediately clear how large the error is relative to the true value.

  However, MAPE has a notable drawback: it can become unstable and tend towards infinity when the actual values y_i are close to zero, as this causes a division by a very small number. Consequently, its use is not recommended for datasets with output values near zero.

- **Coefficient of Determination (R^2)**

  The coefficient of determination R^2 measures the percentage of the variance in the dependent variable that is explained by the model. If the value of R^2 is close to 1, it means that the model satisfactorily describes the variance of the output. Conversely, values close to 0 indicate that the model fails to account for the variability in the data. This index is calculated by the following formula [22]:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

  where n is the number of observations, y_i is the actual value, $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean of the actual values.

3.3 Algorithms for Inertia Estimation

As we mentioned earlier, a total of five algorithms were used to estimate the inertia constants of each interconnected area. The operating principle of each algorithm is analyzed in detail in the following subsections.

3.3.1 Multiple Linear Regression

Multiple Linear Regression (MLR), is a supervised machine learning algorithm that tries to find the system's output as a linear combination of its inputs [15]. Consequently, the relationship between the dependent variable and the various independent variables is considered linear. The mathematical description that links the system's output, which in our case corresponds to the inertia constants, with the various input feature values is given by the following relation:

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_b x_b \tag{3.1}$$

Where the terms $x_1, x_2, \ldots, x_b$ are the numerical contents of the input feature columns with dimensions $n \times 1$, while the variables $w_1, w_2, \ldots, w_b$ are the coefficients or, otherwise, the weights of these inputs and, in practice, indicate the degree of influence of each input on the final output. Finally, the term w_0 is a constant (intercept) that aims to further improve the final result.

The objective is the accurate calculation of the weights ($w_0, w_1, w_2, \ldots, w_b$) with the goal of minimizing the sum of the squared errors between the actual value and the value predicted by the model, known as the Sum of Squared Errors (SSE) [5]. The SSE is given by the following relation:

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}i)^2 = \sum i = 1^n \left[y_i - (w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_b x_{ib})\right]^2 \tag{3.2}$$

This specific machine learning algorithm follows the following logic:

We begin with the random initialization of the weight values $w_0, w_1, w_2, \ldots, w_b$ and calculate the set of outputs y with dimensions $n \times 1$. It is almost certain, however, that the resulting values of the matrix y will differ significantly from the actual output values (y_{train}), due to the arbitrary selection of the initial weights. Consequently, a single iteration of the

process is not sufficient; instead, a large number of repeated steps is required until the output values are computed with satisfactory accuracy.

The big question is how our model can know whether the values it has predicted are valid or deviate significantly from reality. The answer lies in Equation (3.2), where in each iteration a new value for the SSE variable is calculated, which serves as an indicator of the deviation between the actual and predicted values. The smaller the SSE value, the better the approximation achieved by the model.

There is a very large number of methods utilized for the estimation of the parameters $w_0, w_1, w_2, \ldots, w_b$ in order to minimize the SSE achieving the smallest possible error.One of the most popular techniques is the Gradient Descent [23].

According to this specific optimization algorithm, in each step the weight values are updated by the following relation:

$$w_j \leftarrow w_j - \gamma \frac{\partial, SSE(w_0, w_1, \ldots, w_b)}{\partial w_j} \tag{3.3}$$

where the term $\gamma > 0$ is known as the *learning rate* and indicates the rate at which the weight values are updated in each step. The weight values $w_0, w_1, \ldots, w_b$ are updated in each iteration and are reused in the computation of the final output values. If the approximation of the final value is satisfactory ,meaning that is smaller than a maximum permissible error or if no further improvement is observed after a predetermined number of iterations, the process stops and Equation (3.1) takes its final form.

3.3.2 Random Forest

A significant drawback of the previous algorithm is its inability to accurately approximate the output in problems where the relationship between the input and output variables is not linear. For this reason, over time, a series of other algorithms were developed to solve problems of this nature. A very powerful and quite unique algorithm widely used for solving regression problems is Random Forest, where the final output value derives from the average of all outcomes obtained from an ensemble of decision trees [18].

To interpret the *Random Forest* algorithm, it is necessary first to analyse the operational principles of its building blocks;the Decision Trees. According to Figure 3.3, a decision tree consists of a set of nodes and edges. The first node, located at the top of the tree, is called the *root*, while the nodes at the bottom are called *leaves*. The length of the longest path, i.e., the

number of edges from the root to the most distant leaf, is defined as the *height* of the tree. The *depth* of a node, on the other hand, is defined as the number of edges from that node to the root of the tree. Consequently, when we refer to the depth of the entire tree, we calculate the depth of its leaf that is farthest from the root. By this logic, the depth and height of a tree are conceptually identical. Another important piece of information is that two nodes located at consecutive levels of the tree and connected to each other have a parent-child relationship. In regression and classification problems, however, unlike in Figure 3.3, only fully binary trees are met;Both the root and the internal nodes have exactly two children. The estimation of the final value y follows the logic below:

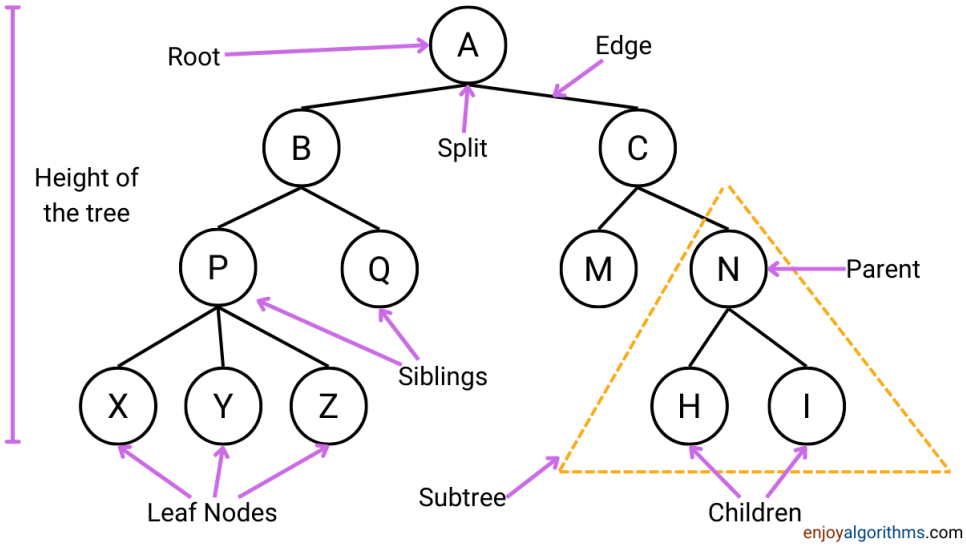

Figure 3.3: A representation of a tree in computer science. [Decision Tree: A Tree-based Algorithm in Machine Learning]

An initial dataset (*training dataset*) waits temporarily at the root of the tree. Each node of the decision tree, except for the leaves, contains a condition that separates the data arriving at it.Typically, this condition is an inequality relation, since in our case we have continuous variables to process and predict (example: $x_1 \geq 3$). Thus, each node functions as a passage point or, even better, as a data junction, where the path followed by each observation is determined by the values of its features [18].

After the initial *training dataset* is split at the root of the tree, two new data subsets will be found at its children, ready to encounter new inequality conditions and be divided further. The general rule for splitting is that if the condition is satisfied, the respective tuple (a record of the *training dataset*) will be directed to the left child of the node containing the

condition; otherwise, it will proceed to the right child. This process continues for all tuples in the training set until they reach their final destination:the leafs . The depth of each decision tree is determined by the data analyst according to the needs of the problem.

But how are the conditions encountered at each node formulated to achieve the optimal partitioning of the data?

One of the many ways a dataset, regardless of its size, can be described is through its *variance*, which is a measure of the disorder and randomness of the data that constitute it. The condition considered suitable is the one that, at each splitting step, induces the largest possible reduction in variance between the parent and its children, as the goal is to group tuples with common characteristics. The mathematical relation that allows us to calculate the variance of a dataset is given by Equation (3.4):

$$Var = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{3.4}$$

At each splitting stage during training, the variance of all possible datasets that can result from every possible split is calculated. For example, one might examine the two subsets resulting from the condition $x_1 \geq 21$ or the two subsets resulting from the condition $x_4 < -3$.

Therefore, for each splitting scenario, the difference in variance between the parent and the children is calculated. This difference is given by the following equation:

$$Var_{\text{reduction}} = Var(\text{parent}) - \sum_{i} w_i, Var(\text{child}_i) \tag{3.5}$$

where the term w_i denotes the percentage of the initial dataset inherited by each child.

This process is repeated for all possible splitting scenarios in all nodes of the tree until we reach the leaves. At this point, the tree is ready to receive the *testing dataset*. The elements of the *testing dataset*, for which the model does not know the output, enter the tree sequentially and follow the path dictated by the conditions of each node until they end up at a leaf. At that point, the final output value is determined as the average of the outputs of the training tuples that belong to that specific leaf.

In the case of the *Random Forest*, the above training and prediction process is applied to an ensemble of many decision trees, which operate in parallel. Each tree is trained on a different sample of the same initial training set, obtained through sampling with replacement (*bootstrapping*). Thus, although all trees ”see” data from the same original source, the composition of the training set for each tree differs,increasing the diversity among them. After

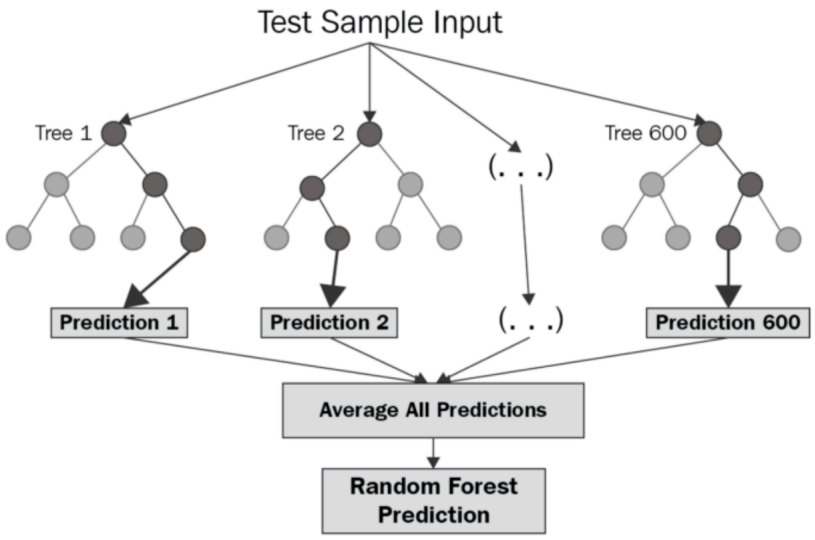

Figure 3.4: Representation of the Random Forest algorithm's operation. [Random Forest]

training, each tree in the *Random Forest* receives the *testing dataset* and produces its own prediction. In the regression problem, the final output of the model is derived as the average of the predictions of all the trees, as shown in Figure 3.4.

Although this algorithm achieves high accuracy in its predictions, it is characterized by high computational cost due to the large number of operations and comparisons required for the training and operation of numerous trees. Additionally, to ensure the best possible results, the appropriate determination of input parameters, such as the maximum depth of the trees and their total number, is essential. However, despite the computational cost, this technique remains one of the most popular methods for regression problems. From a mathematical perspective, the output predicted by the model will be given by Equation (3.6).

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f_t(x) \tag{3.6}$$

Where T is the number of trees in the forest and $f_t(x)$ is the prediction of each individual tree.

3.3.3 Support Vector Machines

The *Support Vector Regression* (SVR) algorithm is another particularly valuable tool in our collection for estimating the inertia of a power system. It stands out for its flexible and efficient handling of non-linear datasets, utilizing a *kernel* to project the data into higher-dimensional spaces where the relationships between variables become linearly separable [24].

Although it often delivers excellent performance in both regression and classification problems, as well as effectively handles outliers, in most cases it requires particularly careful preprocessing and correct selection of hyper-parameters to produce reliable and stable results, thereby increasing the required computational cost.

The fundamental idea of SVR is to find a function $f(\mathbf{x})$ that estimates the target values y_i with an error smaller than a predetermined tolerance ϵ, while simultaneously ensuring the minimum possible complexity. The general form of the model is:

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \tag{3.7}$$

where $\phi(\mathbf{x})$ is a transformation of the inputs into a higher-dimensional space, $\mathbf{w}$ is the weight vector, and b is the bias term.

The goal is the minimization of the following cost function [24]:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{3.8}$$

subject to the constraints:

$$\begin{cases} y_i - f(\mathbf{x}_i) \leq \epsilon + \xi_i \\ f(\mathbf{x}_i) - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{3.9}$$

where C is a parameter that controls the trade-off between the model's complexity and the permissible errors, and ξ_i, ξ_i^* are slack variables that allow deviations larger than ϵ. A schematic representation of this approach in two-dimensional space is shown in Figure 3.5.

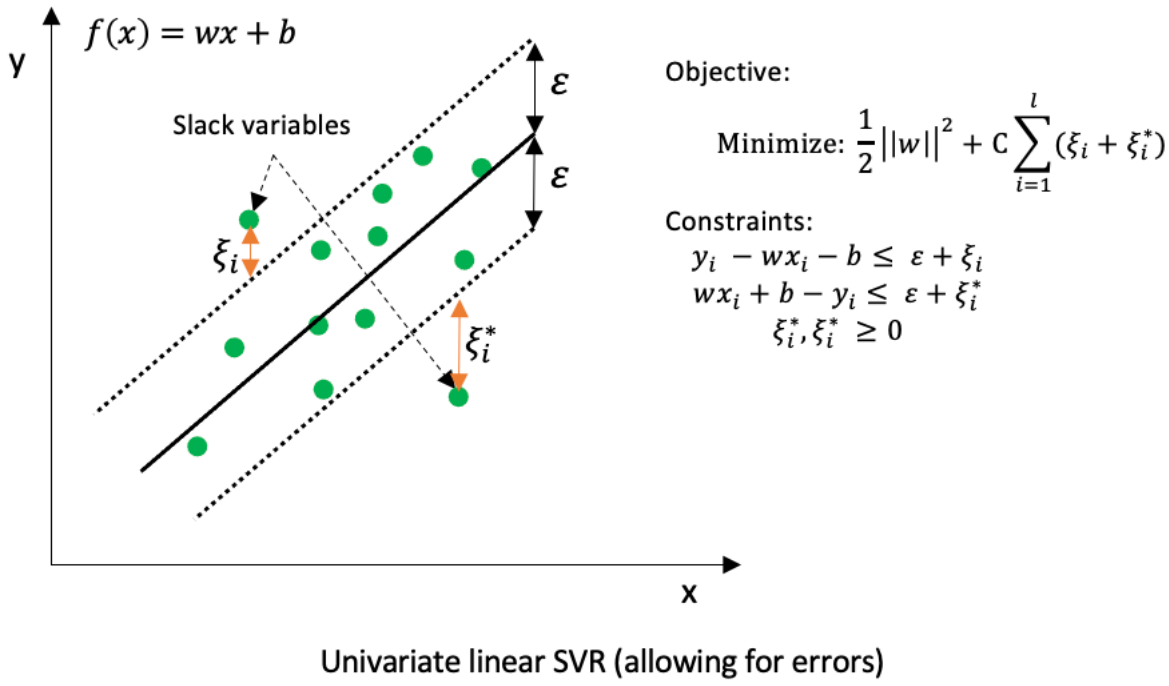

Figure 3.5: Representation of the SVR algorithm's operation in two-dimensional space. [From Theory to Practice: Implementing Support Vector Regression for Predictions in Python]

In simple terms, SVR tries to find a line or in our case, due to the number of input variables, a hyperplane that passes "close" to the data, ignoring small deviations (the ϵ-insensitive tube) and penalizing large ones, so that the model generalizes well to new data. However, as the number of independent variables increases, so does the number of dimensions. Therefore, with three variables and beyond, it becomes completely impossible for us to schematically represent both the hyperplane and the output values of the data it will approximate.

3.3.4 EXtreme Gradient Boosting

XGBoost (*Extreme Gradient Boosting*) is an exceptionally powerful machine learning algorithm based on decision trees, just like *Random Forest*. Its operation relies on the technique of *gradient boosting* [25], thereby ensuring impressive prediction accuracy in a short time. The fundamental idea is based on the construction of successive simple models (*weak learners*), typically small trees, which gradually improve upon the errors of the previous ones. Therefore, instead of trying to learn the prediction function $f(x)$ directly, XGBoost

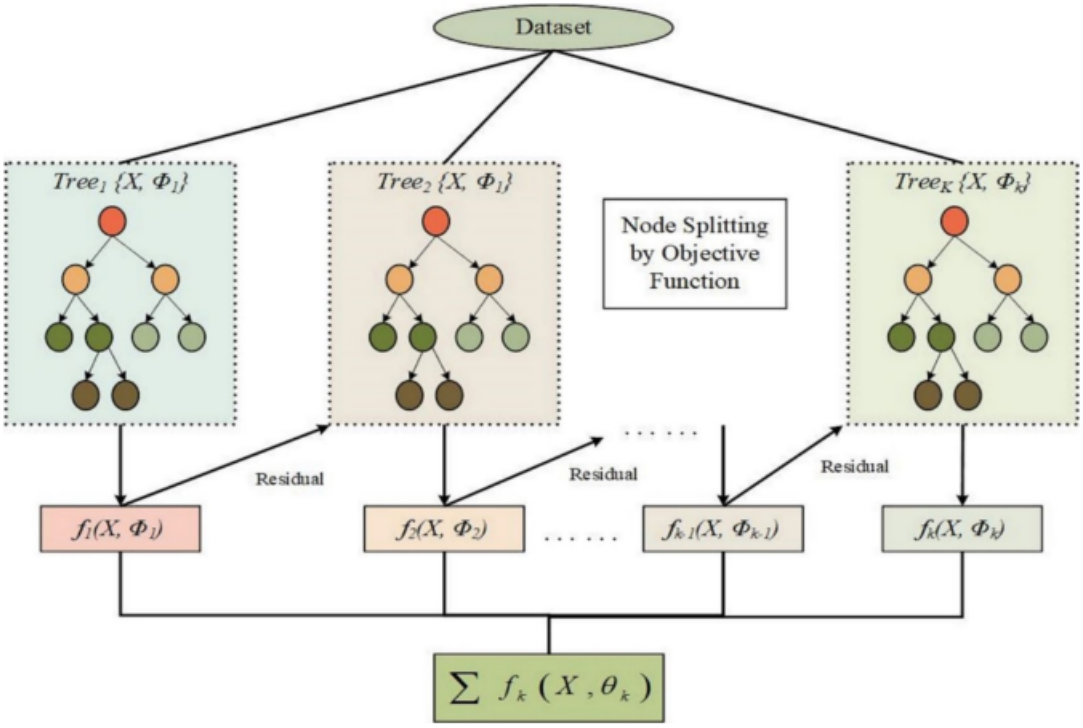

Figure 3.6: Representation of the XGBoost algorithm's operation [3]

constructs a sum of K simple functions (trees):

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F}$$

where $\mathcal{F}$ is the set of all possible decision trees.

As mentioned earlier, the training is performed step-by-step. When we are at the t-th step, having the model $\hat{y}_i^{(t-1)}$, we add a new tree f_t that reduces the error:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)$$

The objective, as with most machine learning algorithms, is the minimization of a cost function. In this case, XGBoost tries to minimize the following relation [26]:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{k=1}^{t} \Omega(f_k)$$

where l is the loss function (e.g., squared error) and $\Omega(f_k)$ is a regularization term that restricts the tree's complexity (e.g., depth and number of leaves). To find new trees f_t, a second-order Taylor expansion is performed around the predictions of the previous model:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \Omega(f_t)$$

where:

$$g_i = \frac{\partial l(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}, \quad h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(t-1)})}{\partial (\hat{y}_i^{(t-1)})^2}$$

are the first and second derivatives (gradient and hessian).

During the tree construction, for each candidate *split*, XGBoost calculates the *gain*:

$$\text{Gain} = \frac{1}{2}\left[\frac{\left(\sum_{i\in L} g_i\right)^2}{\sum_{i\in L} h_i + \lambda} + \frac{\left(\sum_{i\in R} g_i\right)^2}{\sum_{i\in R} h_i + \lambda} - \frac{\left(\sum_{i\in L\cup R} g_i\right)^2}{\sum_{i\in L\cup R} h_i + \lambda}\right] - \gamma$$

where L and R are the left and right children, the parameter λ controls the smoothness of the weights, and γ is the cost of creating a new leaf.

Taking everything into consideration we could say that the final output values of the matrix y result from the combined sum of the predictions of all decision trees. Starting from the first tree, an initial prediction of the y matrix values is produced, which is derived from the mean value of the column of the actual output values and, as expected, is quite far from reality. Subsequently, the difference between the actual and predicted value (*residual*) is calculated and is then fed as input to the immediately next tree. Then, the next tree calculates the values of a new matrix, which will be added to the initial one, aiming to reduce the error between the actual and predicted values. This process is repeated for a specific number of trees. In contrast to *Random Forest*, where the trees operate independently, in XGBoost each tree is constructed to correct the errors of the previous one, with the result that the final prediction is the cumulative outcome of all trees. The basic operating principle of this algorithm is illustrated in Figure 3.6.

3.3.5 Multi-layer Perceptron

The final model that was examined, and indeed, as we will see later, produced the most accurate predictions for the inertia values, is the Multi-Layer Perceptron (MLP) algorithm. This specific algorithm belongs to the family of neural networks, which constitute an extremely popular family of algorithms, known by the term deep learning models, a subcategory of Machine Learning algorithms. The spectrum of applications where neural networks are used is quite broad, extending from simple algorithms for estimating values to robotics, image processing, and natural language processing. This immense popularity is largely due to the ability of these algorithms to handle non-linear data effectively [5].

A common characteristic of all models belonging to the neural network family is the neurons. The latter, organized in layers, take information as input, apply a mathematical

transformation to it using activation functions, and pass the final result to the neurons of the subsequent layers.

The MLP model is the simplest form of a neural network and is structured from a specific number of *layers*, which are connected to each other via edges. Being more precise, we distinguish the input layer, one or more hidden layers, and, finally, the output layer, where the final prediction value is produced.

This specific neural network takes as input a vector consisting of numerical values, each of which corresponds to the value of a *feature* of a tuple in the dataset. Each layer consists of a specific number of neurons, with the *input layer* having exactly as many neurons as the input features of the network (i.e., the independent variables), while the *output layer* has as many neurons as the output variables.

The neurons,also known as nodes, in this specific model are fully interconnected (*fully connected layers*). This means that each node in the hidden layers receives as input the outputs of all neurons in the previous layer and, subsequently, passes the result of the output it produces to all nodes in the next layer. The neurons of two adjacent layers, as mentioned previously, are connected to each other via edges, each of which has its own *weight*. This weight is essentially a numerical coefficient that multiplies the output of one neuron before it is passed to the next. The value of each weight ($w_0, w_1, w_2, \cdots, w_b$) expresses the degree of influence of one neuron's output on the immediately next one, while the final estimation of their values is the key to constructing a functional and effective prediction model [5].

The training process begins with the input of the first tuple from the *training dataset* into the *input layer*. This input has the form of a column vector ($b \times 1$) and contains the numerical values of the input features for a single record. At this layer, each neuron "holds" one and only one value, which corresponds to each input feature. The neurons of the *input layer* do not impose any mathematical transformation on the data; they simply pass it to the next layer, i.e., to the *hidden layers*, where more complex mathematical computations are applied to the incoming information, utilizing non-linear activation functions.

After the data is passed to the first *hidden layer*, initially each node sums all the input data that originated from the *input layer* and have been multiplied by the corresponding weight coefficient of each edge. Once this summation is completed, the result is fed as an argument into an activation function, which tends to vary depending on the nature of the problem we are addressing. The most well-known and widely used activation functions, along with their

corresponding mathematical description, are shown in Table 3.1.

Table 3.1: Most popular activation functions [5]

Activation Function	**Mathematical expression** f
ReLU	$\mathrm{ReLU}(x) = \max(0, x)$
Sigmoid	$\sigma(x) = \dfrac{1}{1 + e^{-x}}$
tanh	$\tanh(x)$

Although it is possible to use different activation functions throughout the entirety of a neural network, its division into layers imposes the exclusive use of a single type of such function in each one of them.

After the activation function of each neuron produces the output value, it is forwarded to the next layer, which can be either another hidden layer or the output layer. The number of hidden layers, combined with the number of neurons belonging to them and the activation function used, constitute *hyperparameters* that must be taken into account during the model's design.

In mathematical terms, the above process is expressed by the following proposition: Suppose we have the input values $x_1, x_2, \ldots, x_n$, which are fed into a neuron j in layer h. The output of this neuron is calculated as:

$$z_j^{(h)} = \sum_{i=1}^{n} w_{ij}^{(h)} x_i + b_j^{(h)} \tag{3.10}$$

where $w_{ij}^{(h)}$ is the weight of the connection between the i-th neuron of the previous layer and the j-th neuron of layer h, while $b_j^{(h)}$ is the *bias* of the neuron. Subsequently, the activation function $f(\cdot)$ is applied to yield the final output:

$$a_j^{(h)} = f\left(z_j^{(h)}\right). \tag{3.11}$$

Figure 3.7 accurately depicts the practical application of Equations (3.10) and (3.11).

The same logic applies to all nodes belonging to the hidden layers of the neural network. Regarding the output layer, the data, having already undergone a large number of mathematical transformations in the *hidden layers*, enter one final time into a new layer of neurons. If it is a classification problem, the nodes will sum the values from the various edges, pass them through an activation function, and determine the final values. In regression problems,

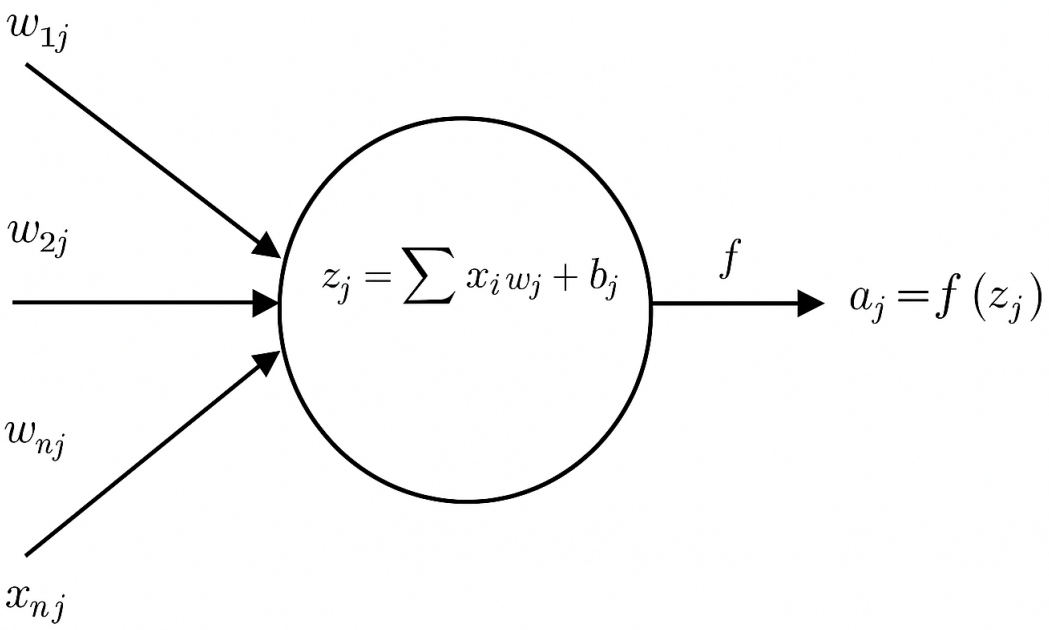

Figure 3.7: Operating principle of a hidden layer neuron

however, since the goal is the estimation of continuous values, the use of activation functions in the output layer is usually avoided. A representation of a fully connected neural network, similar to the one used in the context of this thesis, is shown in Figure 3.8.

The neurons in the *output layer* produce the final output values of the system. After the pass of the first tuple from the *training dataset* is completed, the second tuple follows, and the process repeats.

The fundamental question at this point is how the architecture of Figure 3.8 can accurately calculate the inertia values of a power system area.

The input data, after having previously undergone *pre-processing* to be more easily manageable and compatible with the neural network, are fed into it organized in groups called *batches*. The tuples of each batch enter the neural network sequentially and undergo the processes we described earlier. After the model's predictions are generated for all tuples in the first batch, a fundamental process for neural networks follows: *backpropagation*.

Initially, the weights of the edges are initialized randomly, and consequently, the predicted values resulting from the pass of the first *batch* may be quite far from reality as we mentioned before. After the first set of predictions is generated, their validity is assessed using various

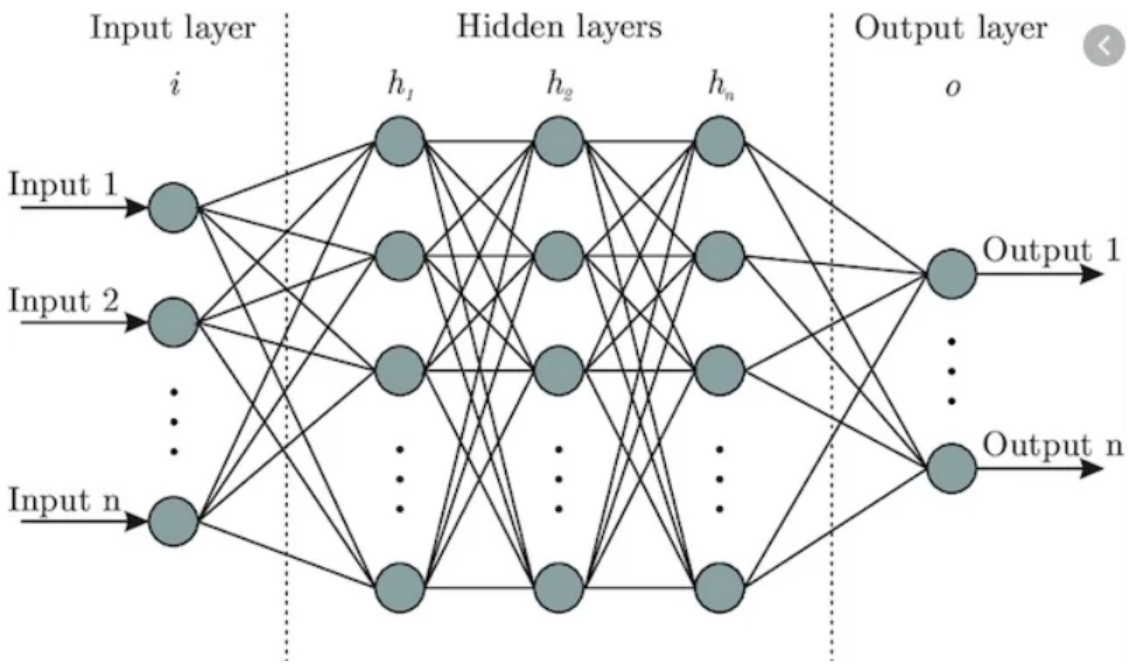

Figure 3.8: Fully connected neural network.

metrics, which typically check the difference between the actual and predicted values.

Subsequently, during *backpropagation*, the weight values are updated throughout the entire neural network with the goal of minimizing the error. The manner in which the new values are formed depends on the specific *optimizer* that has been selected (e.g., Adam, SGD).

After the pass of the first batch and the modification of the weights, the passage of the tuples from the second batch through the neural network follows. The number of records in each batch is usually a power of 2 (e.g., 32, 64, 128) and also constitutes a hyperparameter, which must be chosen carefully to ensure high prediction accuracy. This process is repeated for all batches of the *training dataset*, until all input data have passed through the neural network at least once. At that point, we say that one *epoch* has been completed.

The number of epochs is selected in such a way as to achieve the maximum possible accuracy. In practice, the model's performance is examined for various scenarios with a different number of epochs, and the optimal one is selected. After the completion of the total number of iterations, the weights have now been shaped, and the model is ready to test its capabilities on data unknown to it (*testing dataset*). If the error fluctuates within satisfactory limits, this means that the model has managed to formulate the appropriate function that connects the inputs to the outputs and possesses strong predictive ability.

Chapter 4

Frequency Response Models of Interconnected Power Systems

4.1 Introduction

This chapter presents a detailed simulation of the dynamic behavior of two interconnected control areas of a power system, using system frequency response models. Within the framework of this thesis, multiple scenarios with different combinations of generating units were developed in the Matlab-Simulink environment, aiming to investigate the system's dynamic response under various operating conditions. These scenarios cover a broad spectrum of linear models that approximate the behavior of different types of generating units, such as hydropower plants, steam turbines with and without reheat, while in subsequent sections, gas turbine plants and biomass units will also be examined.

4.2 Basic Block Diagram

The block diagram depicting the two interconnected control areas is presented in Figure 4.1. Observing the diagram, four fundamental components can be distinguished for each area:

- **Power System Transfer Function (Power System Component):** This is a transfer function whose output corresponds to the total frequency response of the area (Δf), expressed in p.u.

- **Disturbance Input (ΔP_d) :** Represents the change in power demand.
- **Controller:** A subsystem that, through the use of integral controllers, aims to restore the frequency to its nominal value. This essentially implements the secondary frequency regulation (Load Frequency Control) of each area.
- **Power Plants:** A subsystem consisting of a mix of various generating units, the composition of which varies depending on the scenario under examination.

The functionality and internal architecture of each of the aforementioned components will be analysed extensively in the following subsections.

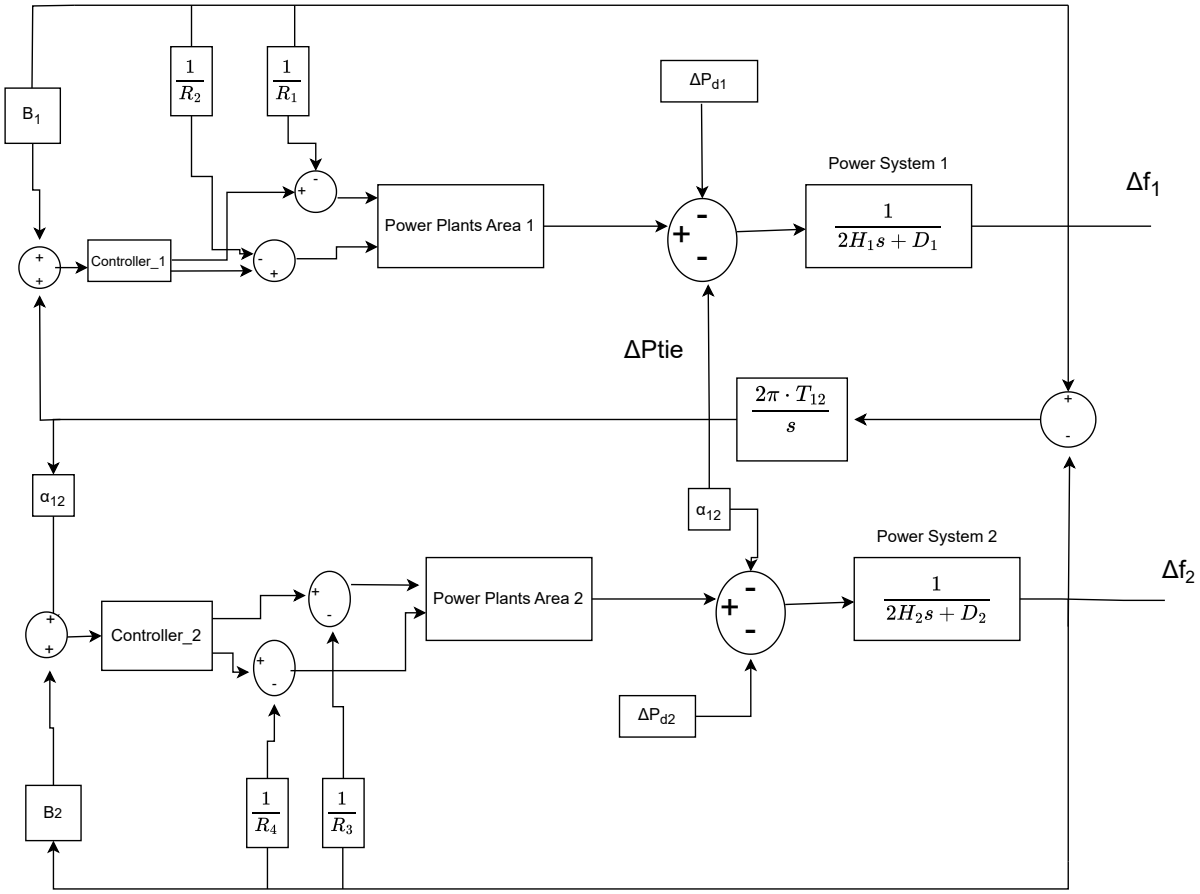

Figure 4.1: Diagram of two interconnected control areas.

4.2.1 Power System Transfer Function (Power System component)

This specific transfer function includes the total inertia constant of the respective area, H (in s), as well as the load-damping factor D. The corresponding loop for Area 1 is presented

with greater clarity in Figure 4.2.

It is quite obvious that both the value of the equivalent inertia constant H_{eq} of each area and that of the damping constant D significantly influence the area's dynamic response. The inertia constant has a greater impact on the system's frequency response and typically varies considerably throughout the day. In contrast, the damping coefficients D can be considered approximately constant and known. Therefore, assuming the values of D_1 and D_2 are known, the objective of this thesis is the accurate estimation of the individual inertia constants H_1 and H_2.

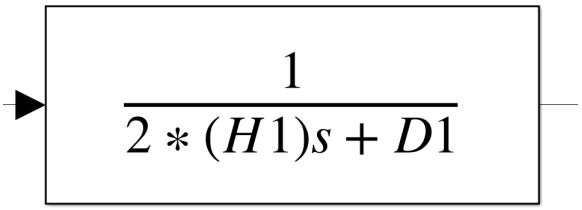

Figure 4.2: Transfer function of Power System

4.2.2 Disturbance Input ΔP_d

As mentioned above, ΔP_d expresses the change in power demand in p.u. If its value is negative, it means there is a power surplus on the grid; consequently, the frequency tends to increase and an appropriate signal must be sent to the governor to reduce the output power of the generating units. On the other hand, when the value of ΔP_d is positive, it indicates an increase in electrical energy demand, resulting in a drop in frequency, which requires an increase in generated power. In practice, the power demand is never constant but continuously fluctuates, exhibiting both positive and negative deviations. These fluctuations are mainly due to the stochastic behavior of loads, such as the switching on and off of household appliances, changes in industrial activity, or even exogenous factors like weather conditions.

In this thesis, a time period of 90 seconds was examined, during which the value of the energy demand changes continuously and in a random manner, as depicted in Figure 4.3. During each simulation, the time series representing the demand variation is randomized, in order to represent more realistic system operating conditions.

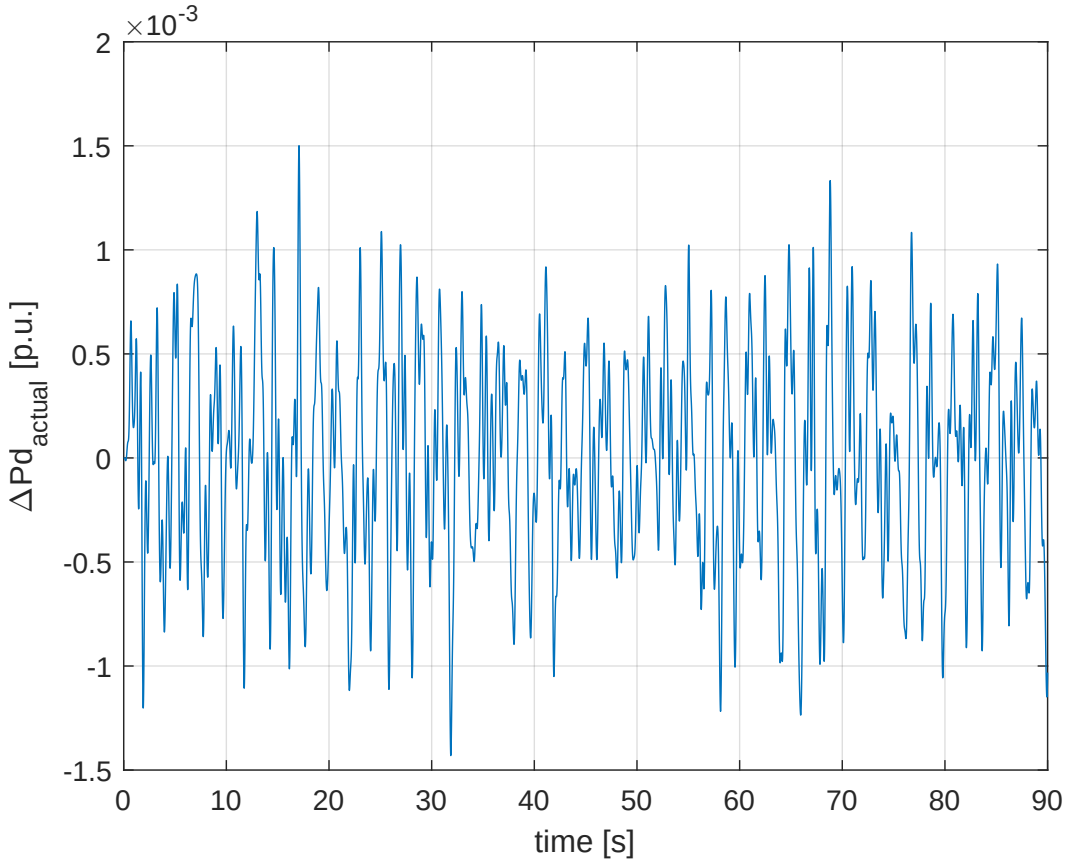

Figure 4.3: Actual deviation of the demand of active power.

The modeling of the subsystem responsible for the generation of the random input variation time series is shown in Figure 4.4. According to this specific figure it is evident that it consists of a signal source which obtains values from a sinusoidal function defined in MATLAB. The values of its constituent components change for each simulation of a different (H_1, H_2) pair, thereby enhancing the stochastic nature of the problem's disturbances. The amplitude of the sinusoidal signals has been defined by the MATLAB function to fluctuate within a p.u. value range, similar to that of small realistic grid disturbances ([-1.5, 1.5] p.u.).

Immediately after, the input signal is directed to an amplifier, which expands this specific set of values by a factor of 3.3. The purpose of this specific modification is to counteract the effect of the following blocks, which tend to reduce the amplitude. The value of 3.3 was proven, after a large number of conducted experiments, to be the ideal amplification factor so that the final output result again fluctuates within the desired range.

The time series subsequently enters a Zero-Order Hold (ZOH) block, whose function is to convert a continuous-time signal into a discrete one. The value captured at the sampling instant is held constant throughout the entire time interval between two consecutive samples.

In our case, this interval is selected to be 0.01 s (resulting in a total of 9001 time samples). This specific configuration bridges the analog and digital worlds and is particularly prevalent in cases where it is necessary to combine discrete blocks (e.g., filters) with continuous signals within the same model.

Finally, the now discretized signal is passed to a Low-Pass Finite Impulse Response (FIR) filter, which is designed to remove high frequencies from the signal, allowing only the low frequencies to pass. This process reduces noise and smooths the waveform, while preserving the signal's fundamental form. In simple terms, this filter cleanses the signal of rapid, undesired fluctuations and retains its slow, underlying shape. For this specific filter, a passband frequency of 1 Hz was selected, meaning that all frequencies up to 1 Hz pass through with minimal attenuation. On the other hand, the stopband frequency was set to 5 Hz, indicating that from 5 Hz and above, the signal is significantly attenuated. The interval between these two values (1–5 Hz) is the transition band, where attenuation increases progressively.

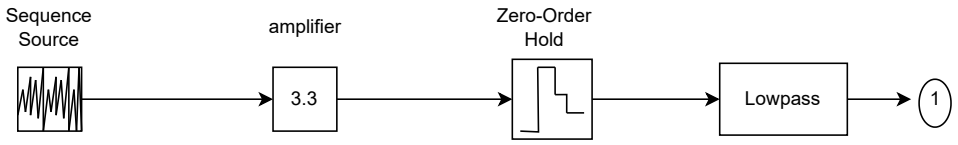

Figure 4.4: Modeling of random power disturbance source.

The stochastic nature of the active power demand variation is observed, as expected, in both Area 1 and Area 2,meaning that two sources of random disturbances are required in both areas. However, due to this stochastic nature, as observed during simulations, the machine learning models are unable to derive accurate estimates for the inertia constants from such small and continuously fluctuating disturbances. A review of the international literature reveals that in the vast majority of cases, the inertia constant is accurately determined during relatively large and typically step disturbances [12, 13]. However, since large disturbances may cause system instability and jeopardize the grid's safety, we resort to the use of controlled probing signals.

Being more precise, to mitigate the problems caused by the random nature of the demand on the system's output, we integrate a battery into the loads of Area 1 in each simulation. This battery is set to charging mode during the time period between 20-40 s. The charging activation is applied with a very short rise time, so that the induced change in the power balance approximates a step disturbance . After this interval ends, the battery is set back to discharging mode. In this manner, the final demand signal results from a combination of two consecutive step changes: an increase due to charging and a decrease due to discharging.The generated pulse is aggregated with the actual, randomly fluctuating demand, as shown in Figure 4.5. The corresponding ΔP_d subsystem is depicted in Figure 4.6.

The probing singal implementation is necessary only for one area of the power system, which we randomly selected as Area 1. In contrast, in the second interconnected area, the variable ΔP_{d2} corresponds directly to the actual value of the active power demand variation in Area 2. By adopting this configuration, we significantly facilitate the identification of the inertia parameters H, as will be analysed in the next section.

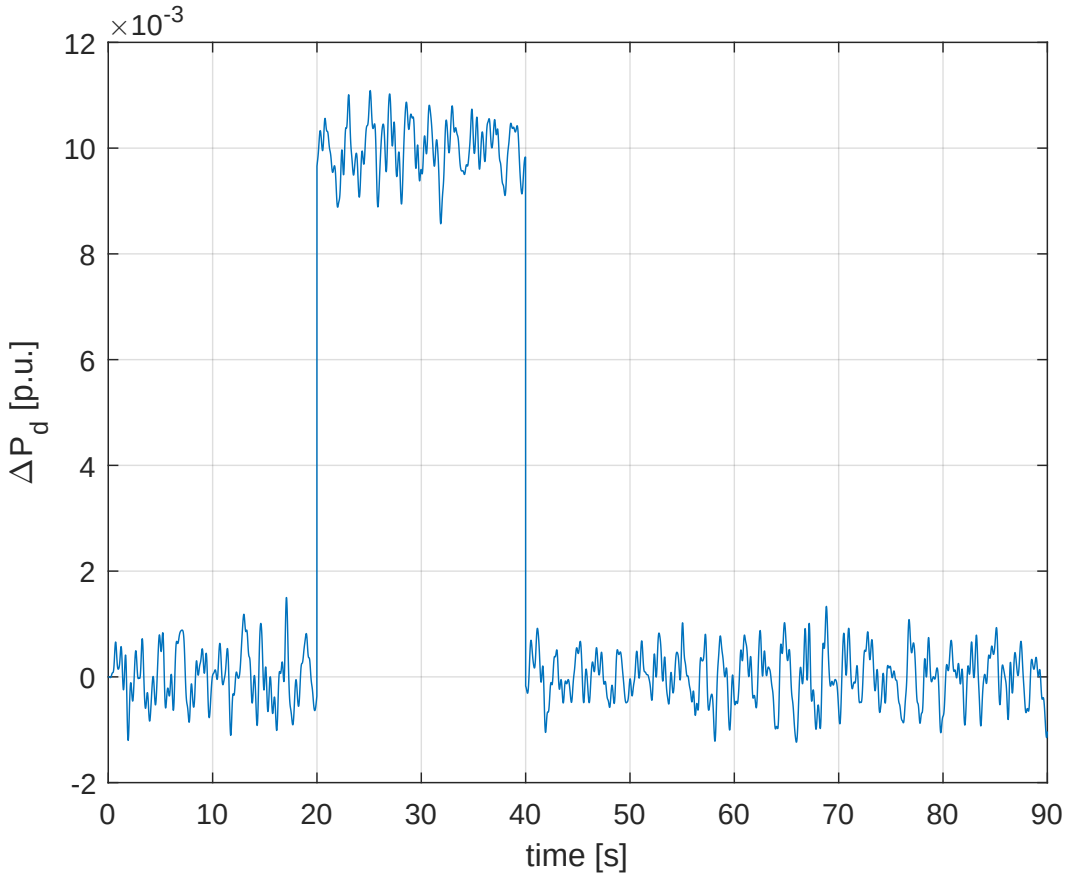

Figure 4.5: Final active power demand signal, resulting from the combination of the actual demand curve with the step change induced by the battery charging.

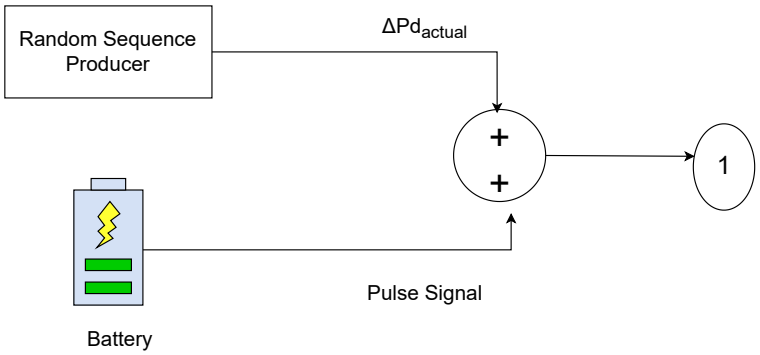

Figure 4.6: Implementation of the ΔP_d subsystem.

4.2.3 Controller

Each of the two subsystems showed in Figure 4.1 consists of the parallel connection of two integral controllers, which are multiplied by the term -1. The negative sign is necessary because a positive frequency error must lead to a command for reducing the output power.

Each integral controller is multiplied by the term K_I, known as the integral gain, which significantly influences the recovery time; the time required for the error to be zeroed. Subsequently, each of the controllers is assigned to one and only one generating unit of the Power Plants subsystem. The purpose of this specific subsystem is to simulate the dynamic behavior of secondary frequency control. The block diagram of the Controller subsystem for Area 1 is shown in Figure 4.7, while the implementation for Area 2 is similar.

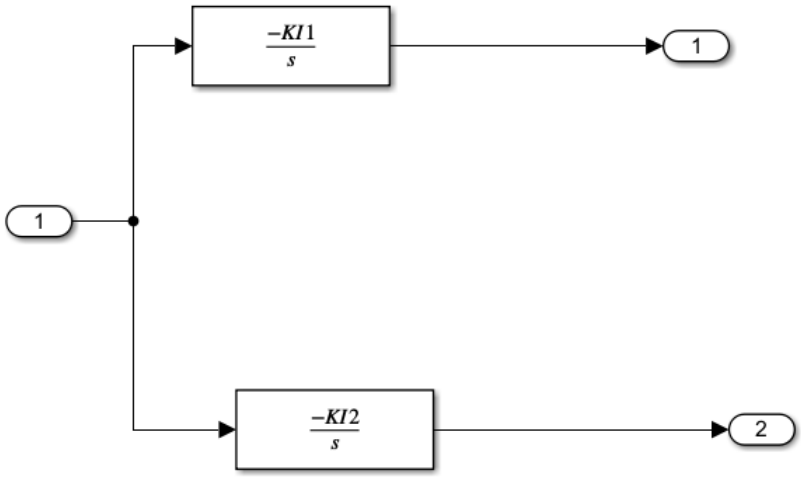

Figure 4.7: Implementation of the Controller subsystem for Area 1.

4.2.4 Power Plants Area Subsystems

As mentioned at the beginning of this section, multiple scenarios have been developed based on different combinations of generating units. The differentiation in the implementation of each scenario in Simulink is located exclusively in the final type of subsystem that remains to be analysed, namely the *Power Plants Area*. Two such subsystems appear in the configuration of Figure 4.1: the first comprises the generating units of Area 1, and the second comprises the generating units of Area 2.

Within the framework of this work, a total of three distinct scenarios were examined. These scenarios result from alternative combinations of generating units for the two areas of the studied power system. These scenarios are presented and analysed in detail in the following sections.

Scenario 1

In the present scenario, Area 1 consists of the parallel connection of two distinct sets of transfer functions, which describe the dynamic behavior of two types of steam turbines. In the first case, the transfer function corresponds to a single reheat steam turbine, while in the second, it describes the operation of a non-reheat steam turbine. The subsystem of Area 1, which comprises this parallel connection of the block diagrams of each steam turbine, is shown in Figure 4.8.

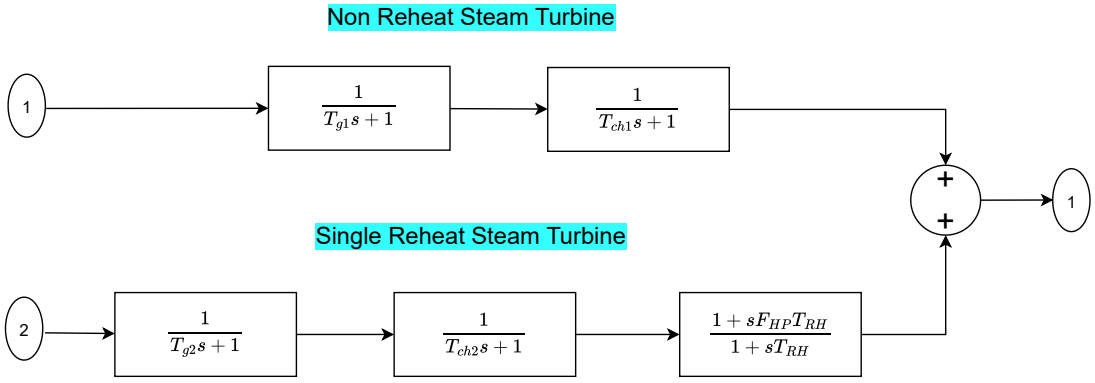

Figure 4.8: Block diagram of the Power Plants Area 1 subsystem.

In this figure, we observe that the input is directed into two branches. The upper branch describes the dynamic behavior of a non-reheat steam turbine. The parameter T_{g1} represents the turbine governor time constant, while the variable T_{ch1} represents the high-pressure chamber time constant.

Regarding the lower branch, it illustrates the transfer functions that constitute a single-reheat steam turbine. In this second branch, there is an additional reheating stage after the high-pressure chamber, where the steam is reheated before continuing to the turbine. The variable T_{RH} represents the reheater time constant, while F_{HP} represents the fraction of power passing through the high-pressure stage. The values T_{g2} and T_{ch2} reflect the corresponding time constants of the second steam turbine. Finally, these two branches are summed to produce the final output of the subsystem.

Area 2, on the other hand, is structured from the parallel connection of two block diagrams that simulate the operation of a hydro turbine and a diffrent non-reheat steam turbine. The internal layout of the generating units subsystem for Area 2 is presented in detail in Figure 4.9.

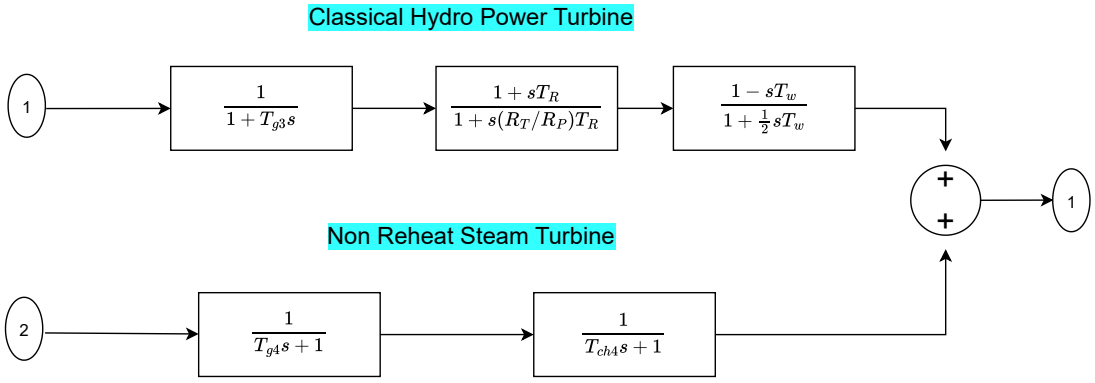

Figure 4.9: Block diagram of the Power Plants Area 2 subsystem.

Regarding the non-reheat steam turbine, its structure is identical to that of the corresponding subsystem in Area 1. The parameter T_{g4} represents the turbine governor time constant, while the parameter T_{ch4} expresses the high-pressure chamber time constant. For the purposes of scenario differentiation and more realistic simulation, the numerical values of these parameters were set slightly different from those of the corresponding steam turbine in Area 1.

The hydroelectric plant, on the other hand, simulates the dynamic behavior of a typical hydro turbine. The parameter T_{g3} represents the governor time constant, while the variable T_R represents the turbine time constant itself. The variables R_T and R_P relate to the hydraulic resistance and water pressure in the system, respectively, and their ratio (R_T/R_P) directly affects the turbine's dynamic response. Finally, the term T_w expresses the water time constant, which is associated with the water flow within the penstock.

The units and physical significance of all parameters in the diagrams describing the dynamic behavior of the generating units are shown in Table 4.1.

Table 4.1: Summary presentation of parameters for Area 1 and Area 2 models

Area	Symbol	Unit	Physical Significance
Area 1	T_{g1}	s	Governor time constant of the non-reheat steam turbine.
	T_{ch1}	s	High-pressure steam chest time constant.
	T_{g2}	s	Governor time constant of the single-reheat steam turbine.
	T_{ch2}	s	High-pressure steam chest time constant.
	T_{RH}	s	Reheater time constant in the reheat steam turbine.
	F_{HP}	-	Fraction of power passing through the high-pressure stage in the reheat steam turbine.
Area 2	T_{g4}	s	Governor time constant of the non-reheat steam turbine.
	T_{ch4}	s	High-pressure steam chest time constant of the steam turbine.
	T_{g3}	s	Governor time constant of the hydro turbine.
	T_R	s	Hydro turbine time constant (rotor inertia and flow mechanism).
	R_T, R_P	-	Parameters of hydraulic resistance and water pressure in the penstock; their ratio affects the dynamic response.
	T_w	s	Water time constant, expressing the flow delay dynamics in the penstock.

Scenario 2

This particular scenario is characterized by increased hydroelectric unit penetration. In Area 1, it includes two different transfer function blocks, which describe the operation of two different hydro turbines (Hydro Turbine 1, Hydro Turbine 2). These two units are connected

in parallel and their output power is summed to produce the total generated power of Area 1. Area 2, on the other hand, consists of the parallel combination of a single-reheat steam turbine and a non-reheat steam turbine.

The values of the parameters used in the implementation of the two-area interconnected control system whether they relate to the generating units or to the primary and secondary frequency control parameters may vary from scenario to scenario, aiming to ensure the stability of the grid.

Scenario 3

In this scenario, Area 1 is constructed from the parallel operation of a linear model describing a hydro turbine and a single-reheat steam turbine. The implementation of Area 2 is identical to Area 1, featuring the same generating units but with minor variations in their parameter values.

This scenario represents a more balanced unit distribution, as both areas combine a hydroelectric and a reheat thermal unit. This configuration enables the study of the interaction between the two technologies under interconnected system conditions.

4.2.5 Summary of Scenarios

The generating units that constitute the areas of the examined power system for each scenario are summarized in Table 4.2.

Table 4.2: Summary presentation of the generating unit scenarios in the two areas

Scenario	Area 1	Area 2
Scenario 1	Non-reheat steam turbine, Single-reheat steam turbine	Non-reheat steam turbine, Hydro turbine
Scenario 2	Hydro turbine 1, Hydro turbine 2	Non-reheat steam turbine, Single-reheat steam turbine
Scenario 3	Hydro turbine, Single-reheat steam turbine	Hydro turbine, Single-reheat steam turbine

4.2.6 Additional Grid Parameters

Beyond the four main subsystems found on the interconnected power system implementation of Figure 4.1, there are also several additional parameters related to the primary and secondary frequency control mechanisms, as well as the functionality of the interconnection between the control areas of the system. Firstly, we observe the constants R_1, R_2, R_3, R_4, every single one of them corresponding to the droop of a generating unit.

Furthermore, the parameters B_1 and B_2 denote the frequency bias factors and determine the response of the secondary control in each area. Their value is given by the relation

$$B_i = \frac{1}{R_{eq,i}} + D_i \quad \text{(in p.u.)},$$

where $R_{eq,i}$ is the equivalent value of the droop of the units in the i-th area and D_i is the load damping constant of the corresponding area.

Finally, we observe the transfer function

$$\frac{2\pi T_{12}}{s},$$

where the constant T_{12} is called the *synchronizing coefficient* between Area 1 and Area 2 and is measured in seconds. This parameter expresses the dynamic coupling between the two areas, i.e., how quickly and to what extent power changes in one area affect the frequency of the other.

4.2.7 Numerical Parameter Values

As mentioned previously, each scenario differs from the others, on the one hand due to the use of a different generation mix (different types of units) and on the other hand due to the use of different values for the model parameters. For certain parameters, such as the slope of the droop curves and the integral gains K_I of the controllers, differentiation was deemed necessary for the network to remain stable.Other parameters,however, were slightly modified per scenario to correspond to more realistic conditions.

The context in which a parameter appears may differ from scenario to scenario, meaning that the same nomenclature may correspond to a different physical quantity. For example, the parameter T_{g1} in Scenario 1 denotes the turbine governor time constant of a single-reheat steam turbine, whereas in Scenario 3 it denotes the governor time constant of a hydro turbine. The parameter values for each scenario are summarized in tables below, while their physical significance in each case is analysed at the same time.

Scenario 1

Regarding Scenario 1, the physical significance of the parameters was analyzed previously in Table 4.1, while Table 4.3 collectively presents the numerical values of all parameters encountered in this specific scenario. Three colored rows in the table indicate the separation between the two areas and the intermediate network, making it clear whether a parameter belongs to a generating unit operating in Area 1, in Area 2, or is found between them.

Table 4.3: Parameter table for Scenario 1

Parameter	Numerical Value
D_1	0.9
T_{g1}	0.23
T_{ch1}	0.39
T_{g2}	0.14
T_{ch2}	0.5
T_{rh}	7
F_{hp}	0.05
R_1	0.05
R_2	0.05
K_{I1}	0.25
K_{I2}	0.25
B_1	40.9
D_2	1.5
T_{g3}	0.4
T_w	2.25
T_r	5
R_t	0.38
R_p	0.05
T_{g4}	0.3
T_{ch4}	0.45
R_3	0.05
R_4	0.05
K_{I3}	0.25
K_{I4}	0.25
B_2	41.5
T_{12}	2
a_{12}	-1

Scenario 2

In Scenario 2, Area 1 consists, as already mentioned, of the parallel connection of two blocks representing two different hydro turbines. In this scenario, the parameters T_{g3} and T_{g4}

correspond to the governor time constants of the respective hydro turbines, while the terms T_R, R_T, R_P, and T_w, which remain common to both models, correspond to the physical quantities of the hydro turbine time constant, hydraulic resistance, water pressure, and water time constant, respectively.

Regarding Area 2, we mentioned that it is comprised of two steam turbines, one of which operates without reheat while the second reheats its steam once. The parameters T_{g1} and T_{g2} correspond to the governor time constants of each turbine, while the terms T_{ch1} and T_{ch2} express the high-pressure steam chest time constants of the non-reheat steam turbine and the single-reheat steam turbine, respectively. Finally, the parameters T_{RH} and F_{HP} pertain exclusively to the single-reheat steam turbine and correspond to the reheater time constant and the fraction of power passing through the high-pressure stage, respectively.

The numerical values of all parameters of the models that constitute the two areas of Scenario 2 are presented in Table 4.4.

Scenario 3

In Scenario 3, both areas are composed of the parallel connection of two linear models, which represent a hydro turbine and a single-reheat steam turbine.

Within the framework of this specific scenario, the time constants T_{g1} and T_{g2} represent the turbine governor time constants for the hydro turbine and the steam turbine of Area 1, respectively. The parameters T_R, R_T, R_P, and T_w retain the same meaning as in the previous scenarios. Furthermore, the term T_{ch2} constitutes the high-pressure steam chest time constant of the steam turbine in Area 1.

Similarly, in Area 2, the parameters T_{g3} and T_{g4} relate to the turbine governor time constants for the hydro turbine and the steam turbine, while the term T_{ch4} corresponds to the high-pressure steam chest time constant of the steam turbine. The numerical values of all parameters are presented in Table 4.5.

Table 4.4: Parameter table for Scenario 2

Parameter	Numerical Value
D_1	0.9
T_{g1}	0.23
T_{ch1}	0.39
T_{g2}	0.14
T_{ch2}	0.5
T_{rh}	7
F_{hp}	0.4
R_1	0.25
R_2	0.25
K_{I1}	0.25
K_{I2}	0.25
B_1	8.9
D_2	1.5
T_{g3}	0.4
T_w	2.25
T_r	5
R_t	0.38
R_p	0.05
T_{g4}	0.5
T_{ch4}	—
R_3	0.05
R_4	0.05
K_{I3}	0.25
K_{I4}	0.25
B_2	41.5
T_{12}	2
a_{12}	-1

Table 4.5: Parameter table for Scenario 3

Parameter	Numerical Value
D_1	0.9
T_{g1}	0.45
T_{g2}	0.14
T_{ch2}	0.5
T_{rh}	7
F_{hp}	0.4
R_1	0.25
R_2	0.05
K_{I1}	0.25
K_{I2}	0.25
B_1	24.9
D_2	1.5
T_{g3}	0.4
T_w	2.25
T_r	5
R_t	0.38
R_p	0.05
T_{g4}	0.16
T_{ch4}	0.55
R_3	0.25
R_4	0.05
K_{I3}	0.25
K_{I4}	0.25
B_2	25.5
T_{12}	2
a_{12}	-1

Chapter 5

Development of Proposed Models

5.1 Introduction

This chapter analyzes the approach followed for the development of the proposed models. Specifically, a detailed reference is made to the methodology of collecting and processing the data that will be used for developing the machine learning methods. Furthermore, the training philosophy of the machine learning models for estimating inertia constants from PMU measurements is presented in detail.

5.2 Synthetic Dataset Construction

The first step in developing the models is the collection of training data. Due to the limited access to official real-world measurement data, this thesis develops realistic PMU frequency responses using the frequency response models analyzed in Chapter 4 and implemented in Simulink. In this way, a synthetic dataset was created. More specifically, the data generated with the help of Simulink include the frequency response of Area 1 (Δf_1), the frequency response of Area 2 (Δf_2), and the change in the tie-line power (ΔP_{tie}). In all cases considered, the data have a time duration of 90 seconds. A representative example of the dynamic responses is presented in Figure 5.1. The machine learning models to be developed in the context of this thesis aim to estimate the inertia constants H_1 and H_2 of the two areas of the power system, using the dynamic responses Δf_1, Δf_2, and ΔP_{tie} as inputs.

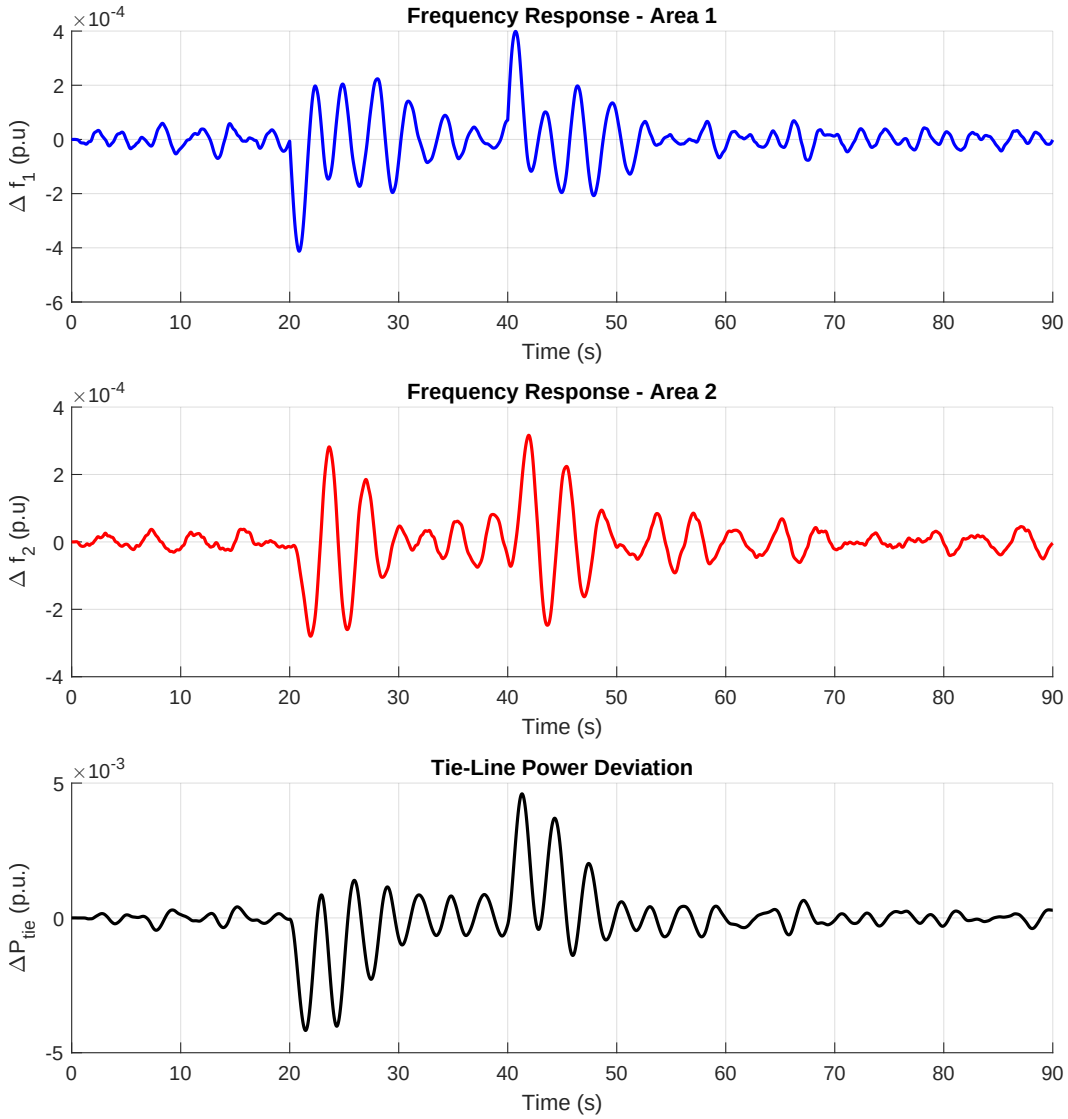

Figure 5.1: Dynamic responses of the system for a random pair (H_1, H_2)

Keeping all the parameters presented in the previous chapter constant, for each scenario of generating units, we capture and store the frequency responses as well as the change in the tie-line power, considering a unique combination of values for the inertia constants H_1 and H_2. For each combination of H_1 and H_2, the SFR model is triggered with random small power disturbances, resulting in the dynamic responses Δf_1, Δf_2, and ΔP_{tie}, which will subsequently be used to estimate the values of H_1 and H_2.

It should be noted that, apart from H_1 and H_2, other system parameters also influence the responses Δf_1, Δf_2, and ΔP_{tie} to some degree [6, 27], such as the governor speed droop R and the load-damping constant D. However in any scenario in the current research , these specific coefficients of the units can be considered to have known values. For this reason, in the present thesis, we focus on the estimation of the inertia constants H_1 and H_2 using

machine learning methods.

5.2.1 Probing Signal Synthesis

Considering the set of parameters found in the block diagram of Figure 4.1 as constant in each simulation, our goal is the accurate estimation of the values of the pair (H_1, H_2) under the influence of a random power disturbance for a given set of generating units. However, the unpredictable nature of the power demand at any given moment makes it difficult to construct a machine learning model which, by taking the frequency responses and the tie-line power as inputs, can predict the pair of inertia constants H_1 and H_2. The reason is that the shape of the dynamic responses Δf_1, Δf_2, and ΔP_{tie} is influenced to a very large degree by the stochastic variation of the power demand, which differs randomly in each simulation.

To address this problem, we resort to the use of probing signals. Probing signals are signals that are imposed by the system operator via an enforced change in power. In this way, a small, fully controlled disturbance is performed, which allows, as we will also see in the chapter with the relevant results, a precise estimation of the inertia constants H_1 and H_2.

For the synthesis of the probing signals, we utilize the nature of batteries, which are characterized by their ability to activate near-instantaneously in charging and discharging operations. Thus, we create test signals that satisfactorily approximate the form of pulses, with an amplitude one order of magnitude larger compared to the random power demand.

Thus, the final signal ΔP_d in each simulation results from the combination of the two disturbances, as is also shown in Figure 5.2. The result is that the shape of the final responses of Δf_1, Δf_2, ΔP_{tie} is influenced primarily by the imposed test signal and much less by the random and unpredictable load variation.

Consequently, the problem of the input's randomness is transformed into a pseudo-random one, allowing for the reliable correlation between the pair (H_1, H_2) and the observed responses. In other words, the values of H_1 and H_2 and the frequency responses are now exclusively interdepended. In this way, the training of the machine learning models is significantly facilitated.

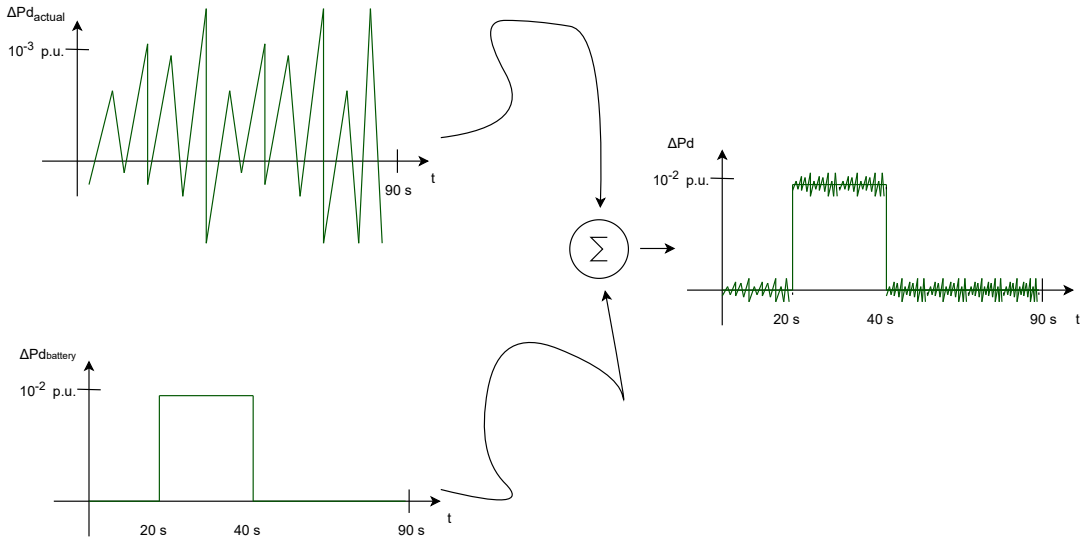

Figure 5.2: Probing signal synthesis.

5.2.2 Inertia Constant Estimation

To gain a deep understanding of the problem we are asked to solve, it is essential to check again the initial diagram of the two interconnected areas in Simulink, which is presented in Figure 4.1. This block diagram, receiving as inputs the changes in power demand in the two areas ($\Delta P_{d1}, \Delta P_{d2}$), produces three fundamental outputs: the frequency response of Area 1 (Δf_1), the frequency response of Area 2 (Δf_2), and the change in the tie-line power (ΔP_{tie}) during which power is exchanged between the two areas.

By substituting all the parameters mentioned in Chapter 4, including the inertia constants H_1, H_2, with numerical values drawn from relevant experiments in the literature [**?**, 28], the model generates, for each input pair $(\Delta P_{d1}, \Delta P_{d2})$, a set of three output time series: Δf_1, Δf_2, and ΔP_{tie}.

As mentioned in the previous subsection, during each simulation the system is triggered from a pseudo-random input disturbance, which is primarily due to the charging-discharging operation of the battery in Area 1, while all other parameters are held constant, except for the pair H_1, H_2. Aiming to examine the system's sensitivity to different values of equivalent inertia per area, which reflect the level of Renewable Energy Source (RES) penetration in the grid, we use a total of 1000 different value pairs (H_1, H_2), with each value ranging in the

interval $[3, 8]$ s, which realistically approximates operational scenarios.

Since inertia is now the only parameter that can influence the outcome, we try to find a model that will link the inertia to the actual system responses, thereby establishing an input-output relationship between the inertias and the system responses.

The whole philosophy behind the development and evaluation of the machine learning models to be used is based on the simple idea that, since a unique pair of inertia values H_1, H_2 determines the shape of the system's frequency responses and tie-line power in Figure 4.1, then logically the inverse reasoning should also hold: a unique combination of frequency responses ($\Delta f_1, \Delta f_2$) and tie-line power (ΔP_{tie}) corresponds exclusively to one and only one pair of values H_1, H_2. Consequently, the training process will be based each time on mapping a group of three time series to two specific output values, the pair of inertia constants.

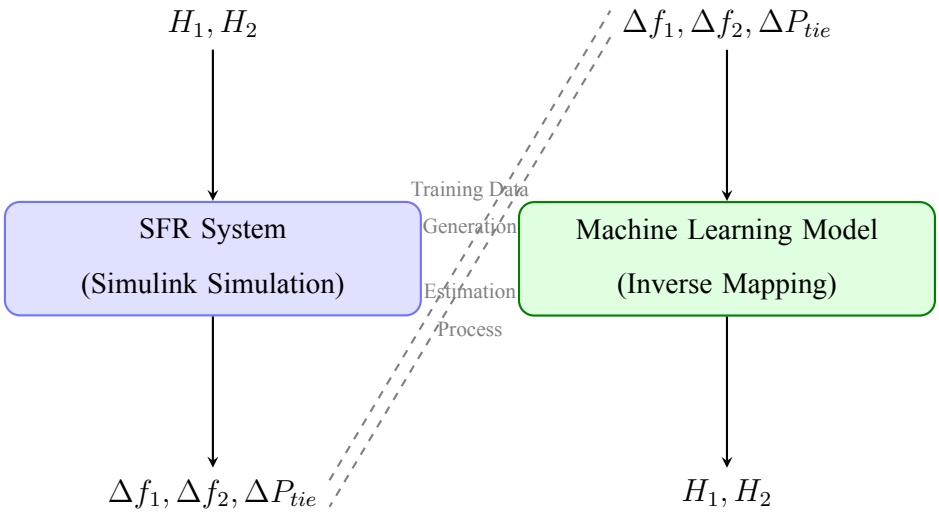

Figure 5.3: Conceptual diagram of the forward and inverse mapping processes. Forward mapping (left): $(H_1, H_2) \mapsto (\Delta f_1, \Delta f_2, \Delta P_{tie})$. Inverse mapping (right): $(\Delta f_1, \Delta f_2, \Delta P_{tie}) \mapsto (H_1, H_2)$. The dashed lines indicate how the simulation output becomes training data for the machine learning model.

In the real world, the data available for processing are the outputs of the system in Figure 4.1 (Δf_1, Δf_2, ΔP_{tie}), which can be easily obtained via measurements. What we cannot know and are called upon to find is the pair of inertia constants H_1 and H_2, which is responsible for the behavior of the system's output time series.

For this reason, we apply an inversion regarding what we consider as input and output in our problem. Although in reality the values of H_1 and H_2 influence the system's output by acting as parameters, we now consider that the input of the inverse problem is the time series

themselves (frequency and power responses), while the output is the pair of H_1 and H_2, which uniquely determines the shape of the responses. This process is called inverse mapping, and its operating logic is illustrated in Figure 5.3.

The problem we are called upon to solve in data science is defined as a sequence tovalue regression problem. In such cases, the goal is, given a set of time series, to be able to estimate an associated scalar output value. Problems of this nature are widely encountered in the literature, with forecasting energy demand from historical load data, estimating the remaining useful life of industrial equipment based on sensor measurements, as well as the analysis of biomedical signals (e.g., electrocardiograms - ECG, electroencephalograms - EEG) being some of the most classic examples [5].

5.2.3 Sampling and Data Collection

The dynamic responses of the frequencies in the two areas, as well as the dynamic response of the change in tie-line power caused from a random pair of values H_1, H_2 (6.5, 5.86) of the system in Figure 4.1, for Scenario 1, were previously presented in Figure 5.1.

To ensure the most efficient and economical storage of the time series without loss of the information they contain, it was decided to sample each time series with a step of 0.01 s. Consequently, each of the time series, from the initial time 0 up to 90 s, will consist of a total of 9001 samples, which will be sufficient to reconstruct it if necessary. The duration of 90 s was deemed appropriate as the termination point for the recording because, within this interval, there is sufficient time for both the primary and secondary frequency control to activate following the step change due to the battery at 20 and 40 seconds.

For the storage of the sampled signal for each of the groups of three time series corresponding to each of the 1000 different (H_1, H_2) pairs, a tensor arrangement was adopted. A tensor is essentially a three-dimensional array, which results from the parallel placement of a set of two-dimensional matrices, as shown in Figure 5.4. A three-dimensional tensor is described by a triple of numbers (w, d, l). By observing Figure 5.4, one can easily perceive that the first coordinate corresponds to its depth, i.e., the number of layers (or, more precisely, the parallel matrices that constitute it). The second coordinate corresponds to the number of rows of each matrix, while the third coordinate concerns the number of its columns.

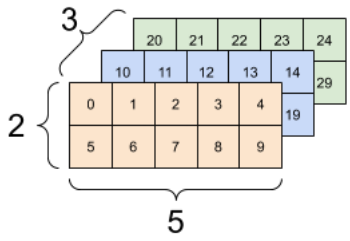

Figure 5.4: Tensor representation (3,2,5)

In our case,the number of columns in each matrix corresponds to the number of input time series (Δf_1, Δf_2, ΔP_{tie}), while the number of rows in each matrix corresponds to the number of samples resulting from the sampling process. The contents of each matrix are the numerical values of the time series for each time instant, according to the sampling rate, and are associated exclusively with a specific pair of values (H_1, H_2). Each matrix, therefore, stores a set of three time series corresponding to a unique (H_1, H_2) combination. The total number of all matrices is equal to the number of different (H_1, H_2) combinations, i.e., 1000, and they are stored in parallel, as shown in Figure 5.5.

Mathematically, the dataset of time series can be represented as a tensor

$$X \in \mathbb{R}^{1000 \times N \times 3},$$

where $N = 9001$ is the number of samples in each time series and the last axis corresponds to the three inputs (Δf_1, Δf_2, ΔP_{tie}). Correspondingly, the output variables, i.e., the pairs of inertia constants, are represented as a two-dimensional matrix

$$Y \in \mathbb{R}^{1000 \times 2},$$

where each row contains the corresponding pair (H_1, H_2) for the three time series of the same matrix in X. That is, there will be a mapping of the first of the three coordinates of the tensor X to each row of the matrix Y.

The same process is repeated for all scenarios of the different generation unit mixes, as the shape of the frequency and power responses is influenced by the type of linear functions used to simulate the dynamic behavior of those units. In the remainder of this chapter, we will use Scenario 1 as a reference, since the data analysis and management strategies applied to the other two scenarios are similar.

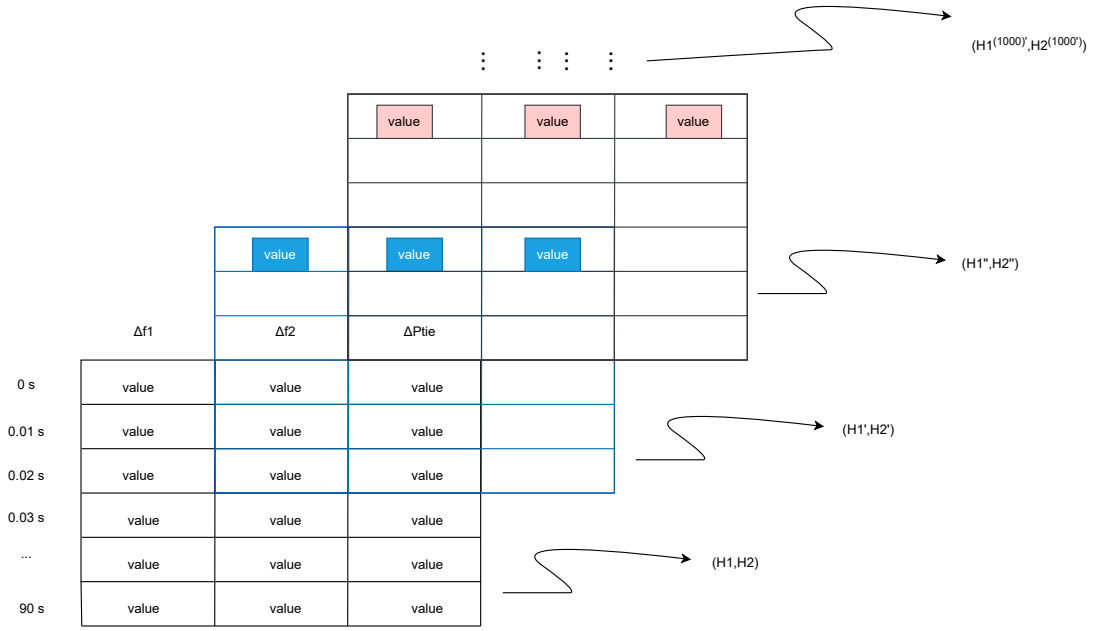

Figure 5.5: Time series storage into a tensor

5.3 Data Preprocessing

Now that the synthetic dataset has been successfully constructed and stored, the next step is its preprocessing before it is fed into a Machine Learning model that will generate predictions.

5.3.1 Feature Extraction

The first transformation the tensor undergoes is a process called feature extraction. Because the complexity and size of the time series make their direct processing difficult, it was decided to study each time series separately and construct a new dataset, consisting of indicators that describe its behaviour. From each input time series (Δf_1, Δf_2, ΔP_{tie}) the following indicators are initially extracted:

- Mean value:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

- Standard deviation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

- Maximum value:

$$x_{\max} = \max_{1 \leq i \leq N} (x_i)$$

- Minimum value / Nadir:

$$x_{\min} = \min_{1 \leq i \leq N} (x_i)$$

- Root Mean Square (RMS):

$$\mathrm{RMS}(x) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} x_i^2}$$

- Peak-to-peak (difference between maximum and minimum value):

$$P2P = x_{\max} - x_{\min}$$

- Time of Nadir: The time instant at which the largest frequency drop is observed.

$$t_{\min} = \arg\min_{t_i} (x(t_i))$$

The new dataset is stored again, but this time in a two-dimensional matrix. This matrix consists of 23 columns, of which the first 21 correspond to the statistical features of each input time series and essentially constitute the set of independent input variables. The last 2 columns contain the pair of inertia constants (H_1, H_2), which are the output variables. According to this logic, the matrix contains a total of 1000 rows, where each row describes the set of three time series corresponding to a unique output combination (H_1, H_2).

Mathematically, the dataset can be represented as follows:

$$D \in \mathbb{R}^{1000 \times 23}, \quad D = [X \mid Y],$$

where

$$X \in \mathbb{R}^{1000 \times 21}$$

represents the independent variables (features) resulting from the statistics of the input time series, while

$$Y \in \mathbb{R}^{1000 \times 2}$$

contains the output variables, i.e., the pair of inertia constants (H_1, H_2).

5.3.2 Rejection of Unstable Signals

After the initial two-dimensional matrix has been constructed, while simultaneously maintaining access to the tensor containing the graphical representations of all input signals (Δf_1, Δf_2, ΔP_{tie}), the next step is to clean the dataset of invalid data. Due to the random assignment of numerical values to the variables H_1, H_2 during the simulations in *Simulink*, where their range was the interval $[3, 8]$, some of the responses of the system in Figure 4.1 exhibited instability behaviour;the appearance of undamped oscillations, continuously increasing frequency or power deviation, and a general inability of the system to return to a new equilibrium after the disturbance were observed [4]. In other words, there are combinations of inertia values (H_1,H_2) that render the system unstable, and therefore, both these specific values and the accompanying power and frequency responses must be discarded, as this thesis exclusively examines stable operation scenarios.

The question posed at this point is how the computer can decide on the presence or absence of instability.

A classical method is the application of the Routh-Hurwitz criterion, which is capable of determining the existence of poles of the closed-loop transfer function in the right-half plane. The general rule is that if even one root is found in the first or fourth quadrant, then the system is considered unstable. However, the existence of multiple generating units makes the construction of the closed-loop transfer function a particularly difficult and time-consuming process.

For this reason, it was decided to utilize the tendency of unstable signals to exhibit a significant increase in their amplitude over time. The instability detection method adopted is based on calculating the RMS value of each signal over the last 10% of the total response time (i.e., from $t = 81$ s to $t = 90$ s). Subsequently, the RMS value of the same signal over the first 30% of the response time (i.e., from $t = 0$ s to $t = 27$ s) is calculated. These two values are divided, and the final result is defined as the *RMS_ratio* index:

$$\text{RMS\_ratio} = \frac{\text{RMS}(x_{81-90s})}{\text{RMS}(x_{0-27s})}.$$

Each of the three input signals to our model has its own index, which describes the relationship between the magnitude of the response amplitude fluctuation at the beginning and the end of the observation period. If this ratio exceeds the value of 2, it means that the signal at the end has at least double the RMS value compared to the beginning, a fact that constitutes

a clear indication of instability.

Therefore, three additional features are added to the initial two-dimensional matrix, increasing its dimension to 1000×26.

Since each row of the matrix now possesses three additional indicators, which signify whether the statistics of the hosted time series correspond to unstable signals or not, we proceed with the deletion of rows. Specifically, if even one of these features has a value greater than 2, then the row containing this value is dropped from the matrix.

Therefore, after the data across the entire matrix were subjected to the above check, the unstable cases corresponding to non-realistic operation scenarios were identified and excluded from further analysis.

5.3.3 Feature Selection

The next step of data preprocessing is feature selection. As mentioned in the chapter on data organization, this process aims to remove features whose degree of influence on the final outcome is negligible. It has been proven that this transformation significantly improves prediction accuracy, as many features are practically uncorrelated with the output and often mislead the model from the correct solution. Furthermore, reducing the data size ensures lower computational cost and faster model training.

Within the framework of this thesis, a correlation matrix was used to select the most representative features. This matrix quantifies the degree of linear correlation between the features and the output, allowing for the removal of those that exhibit low or zero correlation. In this way, only features with a substantial contribution to the prediction are retained, leading to more efficient and accurate models.

The most widely used metric for calculating correlation is the Pearson correlation coefficient, which for two random variables X and Y is defined as:

$$r_{X,Y} = \frac{\mathrm{cov}(X, Y)}{\sigma_X \, \sigma_Y}$$

where $\mathrm{cov}(X, Y)$ is the covariance of the two variables, and σ_X and σ_Y are their standard deviations. The values of the coefficient range in the interval $[-1, 1]$, with values close to $+1$ indicating a strong positive correlation, close to -1 a strong negative correlation, while values close to 0 denote a lack of a linear relationship. The correlation matrix, which shows the degree of influence of each feature on each of the two outputs of the model, is shown in

Figure 5.6.

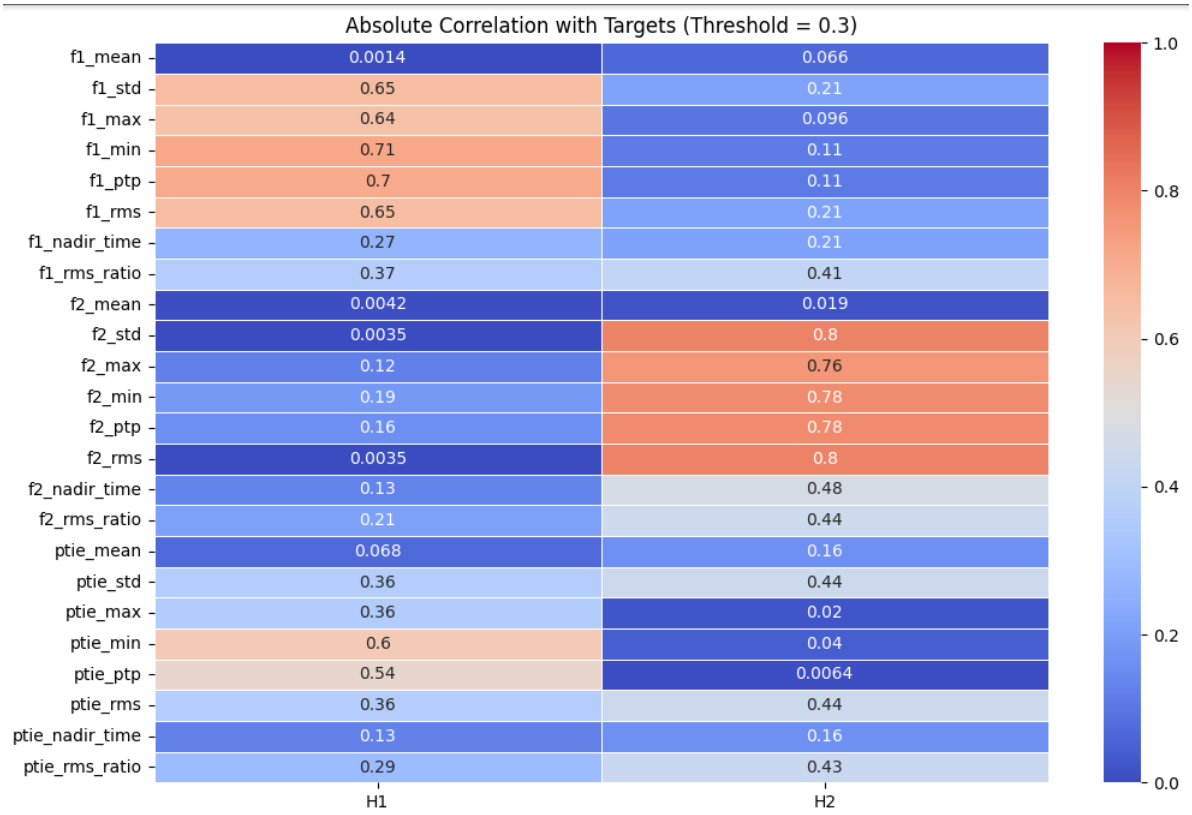

Figure 5.6: Correlation Matrix

A minimum correlation coefficient threshold of 0.3 was set, meaning that any feature exhibiting a degree of linear dependence with both system outputs smaller than this value is immediately rejected from the dataset. By observing the information provided by the correlation matrix, we can conclude that the statistical features describing the frequency response of a specific area have a greater degree of influence on the formation of the equivalent inertia of that same area, but not of the other. Colors are used to depict the proximity of the correlation coefficient to unity: the redder a cell appears, the more its value approaches $+1$, while the bluer it is, the closer it is to zero.

Based on the correlation matrix, a total of 5 input features that did not meet the minimum correlation coefficient criterion were rejected. These specific *features* are the following: *f1_mean*, *f1_nadir_time*, *f2_mean*, *ptie_mean*, *ptie_nadir_time*. Consequently, the final data matrix is now shaped with dimensions 722×21.

5.3.4 Scaling and Splitting

The final stage of preprocessing for the current dataset to be undergone to is scaling. Among the various scaling methods encountered in data science, the *Min-Max Scaler* method was selected for this work. This choice is justified by the fact that the numerical values of most features are already expressed in per unit (p.u.), resulting in them having different value ranges from one another. Without proper scaling, features with larger absolute magnitudes could dominate the training process, leading the model to erroneous estimates.

With the application of the *Min-Max* method, all values are normalized to the interval $[0, 1]$, ensuring that each feature contributes equally to the training process and improving both the stability and convergence of the machine learning algorithms. The mathematical definition of Min-Max normalization for a variable x is given by the formula:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}},$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of the variable respectively, and x' is the normalized value in the interval $[0, 1]$.

Subsequently, two sub-matrices are constructed from the original matrix. The first will include 70% of the instances and will be used for training the models (*training dataset*), while the second will include the remaining 30% and will be utilized to evaluate each trained model with data that is unknown to it (see Chapter 3). Following this, the two new sub-matrices are further split into dependent and independent variables. The matrix of independent variables has 19 columns, while the matrix of dependent variables has 2.

At this point, the data are fed as inputs to the various machine learning models analyzed in Chapter 3. The results of each model, as well as their interpretation, will be presented in the next chapter. The present chapter has served as a reference point for the entire thesis, as it analyzed the process of constructing, storing, and preprocessing the data, which will constitute fundamental components in achieving our ultimate goal: the determination of inertia, which will be presented in the next section.

Chapter 6

Evaluation

6.1 Introduction

This chapter conducts a comprehensive evaluation of the proposed inertia estimation methods described in the previous chapters. Because the performance of a machine learning algorithm cannot be known a priori, in the majority of applications it is common practice to test multiple algorithms for estimating the target value. The suitability of each method depends heavily on the nature and structure of the data. Consequently, it is impossible to predict which algorithm will perform best or will most effectively comprehend the system's behavior unless we first try it.

For this reason, this chapter provides a detailed presentation of the evaluation metrics for each algorithm across all unit commitment scenarios. Interactive plots are also provided, which capture the estimation error and visualize the results from different perspectives, facilitating a clear understanding of the differences between the methods. This approach enables a comparative assessment of their reliability and efficiency.Lastly, having trained our models and evaluated the capability of each algorithm in estimating the pair of inertia constants, we outline how our proposed method could be deployed in a real-world grid. This outlines a pathway for the real-time (online) monitoring of system inertia.

6.2 Results - Scenario 1

The pair of inertia constants (H_1, H_2) is approximated using the five machine learning algorithms analyzed in Chapter 3, namely MLR, Random Forest, XGBoost, SVR, and MLP.

The inherent nature of the problem initially classifies it as a sequence-to-value type. However, through the pre-processing procedure described in the previous chapter, a new dataset was constructed from the original one, resulting in the transformation of the problem into a classic multi-output regression problem with multiple input variables and two output variables. The simultaneous estimation of the two output variables is not merely a technical choice but also reflects the physical reality of the system. Indeed, given that the constants H_1 and H_2 refer to interconnected areas, they simultaneously influence the dynamic behavior of the system. Their parallel estimation via multi-output regression allows the model to leverage this interdependence, leading to more consistent and reliable results. At this point, it is worth noting that the error differs depending on which inertia constant is being estimated by each model; in some cases, the estimation of constant H_1 may be more accurate, i.e., closer to its true value, compared to the estimation of the corresponding constant H_2 of the same pair, and vice versa.

6.2.1 Random Forest Evaluation

The evaluation results for each output, using the Random Forest algorithm, are summarized in Table 6.1.

It is observed that this specific algorithm achieves very high values for the (R^2) metric, with the scores exceeding 0.92 for both outputs, demonstrating its ability to fit the data with high accuracy. The values of the RMSE and MAE errors remain at low levels and are comparable between the two outputs, indicating that the model's performance is balanced. Furthermore, the MAPE fluctuates around 4%, meaning the average relative deviation from the true values is small. Overall, the results confirm that the Random Forest algorithm provides a highly reliable estimation of the inertia constants, with slightly better performance in estimating H_2 compared to H_1.

Table 6.1: Evaluation results of the Random Forest algorithm for the outputs H_1 and H_2.

Output	R^2	**RMSE**	**MAE**	**MAPE (%)**
H_1	0.93537	0.32258	0.23332	4.34
H_2	0.92647	0.31709	0.23889	4.06

The deviation between the actual and the model-estimated values for each inertia constant

can be better visualized in Figure 6.1.

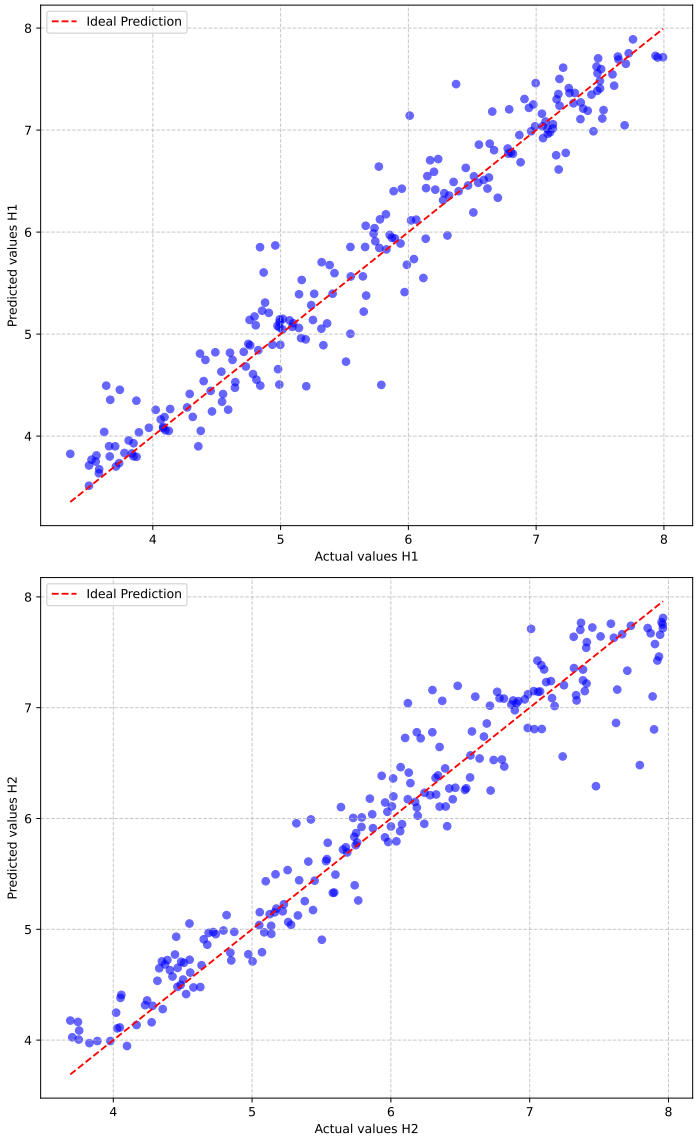

Figure 6.1: Deviation between actual and predicted values of the constants H_1, H_2 from the Random Forest model.

According to the specific Figure, for each of the inertia constants H_1 and H_2, we can represent their error using a two-axis system. The horizontal axis represents the actual value of the inertia in the range $[3, 8]$, while the vertical axis represents the corresponding predicted value resulting from the respective model. Each blue dot appearing on the plane is described by a pair of coordinates, where the abscissa corresponds to the actual value and the ordinate to the predicted value. The optimal case, i.e., the elimination of error, corresponds to a perfect match between the actual and predicted values. Geometrically, this means that all dots would lie on the line $y = x$. Consequently, the distance of each point from this line directly reflects

the capability of the respective algorithm to accurately determine the inertia.

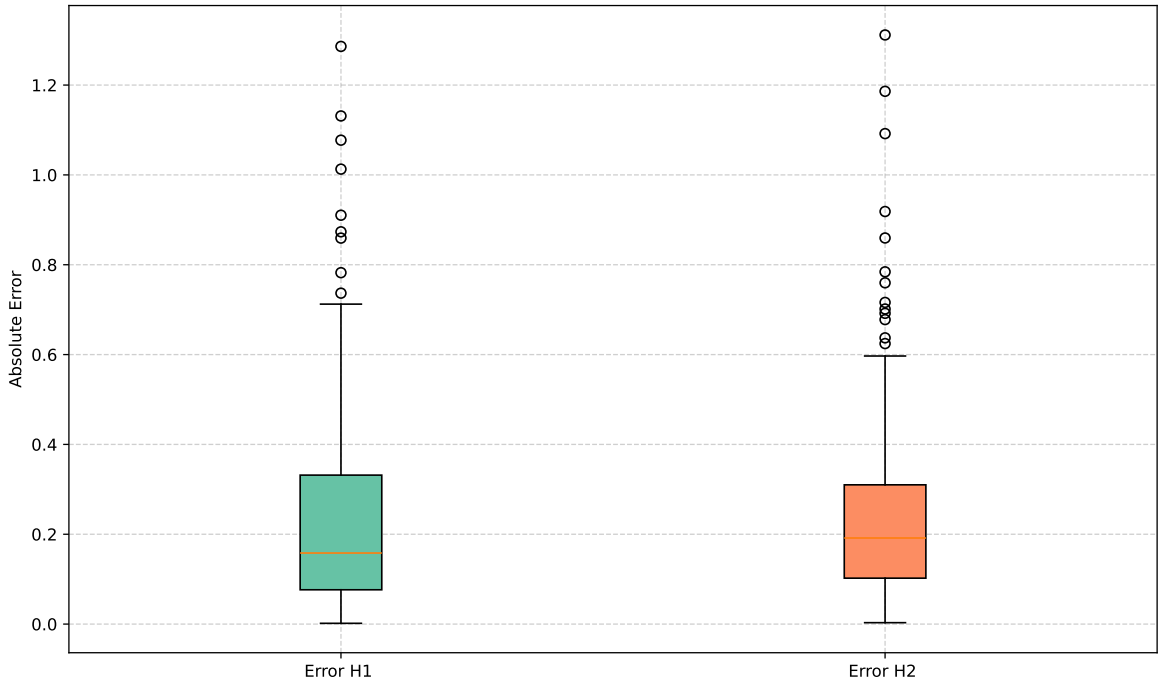

Figure 6.2: Box plots of the absolute error for the predictions of H_1 and H_2 using Random Forest.

An alternative way to visualize the errors is through box plots, as shown in Figure 6.2. In contrast to the previous diagrams, which focus directly on the comparison of actual and predicted values, box plots aim to depict the distribution of the error.

A closer look to Figure 6.2 will reveal that the y-axis represents the absolute error, i.e., the modulus of the difference between the true value of each inertia constant and the value estimated by the model ($|H_{\text{actual}} - H_{\text{predicted}}|$), while the x-axis includes the two inertia constants H_1 and H_2 whose error distributions we aim to show. For each inertia constant, an independent box plot is constructed. The process follows these steps: for each predicted inertia value, the absolute error is calculated, and the results are collected in a list. This list is then sorted in ascending order, and the quartiles are calculated. Each box of the plot extends from the first quartile ($Q1$, the 25% of the data) to the third quartile ($Q3$, the 75% of the data), thus encompassing the middle 50% of the distribution. Inside the box, a black line corresponds to the median of the error, which is robust to outliers and represents the ”typical” error. On the other hand, the whiskers extend to a range $[a, b]$, which is determined by the relations:

$$a = Q1 - 1.5 \cdot IQR, \qquad b = Q3 + 1.5 \cdot IQR, \qquad IQR = Q3 - Q1 \tag{6.1}$$

Where the values $Q1$ and $Q3$ correspond to the absolute error at the 25% and 75% of the sorted list, respectively. Any value outside the range $[a, b]$ is considered an outlier and is represented in the diagram as a separate dot.

The interpretation of the *box plots* is particularly useful for comparing the two outputs. A smaller range (width of the box) indicates that the errors are more concentrated around the median, implying the model exhibits greater stability. On the other hand, a larger range means the model shows greater dispersion in its errors. Furthermore, the presence of many outliers may indicate that the model, in some instances, fails to fit properly [29]. In our case, we observe that H_2, despite having many outliers, simultaneously exhibits a smaller error range than H_1. Therefore, we can conclude that the model estimates the inertia of the second area with greater consistency.

The final error visualization method used in this thesis is the cumulative distribution function (CDF) of the absolute error.This function shows the percentage of predictions that have an error less than or equal to a given value [30]. As shown in Figure 6.3, the curves for the two outputs H_1 and H_2 are very similar, indicating comparable model performance for both cases. It is observed that approximately 80% of the predictions have an error of less than 0.4, while for lower error values (< 0.2), H_1 exhibits a slightly higher concentration of predictions with very low error, since its curve envelops that of the H_2 error estimation. Overall, the CDF demonstrates that the model achieves satisfactory accuracy and consistent behavior for both inertia constants.

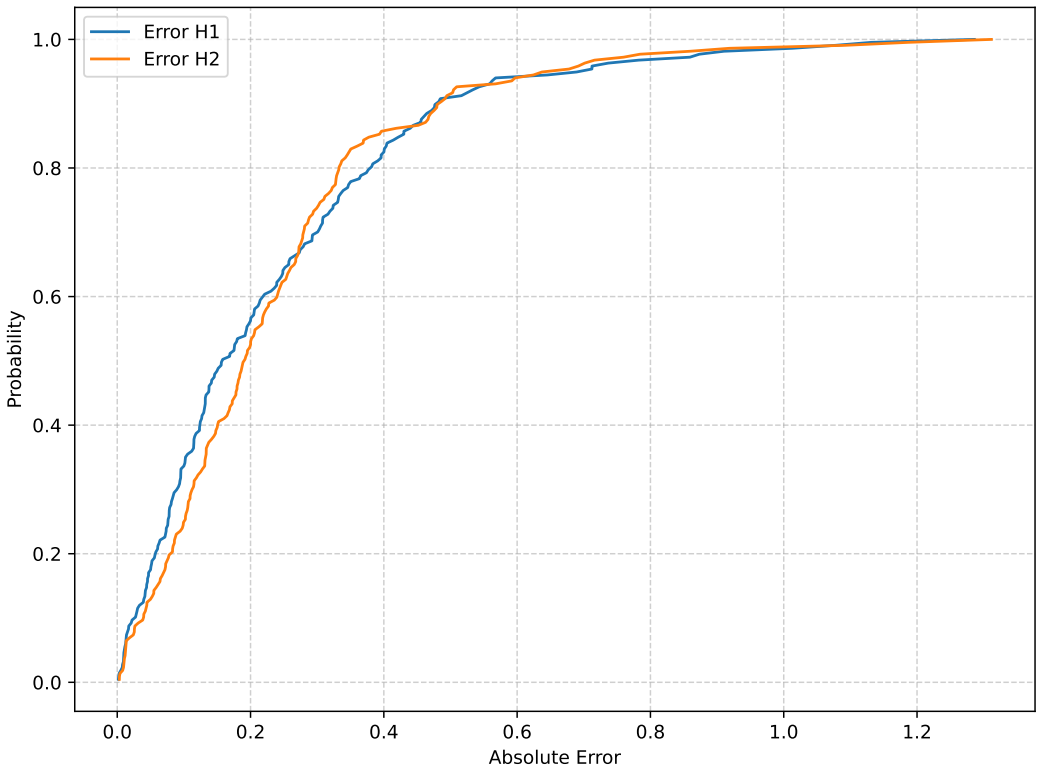

Figure 6.3: Cumulative Distribution Function (CDF) of the absolute prediction error for H_1 and H_2 using Random Forest.

6.2.2 MLP Evaluation

The neural network achieved the highest accuracy, surpassing the performance of every other model we examined.

According to Table 6.2,it is evident that the accuracy achieved by this specific algorithm was excellent, particularly in estimating the constant H_1, where the R^2 exceeds 0.98 and the RMSE and MAE errors remain at low levels. The MAPE for H_1 is only 2.42%, demonstrating exceptional accuracy. Regarding the estimation of inertia H_2, the performance remains very good with an R^2 of approximately 0.94 and a MAPE of 3.39%, yet it is noticeably lower compared to the case of H_1. Overall, the results indicate that the neural network is the most efficient method among the examined algorithms, providing highly reliable predictions.

Below, the visualization diagrams related to this specific model are presented: Figure 6.4 illustrates the deviation between the actual and estimated values for the two inertia constants; Figure 6.5 shows the box plots of the absolute error of the outputs; and finally, Figure 6.6 presents the CDF of the absolute errors for the H_1 and H_2 inertia values.

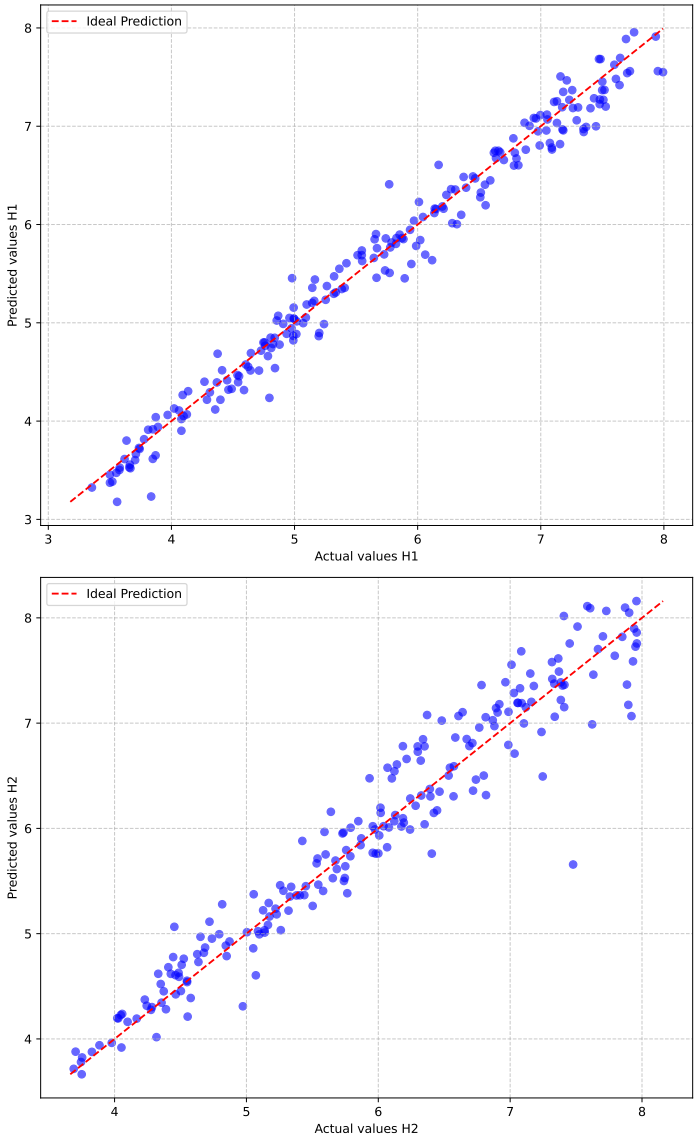

Figure 6.4: Deviation between actual and predicted values of the constants H_1, H_2 from the MLP model.

Table 6.2: Evaluation results of the MLP for the outputs H_1 and H_2.

Output	R^2	**RMSE**	**MAE**	**MAPE (%)**
H_1	0.98178	0.17126	0.13489	2.42
H_2	0.94111	0.28377	0.20352	3.39

Observing Figure 6.4 closely, we note that in the case of this specific algorithm, the blue dots are located at a significantly smaller distance from the line $y = x$ compared to the corresponding dots resulting from the *Random Forest* model. This observation indicates a stronger capability of the neural network to estimate the target inertia values.

According to the *box plot* in Figure 6.5, which corresponds to the MLP model, it is observed that the error range for the inertia of the first area is clearly reduced compared to that resulting from *Random Forest*. This fact indicates greater stability of the MLP in estimating H_1. In contrast, the range of the box corresponding to the second area (H_2) does not show significant variation between the two models. Finally, it is worth noting that the number of outliers, for both the first and the second inertia constant, is noticeably reduced in the case of the MLP, demonstrating the model's greater robustness against extreme deviations.

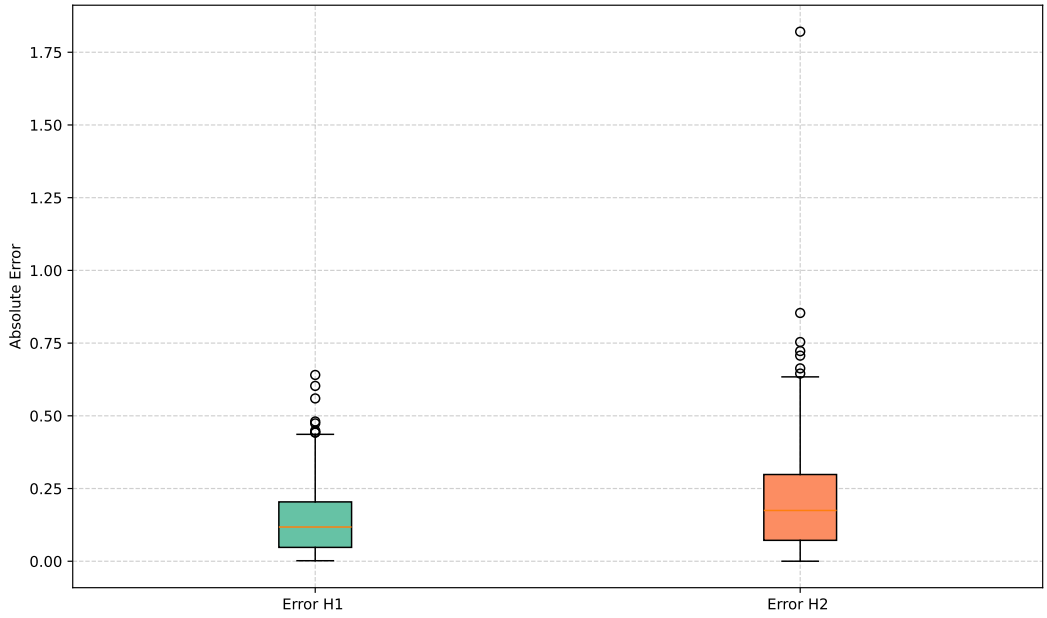

Figure 6.5: Box plot of the H_1, H_2 pair from the MLP model.

The CDF of the absolute error for the MLP model (see Figure 6.6) shows that the estimation of inertia H_1 is more accurate compared to that of H_2. According to the figure 6.6 it is evident that the H_1 curve is consistently positioned higher, which means a larger percentage of errors is concentrated at low values.It is observed that approximately 90% of the errors for H_1 are less than 0.4, while the corresponding percentage for H_2 is lower. This demonstrates that the MLP exhibits increased accuracy and stability in predicting H_1, while the estimation of H_2 remains satisfactory but with greater error dispersion.

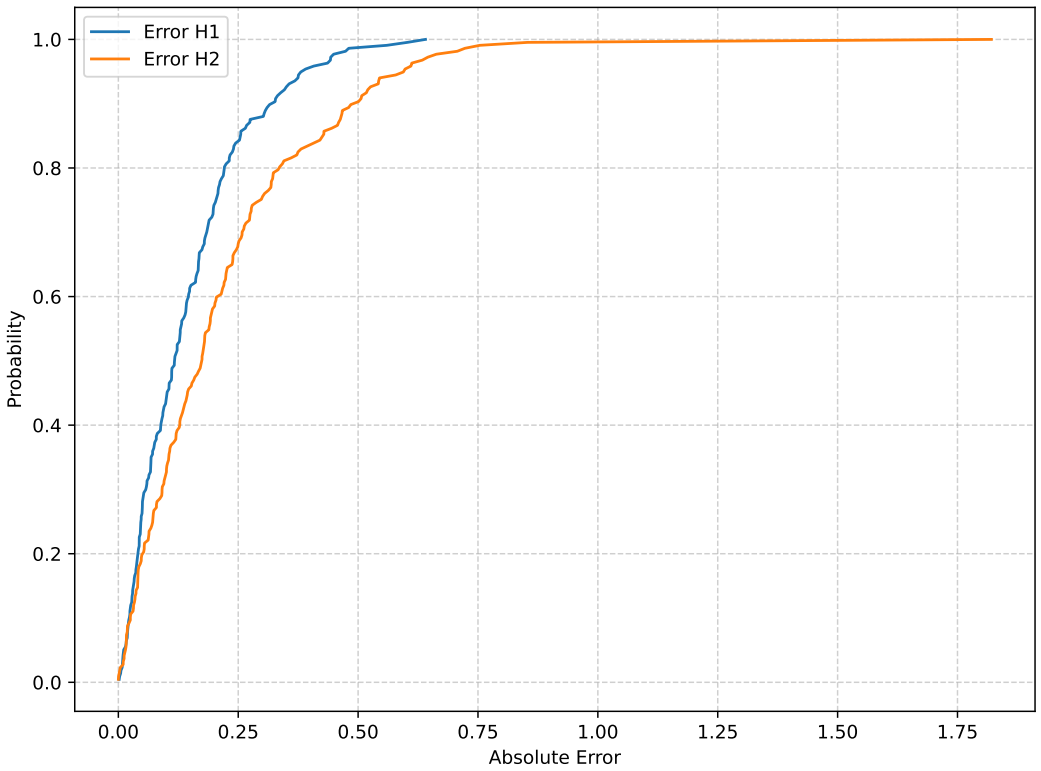

Figure 6.6: CDF of the absolute prediction error from the MLP model for the constants H_1 and H_2.

6.2.3 Evaluation of the Rest Machine Learning Models

A similar approach was used for visualizing the results of the rest machine learning algorithms, which were employed to estimate the values of the inertia constants H_1 and H_2. The accuracy of the calculations for the output pairs varied within a similar range for all algorithms, with minor deviations in each case.

The evaluation metrics for each algorithm are presented in detail for each of the two outputs H_1, H_2 in the following two tables (6.3,6.4) respectively.

Table 6.3: Evaluation results of the machine learning algorithms for output H_1 (Scenario 1).

Algorithm	R^2	**RMSE**	**MAE**	**MAPE (%)**
Random Forest	0.93537	0.32258	0.23332	4.34
MLR	0.82549	0.53008	0.39391	7.57
XGBoost	0.94772	0.30286	0.22047	4.09
SVR	0.92548	0.34640	0.27811	5.33
MLP	0.98178	0.17126	0.13489	2.42

Table 6.4: Evaluation results of the machine learning algorithms for output H_2 (Scenario 1).

Algorithm	R^2	**RMSE**	**MAE**	**MAPE (%)**
Random Forest	0.92647	0.31709	0.23889	4.06
MLR	0.86626	0.42765	0.32525	5.84
XGBoost	0.91648	0.34033	0.25008	4.21
SVR	0.90838	0.35395	0.24953	4.42
MLP	0.94111	0.28377	0.20352	3.39

From the results, it is observed that the MLP algorithm demonstrates the best performance for both outputs, with very high R^2 values and low errors (RMSE, MAE, MAPE). The Random Forest and XGBoost algorithms follow with similar and relatively satisfactory results, while SVR exhibits slightly lower accuracy. Finally, Linear Regression records the lowest performance compared to the other models, confirming its limited ability to capture the complexity of the problem.

The algorithm evaluation results can be grouped by metric and presented graphically in Figure 6.7. Each subplot displays one evaluation metric, and for each algorithm, its performance is shown for both output H_1 and output H_2, facilitating with that way a direct comparative evaluation.

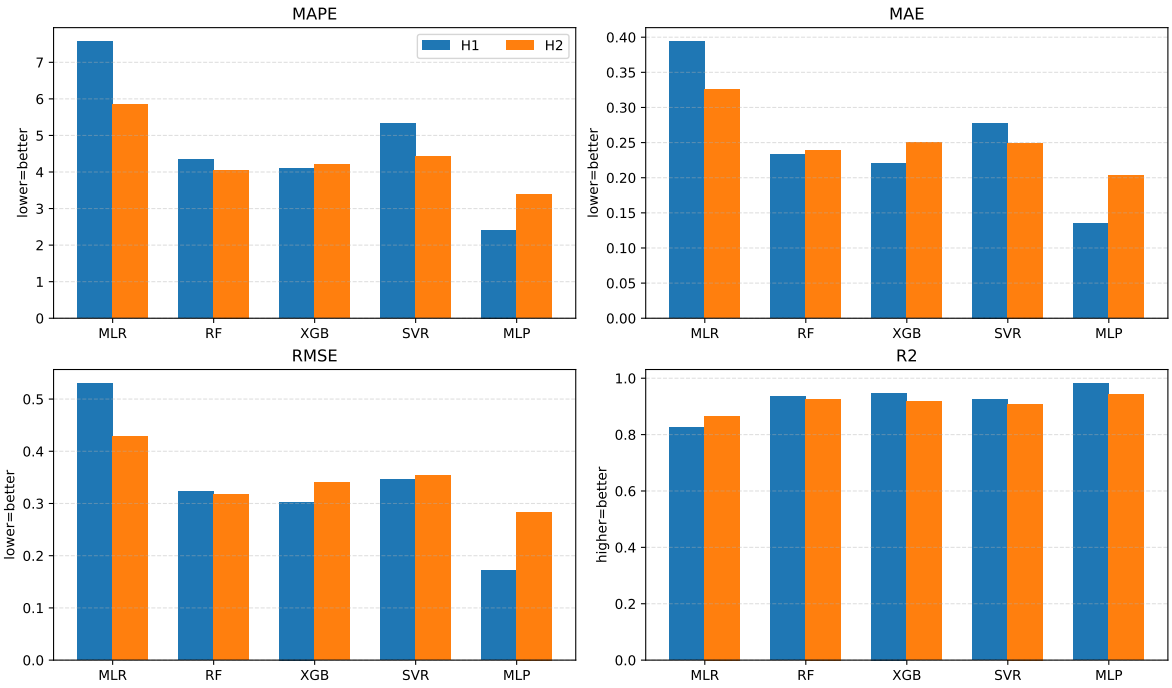

Figure 6.7: Comparison of model performance per metric for Scenario 1.

6.3 Results - Scenario 2

Similar visualization strategies were also applied to Scenario 2. However, while observing the evaluation metric values for the two outputs in Tables 6.5 and 6.6, an unusual behavior is noted across all algorithms. While the models' performance for output H_1 is excellent and clearly improved compared to Scenario 1, demonstrating very high R^2 values (> 0.99) and exceptionally low errors (e.g., MAPE $< 2\%$ for MLP, Random Forest, MLR), they simultaneously fail to approximate with similar accuracy the relationship between the input data and the inertia of the second area (H_2). The performance for H_2 records, in the majority of cases, errors ranging from double to quadruple the values compared to those of H_1. This asymmetry in predictive capability between the two areas suggests an inherent complexity in the system dynamics associated with the second area, which is not equally easily detected by the models.

Despite these significant discrepancies in prediction accuracy between the two outputs, the MLP and Random Forest models once again produce the most reliable and balanced results for both areas, maintaining a high R^2 and the lowest errors among all algorithms.

In Figure 6.8, the imbalance in the models' estimation accuracy between the H_1 and H_2 outputs becomes even more evident across each metric.

Table 6.5: Evaluation results of the machine learning algorithms for output H_1 (Scenario 2).

Algorithm	R^2	**RMSE**	**MAE**	**MAPE (%)**
Random Forest	0.99311	0.1237	0.09672	1.88
MLR	0.99228	0.13094	0.09973	1.91
XGBoost	0.99182	0.13234	0.10656	2.16
SVR	0.98124	0.20411	0.15570	3.01
MLP	0.99495	0.10593	0.08187	1.53

Table 6.6: Evaluation results of the machine learning algorithms for output H_2 (Scenario 2).

Algorithm	R^2	**RMSE**	**MAE**	**MAPE (%)**
Random Forest	0.96067	0.29017	0.22586	4.57
MLR	0.90431	0.45263	0.35163	7.23
XGBoost	0.91417	0.41129	0.26915	5.39
SVR	0.94757	0.33506	0.25895	5.28
MLP	0.96011	0.29223	0.22541	4.67

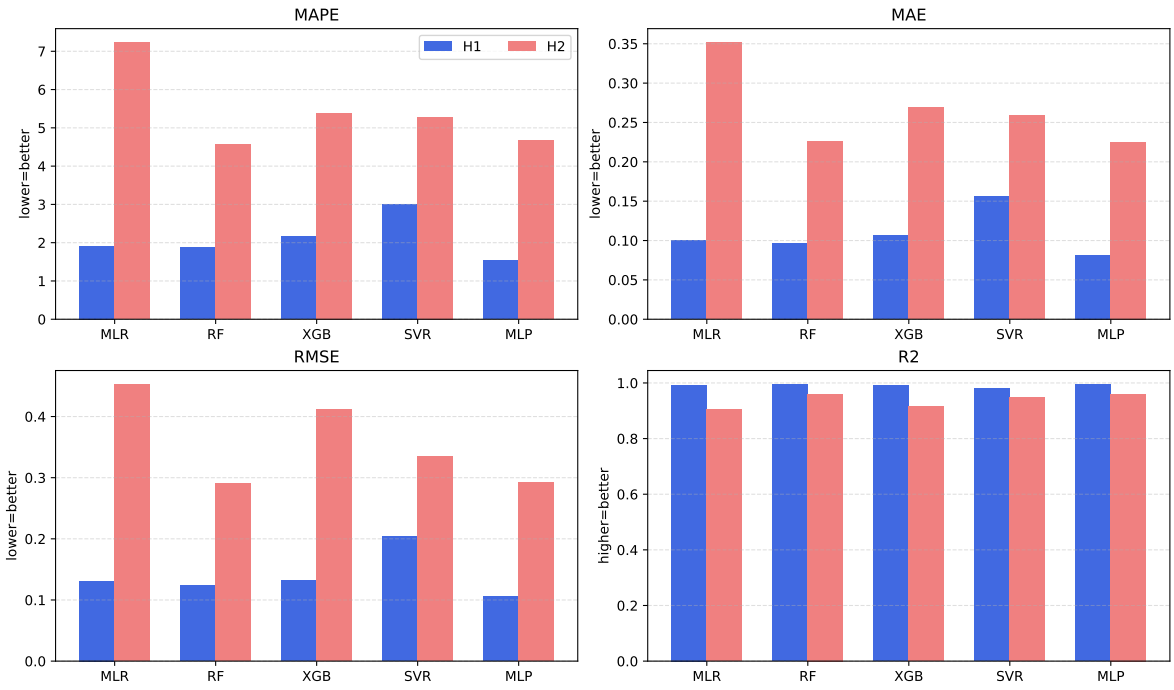

Figure 6.8: Comparison of model performance per metric for Scenario 2.

6.4 Results - Scenario 3

The results of Scenario 3, as shown in Tables 6.7 and 6.8, reveal that the model errors are noticeably smaller compared to the previous scenarios, while at the same time a balance in performance is observed between the two outputs H_1 and H_2. The evaluation metrics values for the two areas are very close to each other, indicating that the models now understand and predict the dynamics of both inertia constants with similar accuracy.

This behavior is directly linked to the existence of a balanced generation mix in the two areas, thereby revealing that the mix of generating units also plays a significant role in the accuracy of inertia estimation. MLP once again maintains its leading position as the most powerful algorithm, recording the optimal values in the evaluation metrics for both outputs, with the MAPE for inertia H_2 even reaching as low as 2.14%. The visualization of the metric values for this specific scenario is shown in Figure 6.9.

Algorithm	R^2	**RMSE**	**MAE**	**MAPE (%)**
Random Forest	0.98337	0.19216	0.15073	2.85
MLR	0.97857	0.21816	0.17489	3.56
XGBoost	0.97561	0.22851	0.17525	3.52
SVR	0.96895	0.26258	0.21222	4.29
MLP	0.98668	0.17198	0.13422	2.56

Table 6.7: Evaluation results of the machine learning algorithms for output H_1 (Scenario 3).

Algorithm	R^2	**RMSE**	**MAE**	**MAPE (%)**
Random Forest	0.98119	0.20070	0.15827	3.05
Multiple Linear Regression	0.97770	0.21850	0.17698	3.42
XGBoost	0.97957	0.20069	0.16164	3.06
SVR	0.97115	0.24855	0.19567	3.89
MLP	0.99014	0.14531	0.11451	2.14

Table 6.8: Evaluation results of the machine learning algorithms for output H_2 (Scenario 3).

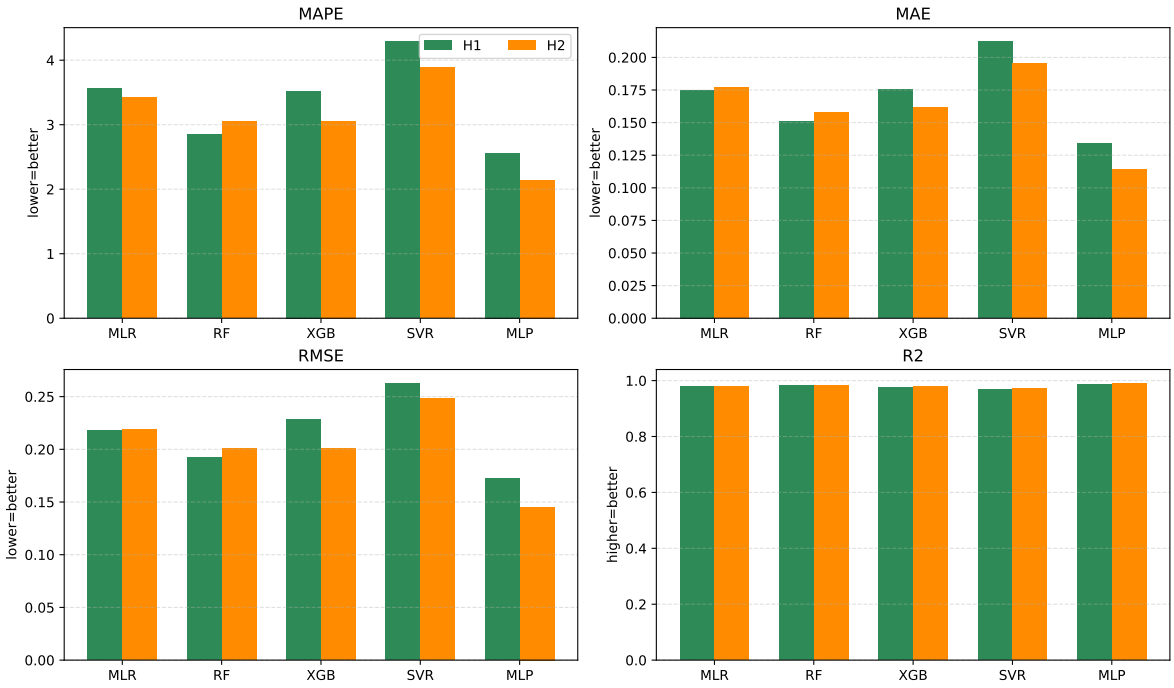

Figure 6.9: Comparison of model performance per metric for Scenario 3.

6.5 Online Inertia Estimation

The development and training of the algorithms presented in this thesis enable the estimation of power system inertia in real-time, using data from system dynamic responses. Specifically, after evaluating the algorithms and concluding that the most reliable machine learning algorithm is the MLP, we propose the development of a real-time inertia estimation system.

The proposed system will be based on a network of smart meters that will record the required dynamic responses ($\Delta f_1, \Delta f_2, \Delta P_{\text{tie}}$) every 90 seconds. These meters will communicate via a secure protocol with the Control Center (CC).

At the CC, a central server will host the trained MLP model. The server will receive the frequency and pie-tie response data from the grid, preprocess it, and feed it into the machine learning model, calculating the corresponding inertia values every 90 s.

It should be emphasized that this process is executed continuously. Every 90 s, the server at the CC receives new data, updating the inertia estimation on an ongoing basis. This enables live monitoring of the power system and immediate adjustment of the estimates based on the latest available values. This method constitutes an online inertia estimation, as the estimates are updated continuously and provide an accurate representation of the system's state at any

given time,(see Figure 6.10).

In this way, continuous and accurate real-time monitoring of system inertia becomes feasible, being a critical tool for the management and control of modern power systems. However, it must be emphasized that for our trained model to function and accurately determine the pair of output inertia constants, the repeated application of a rectangular power pulse at periodic time intervals is required. This pulse will modify the power demand in Area 1, as analyzed in Chapter 5, and will always be applied from the 20th to the 40th second of each recording period.

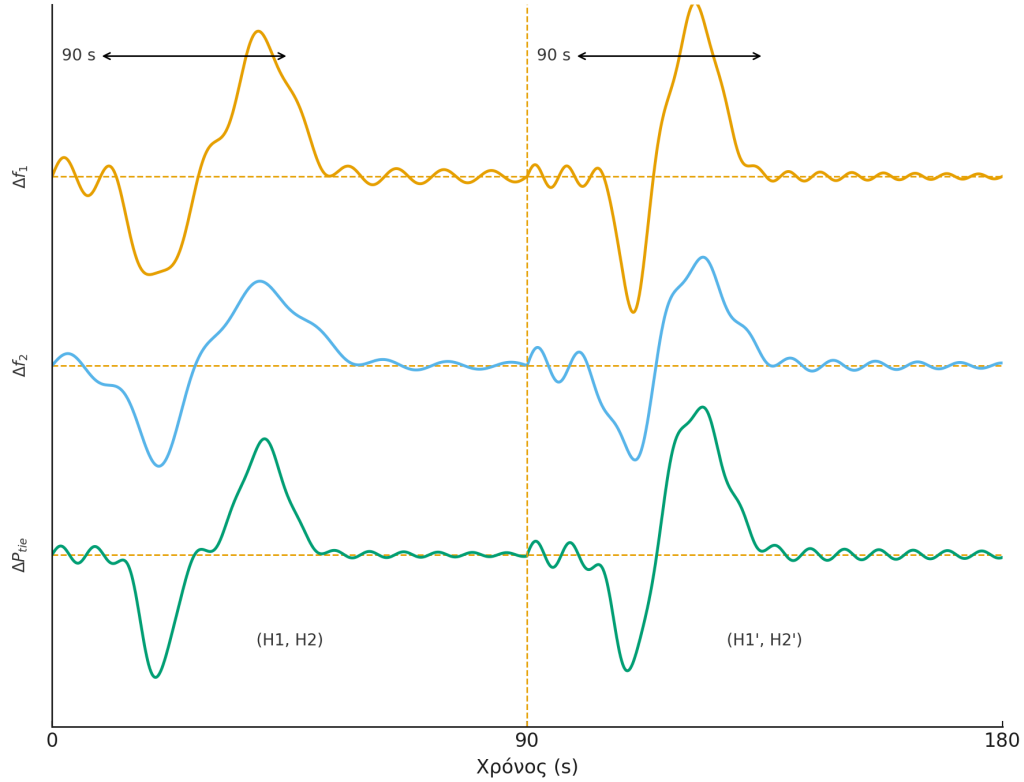

Figure 6.10: Online estimation of the inertia pair.

Chapter 7

Extension of the Models to a Three-Area Interconnected System

The fundamental principles and strategies applied in Chapters 4, 5, and 6 to the two-area interconnected system are extended in this chapter to a more extensive scenario, this time consisting of three interconnected areas. As expected, the size of data, the computational power requirements, and the complexity of the problem increase significantly. Nevertheless, the analysis of a three-area system is of particular interest, as it allows for an in-depth understanding of the dynamic interaction between the areas. In fact, this specific configuration more accurately approximates the real-world conditions encountered in modern interconnected power systems, where complexity and interdependence are fundamental characteristics (e.g., the North American Grid with 4 interconnecred areas [2]).

7.1 Simulink Implementation

There are two possible ways to interconnect three areas, which are illustrated in Figure 7.1. In the first scenario, each area is directly connected to all others, while in the second , Area 1 acts as a link between Area 2 and Area 3. For the purposes of this thesis, the more complex case was examined;the one of the full interconnection of all areas.

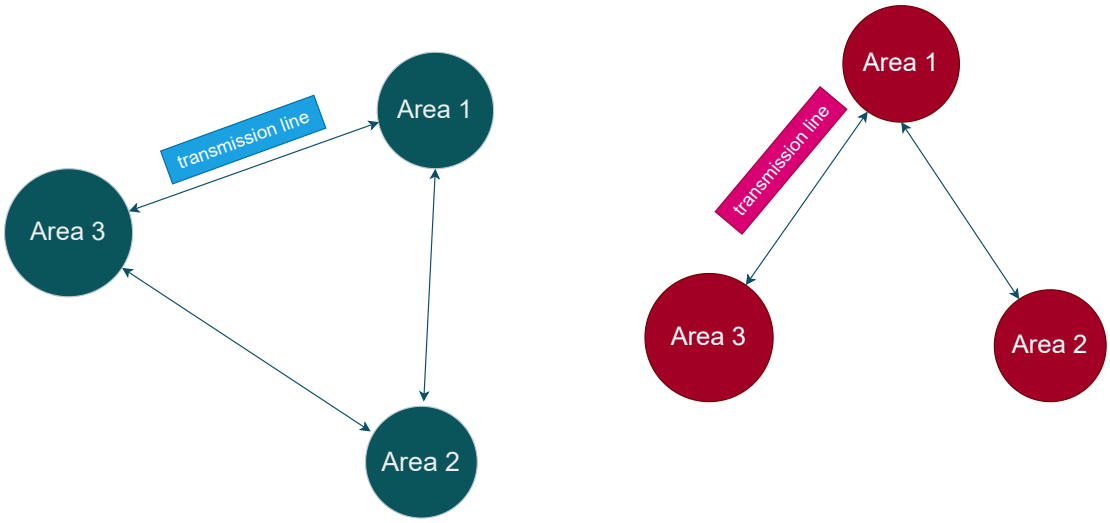

Figure 7.1: Three-area interconnection scenarios.

7.1.1 Basic Block Diagram of Three Interconnected Areas.

The modeling of the three interconnected areas in *Simulink* is just an extension of Scenario 1. In this case, instead of a set of subsystem pairs, as analyzed in Chapter 4,each corresponding to its own area,we now have a set of subsystem triplets. The generating units in areas 1 and 2 remain the same as those in Scenario 1, while in area 3, three generating units are implemented. Among these, two are new types that simulate the dynamic behavior of a gas turbine and a biomass unit, aiming to increase the diversity of the generation mix in order to more realistically approximate a modern power system. The third unit is a single-reheat steam turbine. The basic system layout is illustrated in Figure 7.2 [31].

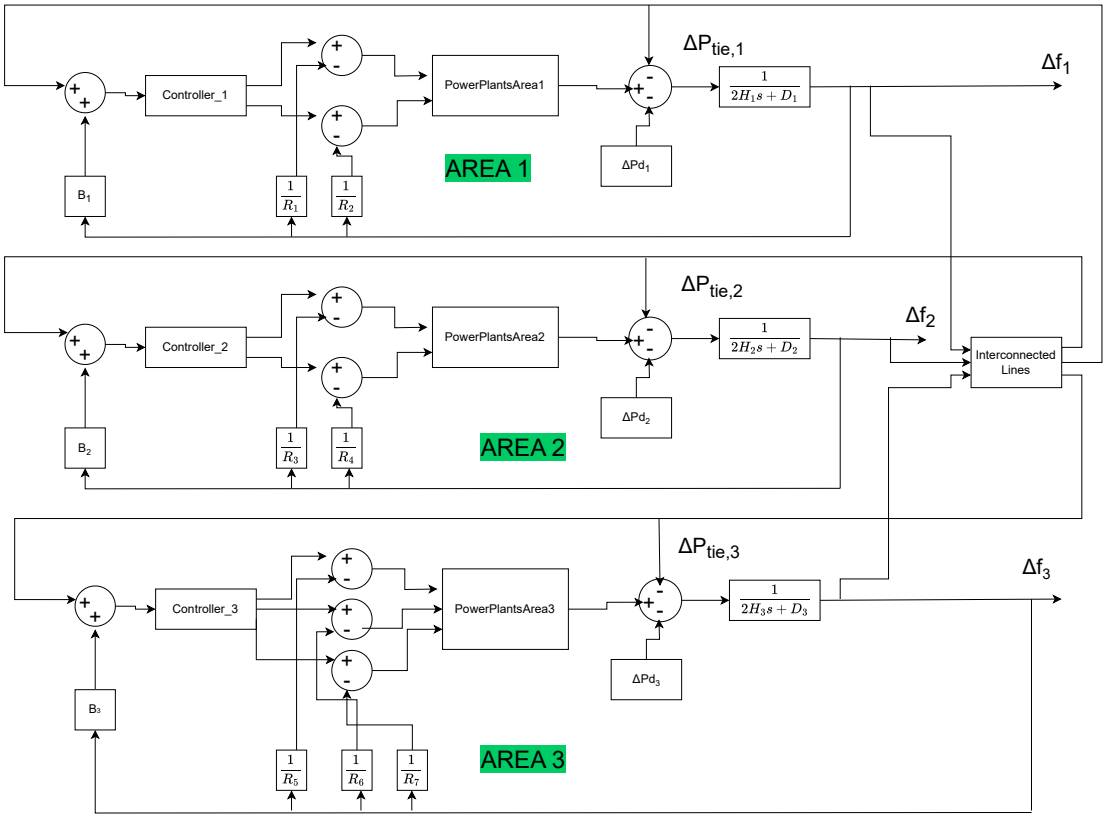

Figure 7.2: Basic block diagram of three interconnected areas.

Another significant difference from Scenario 1, in the construction of the Simulink models and,and as a result in the training of the machine learning algorithms as well, is the use of batteries to generate the probing signal in both Area 1 and Area 3. This differentiation was deemed necessary because an artificial pulse in the power demand of Area 1 proved insufficient to mitigate the influence of the stochasticity from the real power demand variation. As a result, the machine learning models were unable to learn the relationship between the input and output. Consequently, probing signals are applied using batteries in both Area 1 and Area 3.

7.1.2 Interconnected Lines Subsystem

Beyond the differences from Scenario 1 like the adoption of a battery configuration in two different areas and, obviously, the addition of a third area (Area 3) a careful examination of the layout in Figure 7.2 reveals a new subsystem named ”Interconnected Lines”. This specific implementation encompasses all the transmission line interconnections between the three areas, ultimately connecting each area to its respective tie-line ($\Delta P_{tie,1}$, $\Delta P_{tie,2}$, $\Delta P_{tie,3}$). In

Figure 7.3, we can see the complete set of internal connections housed within this subsystem. The utility of this subsystem lies exclusively in simplifying the overall diagram and enhancing its comprehensibility, allowing for a more elegant, clear, and organized representation of the three interconnected areas.

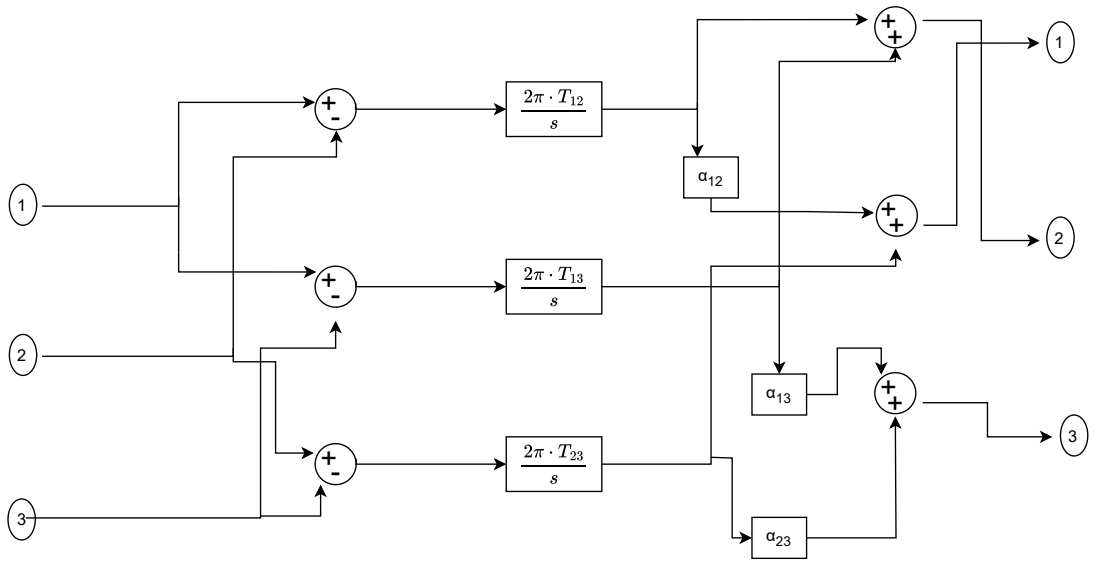

Figure 7.3: Implementation of the Interconnected Lines subsystem.

7.1.3 Power Plants 3 Subsystem

The Power Plants subsystem for Area 3 is shown in Figure 7.4. Upon observation, somebody can distinguish three linear transfer functions connected in parallel, which try to imitate the dynamic behaviour of the three types of generating units that constitute this area.

The first unit corresponds to the modelling of a biomass unit, the implementation of which is identical to that of a single-reheat steam turbine. The difference in its dynamic behavior is found exclusively in the different value ranges taken by the parameters that characterize it.

The second generating unit, also shown in Figure 7.4, simulates the dynamic behavior of a Gas Turbine. A gas turbine uses gases (e.g., atmospheric air, CO_2, or He) as its combustion medium instead of the water utilized in steam turbines. The air enters through the various intake ports, is compressed, ignited in the combustion chamber, and the mechanical work is produced through the expansion of the exhaust gases (Brayton thermodynamic cycle). In the linear model that simulates the gas turbine's operation, we encounter a set of parameters. Among these, the constant c represents the valve positioner constant, while the parameters b,

X, Y, T_F, T_{CR}, and T_{CD} correspond to time constants that characterize the system's dynamic response [6].

Lastly, we meet again a unit that corresponds to the modeling of a single-reheat steam turbine,still one of the most prevalent types of generating units in modern power grids. The parameter T_{g5} represents the time constant of the turbine governor, while the parameter T_{ch5} corresponds to the time constant of the high-pressure chamber. As already mentioned in Chapter 4, the values of the parameters used for modeling the respective units can be differentiated to more realistically approximate actual grid operating conditions.

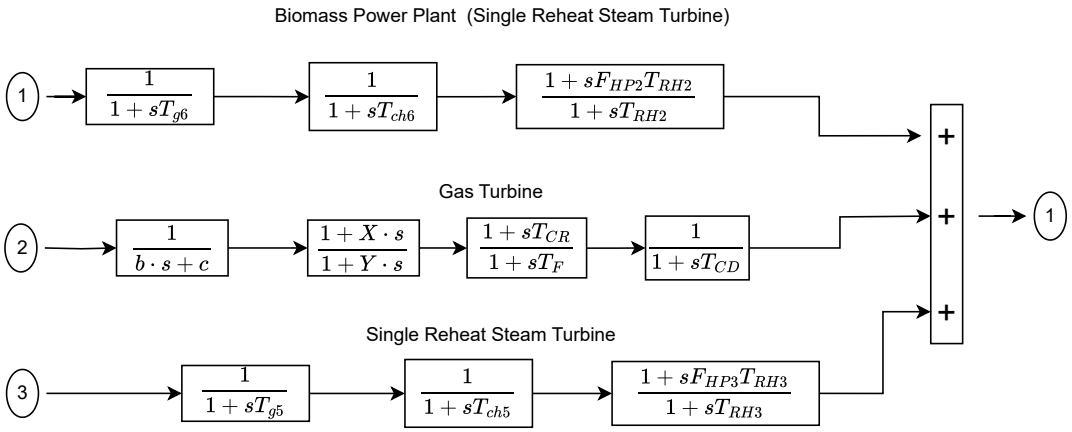

Figure 7.4: Power Plants 3 Subsystem.

The numerical values of the parameters constituting Area 3, combined with the values of the constants found in the interconnected lines subsystem, are summarized in Table 7.1.

Table 7.1: Numerical values of the parameters for Area 3 [6].

Parameter	**Value**	**Unit**
T_{g5}	0.1	s
T_{ch5}	0.4	s
F_{HP3}	0.35	–
T_{RH3}	5	s
c	1	–
b	0.05	s
X	0.6	s
Y	1	s
T_F	0.23	s
T_{CR}	0.01	s
T_{CD}	0.2	s
T_{g6}	0.08	s
T_{ch6}	0.3	s
T_{RH2}	10	s
F_{HP2}	0.3	–
T_{12}	2	s
T_{13}	2	s
T_{23}	2	s
α_{12}	-1	–
α_{13}	-1	–
α_{23}	-1	–
K_{I5}	0.25	–
K_{I6}	0.25	–
K_{I7}	0.25	–
R_5	0.1	p.u.
R_6	0.1	p.u.
R_7	0.1	p.u.
B_3	31.3	p.u.
D_3	1.3	p.u.

7.2 Data Preparation for the Machine Learning Algorithms

7.2.1 Data Collection

Similar steps to those in the previous chapters were followed for the collection of data regarding the three-area power system. In this version of the problem, the generated data include the frequency dynamics of Areas 1, 2, and 3 (Δf_1, Δf_2, Δf_3), as well as the changes in the power flows on the tie-lines between the three areas.

Basically, these are the change in power flowing between Area 1 and the other two areas ($\Delta P_{tie,1}$), the change in power flowing between Area 2 and Areas 1 and 3 ($\Delta P_{tie,2}$), and the change in power between Area 3 and the other two areas ($\Delta P_{tie,3}$). The time duration of each response time series remains at 90 seconds, and the goal is now the estimation of the three inertia constants (H_1, H_2, H_3). In this case, the machine learning models analyzed in the previous chapters will have six different dynamic responses as inputs (three frequency dynamic responses and three dynamic responses from the tie-line powers).

While keeping the parameter values from Table **??** constant, we first store the six system responses in a tensor arrangement. These responses result from the change in power demand in each area for every unique combination of values for the inertia constants H_1, H_2, and H_3. The three-area control system, for random values of H_1, H_2, H_3 all within the range [3, 8], is triggered as mentioned earlier by a combination of random power disturbances and controlled disturbances (probing signals) from batteries. The battery probing signals are applied to Area 1 and Area 3.

Since the system output time series are sampled at 100 Hz (one sample every 0.01 seconds), they are stored in a tensor whose size has doubled compared to that in Figure 5.5. The dimensions of the new tensor containing the entire set of time series will be:

$$X \in \mathbb{R}^{1000 \times N \times 6},$$

where $N = 9001$ is the number of samples in each time series and the last axis corresponds to the six inputs ($\Delta f_1, \Delta f_2, \Delta f_3, \Delta P_{tie,1}, \Delta P_{tie,2}, \Delta P_{tie,3}$). The number of different triples (H_1, H_2, H_3) whose effect on the system output is examined was decided to be 1000 again. Correspondingly, the output variables, i.e., the triples of inertia constants, are represented as a two-dimensional matrix:

$$Y \in \mathbb{R}^{1000 \times 3},$$

where each row contains the corresponding triple (H_1, H_2, H_3) responsible for the six time series in the same row of matrix X.

7.2.2 Data Preprocessing

After the successful collection and storage of the data, we proceed to their preprocessing. The process does not differ at all from the one described in detail in Chapter 5. Initially, statistical features are extracted for each time series (mean value, standard deviation, maximum value, nadir, RMS, and time to nadir), followed by the construction of a new two-dimensional matrix that will host these statistical features for each time series. Being more precise, this matrix will contain a total of 45 columns, of which 42 will correspond to the descriptive metrics of each time series and the remaining three to the output triple (H_1, H_2, H_3). The number of rows in the matrix remains equal to 1000, exactly the total number of different output combinations. In mathematical terms, this translates as follows:

$$K \in \mathbb{R}^{1000\times 45}, \quad D = [X \mid Y],$$

where

$$X \in \mathbb{R}^{1000\times 42}$$

represents the set of statistical indicators describing the behavior of the input time series, while

$$Y \in \mathbb{R}^{1000\times 3}$$

contains the output variables, that is, the triple of inertia constants (H_1, H_2, H_3).

After that, the dataset is ”cleaned” by removing unstable time series and the corresponding output inertia triples that cause this behavior. To identify them, we once again resort to calculating the $\mathrm{RMS}_{\mathrm{ratio}}$ metric for each of the 6 time series, the value of which will classify them as either unstable or stable signals (threshold 2.5). In total, 75 entries are rejected from the dataset, which correspond to unstable signals and the inertia triples that cause them, while the initial dataset has already been expanded with the 6 columns containing the $\mathrm{RMS}_{\mathrm{ratio}}$ values for all 6 time series.

Subsequently, an additional filtering of the dataset is performed, this time by selecting the columns with the highest contribution to determining the final output. The Pearson correlation coefficient is again used as the tool to identify the degree of influence of each feature on each

output, with a minimum acceptable value of 0.3. If a feature does not meet this condition for any of the three outputs, its entire column is removed from the final dataset.

According to the correlation matrix in Figure 7.5, a total of 18 columns are removed from the dataset, which corresponded to the following features:

$$
\begin{aligned}
&df1\_mean,\ df1\_nadir\_time,\\
&df2\_mean,\ df2\_nadir\_time,\\
&df3\_mean,\ df3\_nadir\_time,\\
&ptie1\_mean,\ ptie1\_std,\ ptie1\_rms,\ ptie1\_nadir\_time,\\
&ptie2\_mean,\ ptie2\_std,\ ptie2\_rms,\ ptie2\_nadir\_time,\\
&ptie3\_mean,\ ptie3\_std,\ ptie3\_rms,\ ptie3\_nadir\_time.
\end{aligned}
$$

where *ptie1, ptie2, ptie3* denote the tie-line power variations between the network and each individual area.

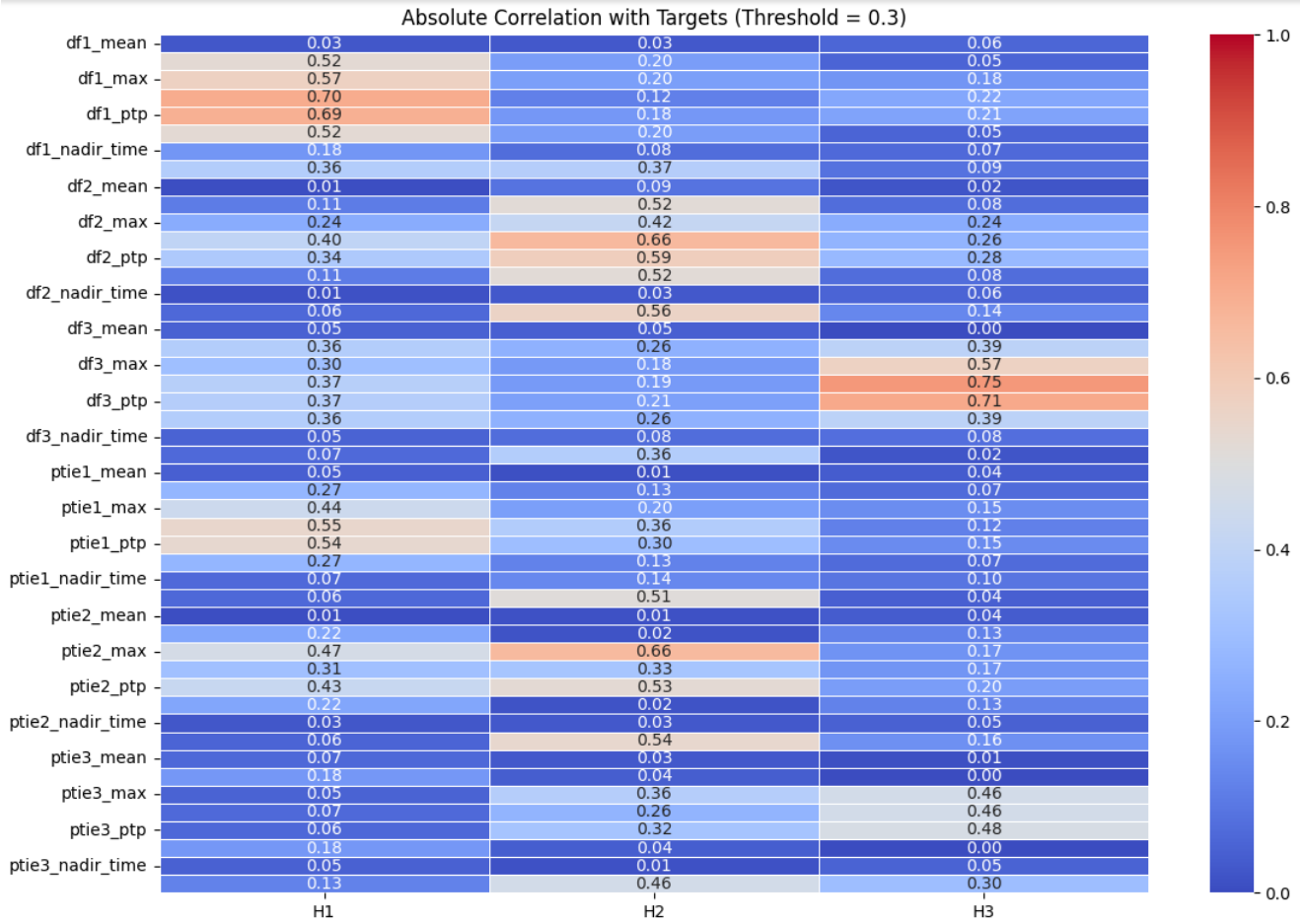

Figure 7.5: Correlation matrix for the three-area scenario.

The next step is the scaling process, where the *Min-Max Scaler* method is selected, as in Scenario 1 of the two-area interconnected system. This was chosen because, after a series of experiments with different scaling methods, it proved once again to be the most effective in accurately determining the outputs.

The final step before the data are fed into the machine learning models is the splitting of the dataset into training and validation sets, followed by organizing them into input variables and target variables. The input variable matrix (X_{train}) has 30 columns, while the matrix containing the output variables has a total of 3 columns.

7.3 Machine Learning Algorithm Evaluation

For estimating the inertia of the three-area power system, the same five machine learning algorithms examined in the previous chapters were used : Random Forest, MLR, SVR, XGBoost, and MLP. However, in contrast to the scenarios studied in earlier chapters, the accuracy in the values of the three output inertia constants was not satisfactory for the majority of the models, with the sole exception of the MLP model,which achieved once again exceptional accuracy in estimating the three output variables.

The performance of both the MLP and the Multilinear Regression (MLR) algorithm which proved to be the weakest method for approximating the outputs is visualized in the subsequent part of this section (Figures 7.6, 7.7, 7.8, 7.9, 7.10, 7.11). At the end of this section, the evaluation metrics for all algorithms for each of the target variables (H_1, H_2, H_3) are summarized in tables and diagrams (7.4, 7.5, 7.6, 7.12).

7.3.1 MLR Algorithm Evaluation

The evaluation metrics for the prediction accuracy of each of the three system outputs, using the MLR algorithm, are summarized in Table 7.2. We observe that the MLR algorithm achieves quite high accuracy regarding the estimation of the inertia value for Area 2, with the R^2 metric reaching very high levels (above 0.96), indicating an excellent ability to capture the correlation between the input variables and this specific model output.Furthermore, the RMSE and MAE errors remain exceptionally low, highlighting the model's accuracy in its estimations. Lastly, the MAPE metric remains below 5%, an error percentage which is generally considered acceptable in inertia estimation applications [32], indicating that the method

exhibits high reliability and accuracy in predicting this specific output.

However, for the other two areas, the resulting deviations are significant (MAPE > 10%), rendering the model overall weak and unsuitable for handling the specific nature of the data and producing valid predictions for these outputs.

Table 7.2: Evaluation results of the MLR algorithm for the outputs H_1, H_2 and H_3.

Output	R^2	**RMSE**	**MAE**	**MAPE (%)**
H_1	0.72639	0.73961	0.53768	11.23
H_2	0.96251	0.27483	0.20851	4.10
H_3	0.75632	0.72430	0.54063	11.04

Figure 7.6 illustrates the deviation between the actual and the estimated values resulting from the MLR model for each of the system outputs, while Figure 7.7 presents the box plots for each of the inertia constants of the three areas (H_1, H_2, H_3). Lastly, Figure 7.8 represents the cumulative distribution function (CDF) plots of the absolute error for each of the three estimated output values.

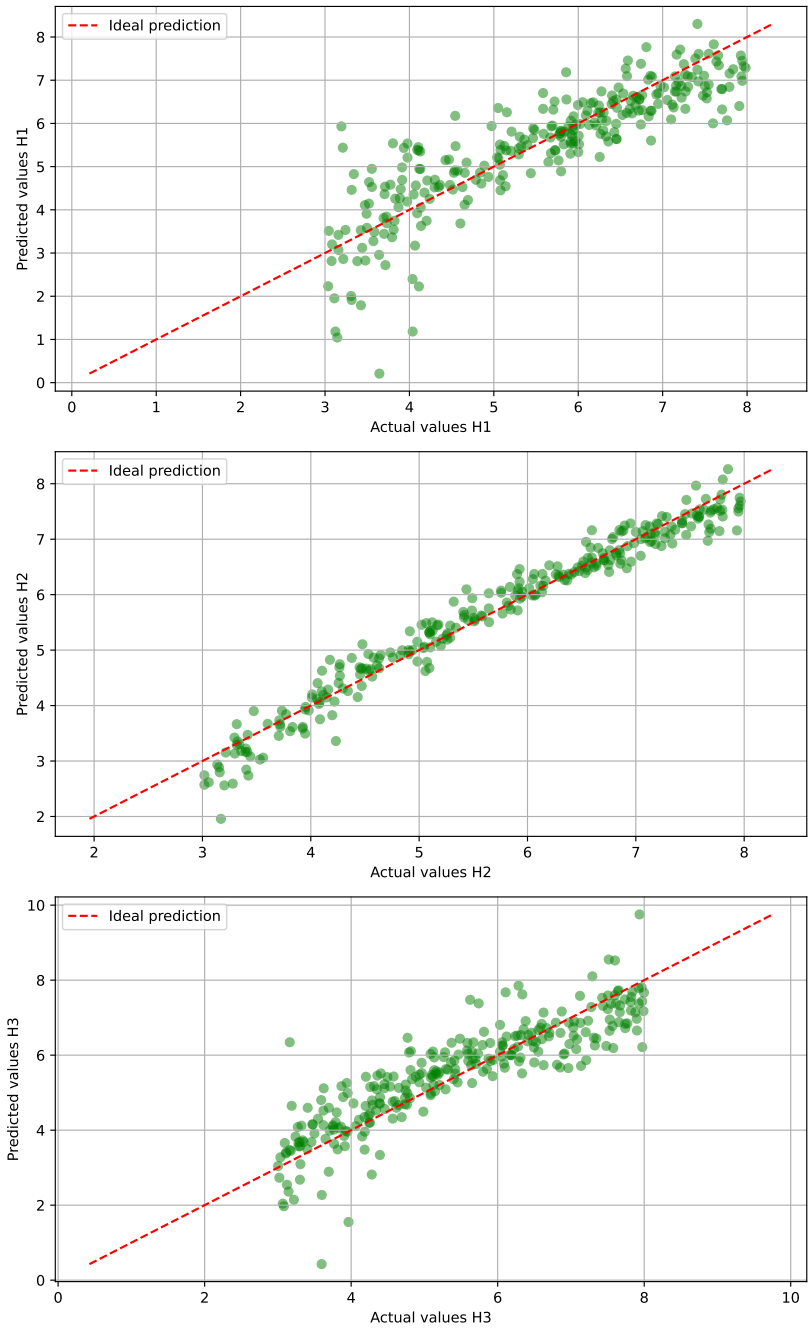

Figure 7.6: Deviation of actual from estimated values of the constants H_1, H_2, H_3 by the MLR model.

Figure 7.7: Box plot of the absolute prediction error of the MLR model for the inertia constants H_1, H_2 and H_3.

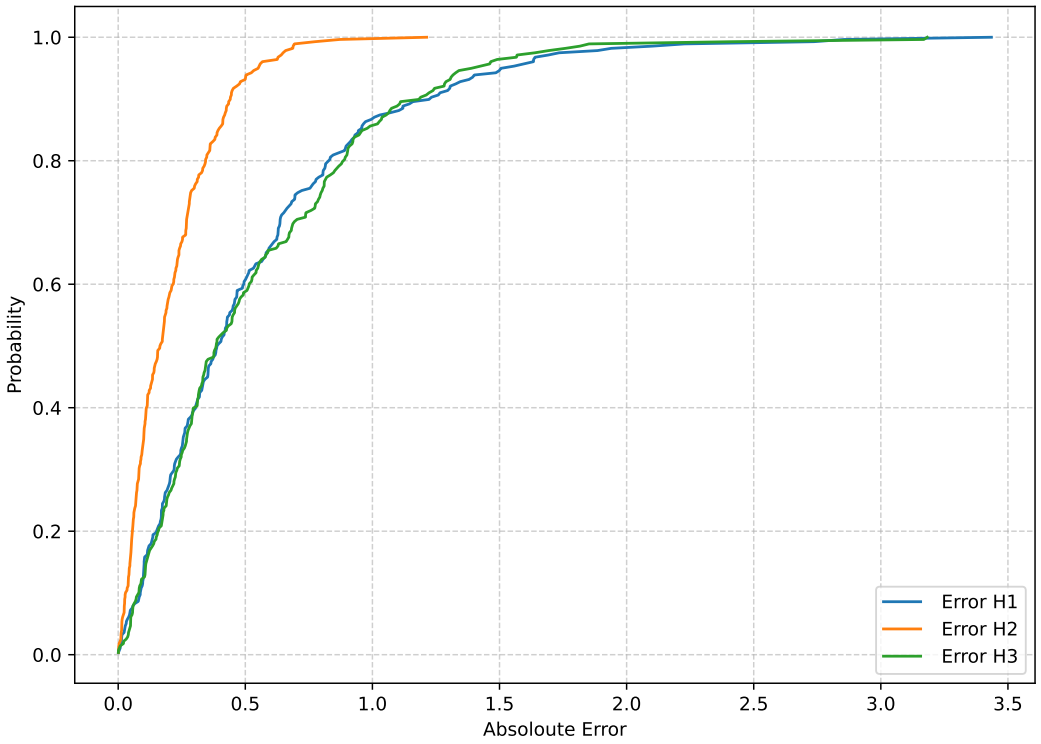

Figure 7.8: CDF of the absolute prediction error of the MLR model for the constants H_1, H_2 and H_3.

These three figures confirm the results of Table 7.2, with the inertia of Area 2 being the only one estimated with high accuracy by this specific model, while the errors in estimating the inertias H_1 and H_3 are unacceptable.

7.3.2 MLP Algorithm Evaluation

The metric values for the MLP algorithm appear to be significantly improved, as shown in Table 7.3. We observe that, in contrast to multilinear regression, the MLP algorithm achieves exceptional accuracy for all three areas of the system. Specifically, the R^2 metric reaches very high levels (above 0.96 for all outputs), indicating the MLP's ability to accurately capture the correlation between the input variables and the model outputs. Furthermore, the RMSE and MAE errors remain very low, while the MAPE does not exceed 4%, a fact which demonstrates the model's reliability in its predictions.

Table 7.3: Evaluation results of the MLP algorithm for the outputs H_1, H_2 and H_3.

Output	R^2	**RMSE**	**MAE**	**MAPE (%)**
H_1	0.96842	0.25532	0.17874	3.22
H_2	0.98268	0.18651	0.14908	2.77
H_3	0.96509	0.27971	0.20856	3.91

Figures 7.9, 7.10, and 7.11 visualize the deviation between the actual and predicted inertia values for each area, the box plots of the three inertia constants, and the cumulative distribution function (CDF) plots of the absolute prediction error for each of the three estimated outputs, respectively.

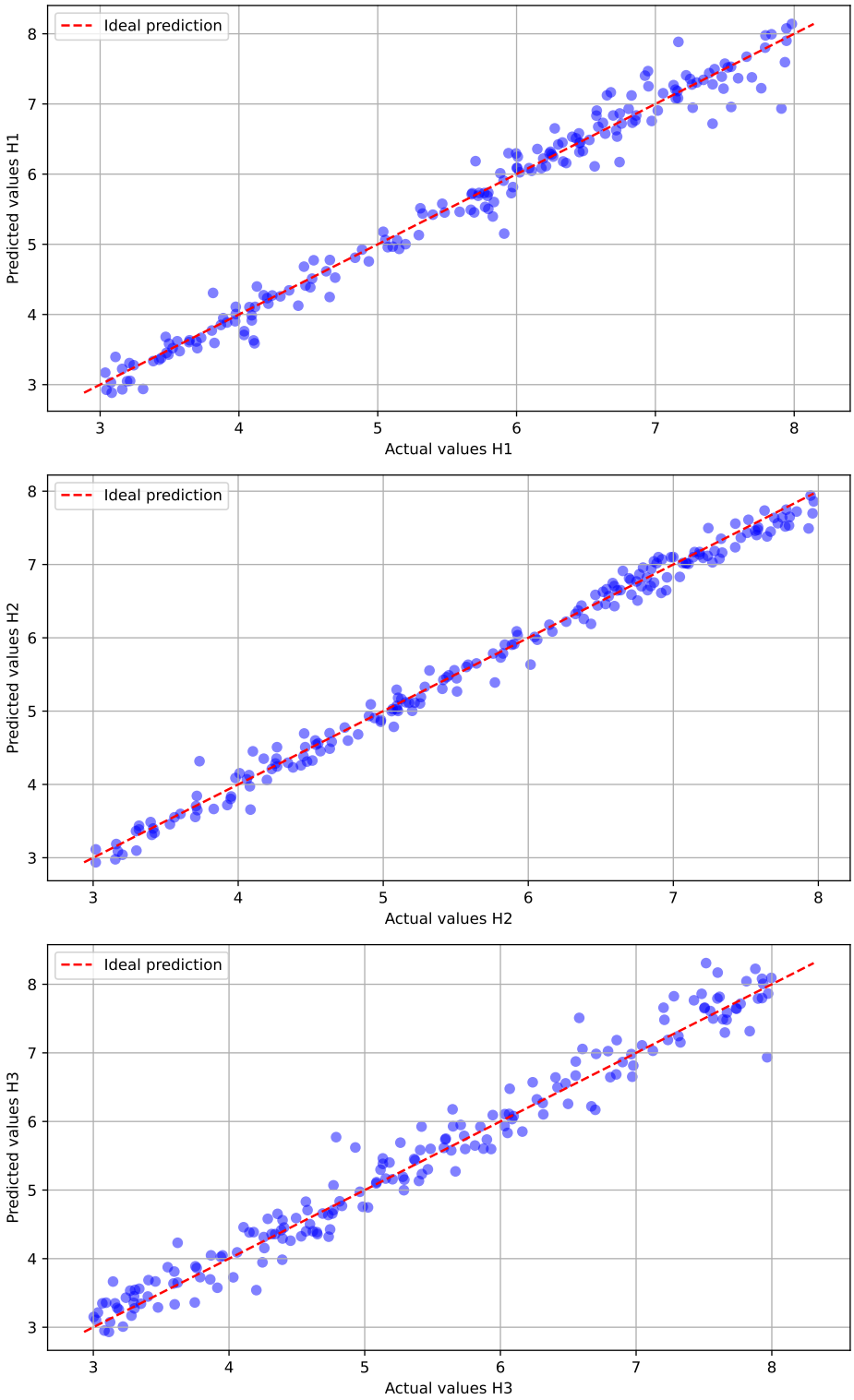

Figure 7.9: Deviation of actual from estimated values of the constants H_1, H_2, H_3 by the MLP model.

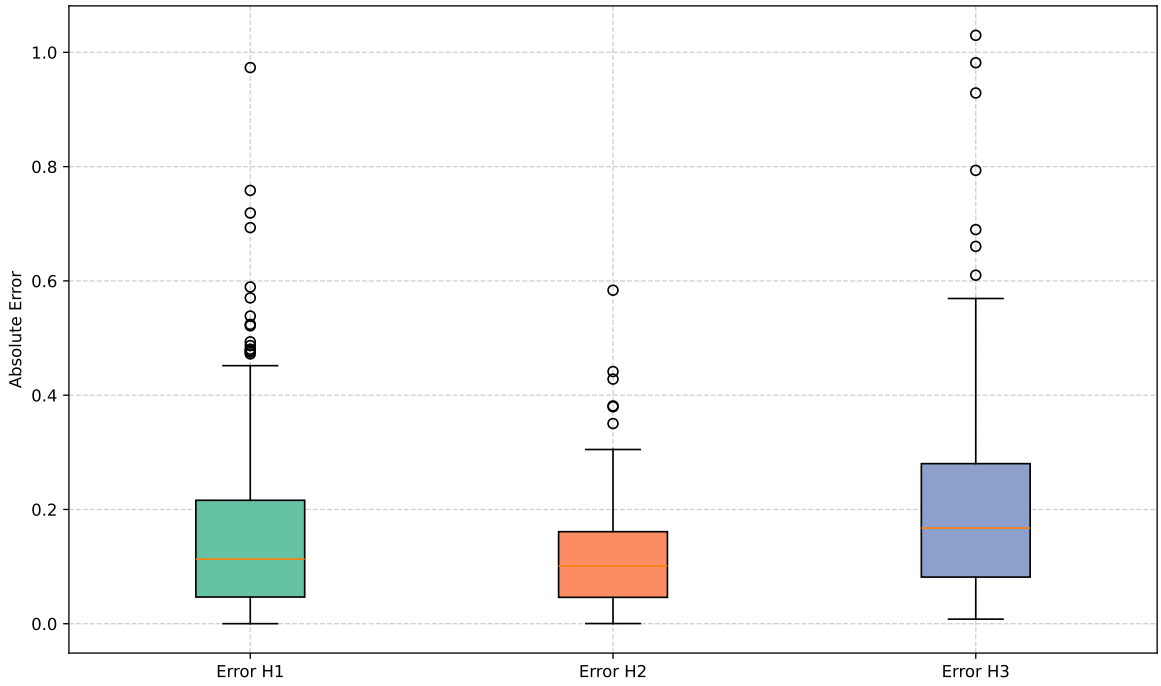

Figure 7.10: Box plot of the absolute prediction error of the MLP model for the inertia constants H_1, H_2 and H_3.

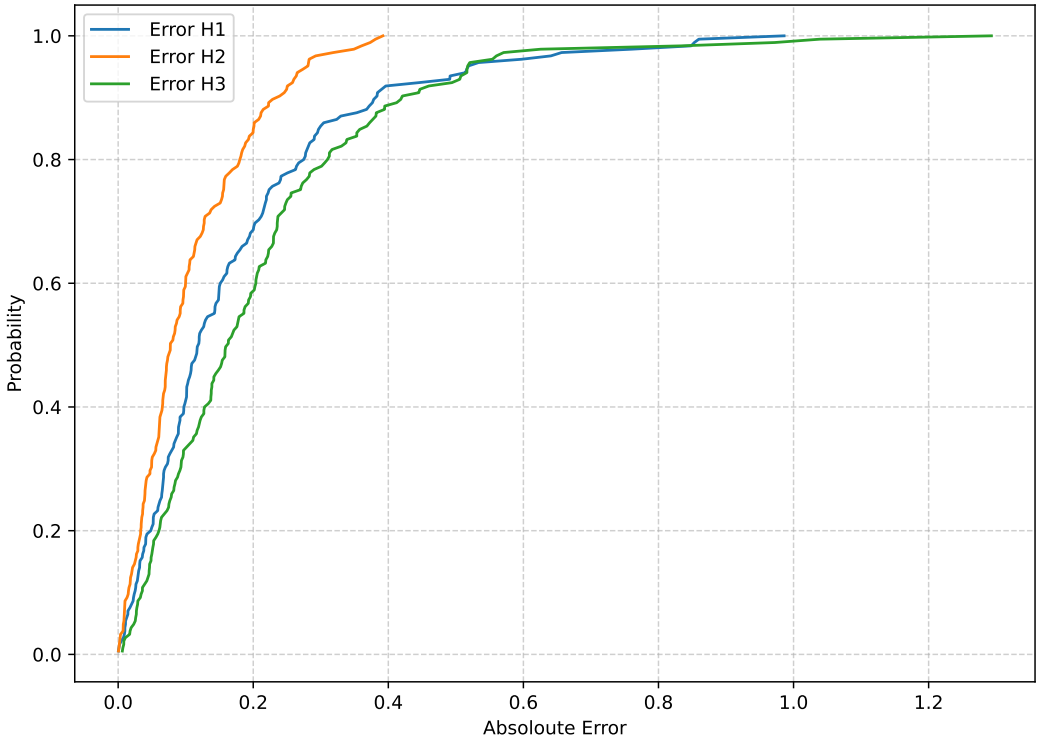

Figure 7.11: CDF of the absolute prediction error of the MLP model for the constants H_1, H_2 and H_3.

The comparison of Figures 7.9, 7.10, and 7.11 with the corresponding figures concerning the MLR algorithm (7.6, 7.7, 7.8), confirms once more the superiority of the MLP neural network compared to the multilinear regression model, regarding the estimation of the inertia constants in all three areas of the system.

7.4 Evaluation of the Rest of Algorithms

The performance of the remaining machine learning algorithms regarding the estimation accuracy of the inertia constants H_1, H_2, and H_3 lies between the two extreme cases we described earlier, without, however, constituting a reliable solution for this specific problem. The evaluation metrics for the accuracy of each algorithm in estimating each of the outputs are summarized in Tables 7.4, 7.5, and 7.6.

Table 7.4: Evaluation results of the machine learning algorithms for output H_1 (Three-Area Scenario).

Algorithm	R^2	**RMSE**	**MAE**	**MAPE (%)**
Random Forest	0.89401	0.47207	0.37476	7.31
MLR	0.72639	0.73961	0.53768	11.23
XGBoost	0.84580	0.55645	0.39973	7.65
SVR	0.89764	0.45238	0.33046	6.73
MLP	0.96842	0.25532	0.17874	3.22

Table 7.5: Evaluation results of the machine learning algorithms for output H_2 (Three-Area Scenario).

Algorithm	R^2	**RMSE**	**MAE**	**MAPE (%)**
Random Forest	0.90435	0.43897	0.33079	6.64
MLR	0.96251	0.27483	0.20851	4.10
XGBoost	0.94893	0.31987	0.22360	4.07
SVR	0.96029	0.28286	0.22208	4.44
MLP	0.98268	0.18651	0.14908	2.77

Table 7.6: Evaluation results of the machine learning algorithms for output H_3 (Three-Area Scenario).

Algorithm	R^2	**RMSE**	**MAE**	**MAPE (%)**
Random Forest	0.84969	0.56886	0.42812	8.80
MLR	0.75632	0.72430	0.54063	11.04
XGBoost	0.83401	0.59078	0.44327	8.83
SVR	0.88514	0.49727	0.38984	7.96
MLP	0.96509	0.27971	0.20856	3.91

One conclusion we can draw by carefully observing the above tables is that the inertia value of Area 2 is estimated with satisfactory accuracy, regardless of the algorithm used. Figure 7.12 interactively presents the differences in the performance of each algorithm in estimating the final output values per metric.

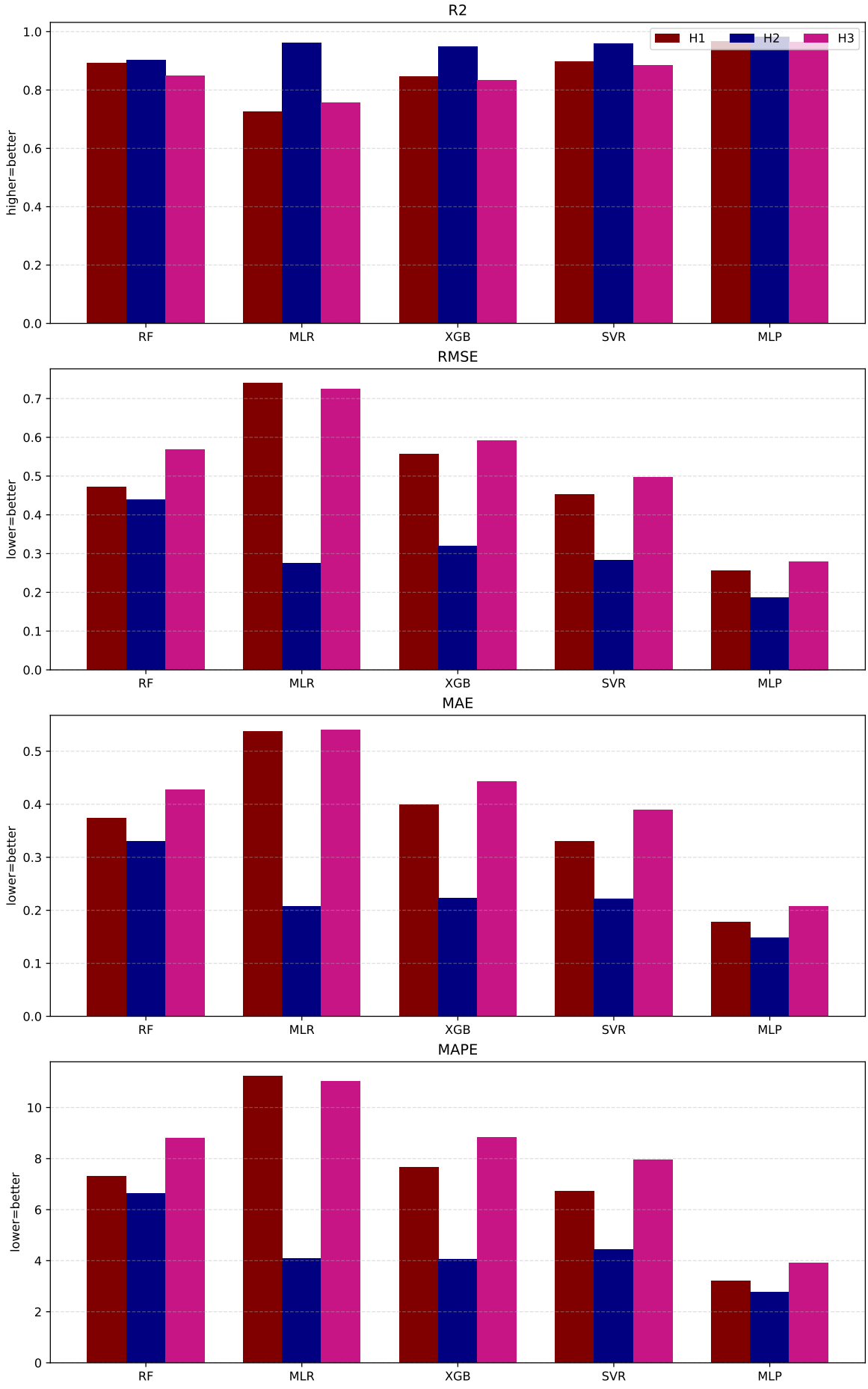

Figure 7.12: Model Performance Comparison per Metric – Three-Area Scenario.

Chapter 8

Conclusions

8.1 Summary and Conclusions

The high penetration of Renewable Energy Sources (RES) in modern power systems entails a large number of challenges that power system engineers are called upon to address with innovative solutions. One of the most significant challenges of the continuously increasing RES penetration, and the consequent replacement of conventional energy sources, concerns the reduction of the overall inertia of power systems. The total grid inertia is an indicator of the power system's resilience to abrupt disturbances in the generation/load balance. Consequently, an urgent need arises for monitoring the inertia of power systems in real-time, aiming to enable network operators to take timely corrective actions, such as the activation of standby hydroelectric plants and modern capacitors, to ensure the frequency stability of the system. In fact, knowledge of the inertia allows grid operators to know at any given time the frequency stability limits of the power system.

Within the framework of this thesis, a set of methods for estimating power system inertia levels was developed, utilizing machine learning algorithms. The proposed methods were initially evaluated on a two-area power system, and their validation was subsequently performed on a three-area power system. Training data for the machine learning algorithms were obtained by simulating power system dynamic behavior in the Simulink environment.

Initially, the validation of the proposed methods was performed on a two-area power system. Three different scenarios were examined, which differed in terms of the generation mix, i.e., the type of generating units. In each scenario, different inertia levels (H_1, H_2) are considered, and the system's dynamic responses ($\Delta f_1, \Delta f_2, \Delta P_{tie}$) are recorded. Subsequently,

a set of descriptive indicators is calculated from the dynamic responses, thereby transforming the problem under investigation into a regression problem. Then, the descriptive indicators are used as inputs for training five different machine learning algorithms: Random Forest, Multiple Linear Regression, XGBoost, SVR, and finally the MLP model, which is classified as a deep learning model (neural network).

Regarding the evaluation of the validity of the estimated inertia levels by the examined machine learning models, a common pattern was observed in all three scenarios examined in the two-region power system. In this pattern, the MLP and Random Forest algorithms produced the most accurate predictions, in that order, followed by XGBoost and SVR. The model with the largest overall error was the Multiple Linear Regression, which exhibited slightly larger errors than the other algorithms in Scenarios 1 and 2. Its total performance in Scenario 3, however, surpassed that of XGBoost and SVR. The achieved accuracy was impressive in general, with the Mean Absolute Percentage Error (MAPE) in some cases (Random Forest, XGBoost, MLR in Scenario 3, and MLP in all scenarios) remaining below 4%, a deviation considered acceptable in inertia estimation applications, as mentioned in relevant studies [32].

Subsequently, our study was extended to the estimation of the inertia constants in an interconnected power system consisting of three areas. Despite the doubling of the data volume, the problem's approach remained identical. The results demonstrate that in this case, MLP constitutes the only reliable solution, providing estimates for the inertia of all areas in the power system with a MAPE of less than 3.91%.

Although the model training was based on the use of a synthetic dataset and not real measurements, the examined models are theoretically capable of being applied under real grid conditions. Specifically, by taking a set of power system dynamic responses as input, they can estimate with high accuracy and report in real-time the inertia constants of the power system's areas.

It is worth mentioning that the examined machine learning models fall short in estimating the correlation between the system's normal operation dynamic responses and the inertia constants. Therefore, the application of a test signal is necessary, which, however, can easily be implemented with the help of a battery. Consequently, the application of the proposed machine learning models in real-world conditions requires the use of automation that would put a battery through charge and discharge cycles, in order to generate the necessary test signals.

The most significant limitation of the proposed models, and more generally of the entire approach developed within the framework of this thesis, was the assumption that the form of the frequency and power responses depends exclusively on the values of the inertia constants in each area. In reality, there are other parameters whose values influence the form of the power system's dynamic responses, such as the system damping (R), the load self-regulation constant (D), and the integral controller gains. Within the framework of this thesis, these specific parameters were considered known and constant throughout all simulations.

Lastly, it is essential to clarify that the generation mix also plays a significant role in the accuracy of the inertia constant estimation. Consequently, if the proposed models are applied under real grid conditions, they must be trained on a specific generation mix to provide reliable predictions.

8.2 Future Work

This thesis can be extended in a number of ways. The most important extensions are those that can help overcome the problems and limitations mentioned above. More specifically, it is proposed to extend the proposed models so that they take into account the unit damping (R), the load self-regulation constant (D), and the gains of the integral controllers of the Load Frequency Control (LFC) loop (K_I).

Another significant future extension concerns enhancing the generalizability of the models, so they can perform reliably under different generation unit mixes. As mentioned previously, different generation mixes result in different output dynamic responses (responses Δf_1, Δf_2, ΔP_{tie}) for the same inertia levels. To address this issue, the machine learning models can be trained with a larger training dataset. That is, a dataset that combines different inertia levels and different generation mixes. In this way, the model learns to recognize the fundamental relationships between the power system's dynamic responses and the inertia levels, regardless of the specific units generating power at any given time. Another alternative which could potentially help us overcome this challenge is the technique of transfer learning, which allows the adaptation of a pre-trained model to new generation scenarios using a smaller training dataset, without the need to retrain the models from scratch. These two approaches/extensions ensure that the developed models remain reliable and accurate, even when the grid operating conditions deviate significantly from the conditions on which they

were trained.

Lastly, to bypass the pre-processing of dynamic responses and the calculation of the statistical indicators required for developing the models examined, the use of state of the art neural networks capable of processing raw data (raw time series) directly can be leveraged. Architectures such as Recurrent Neural Networks (RNNs) and their variants LSTM/GRU, Temporal Convolutional Networks (TCNs), as well as Transformers for time series, may prove to be particularly effective techniques. In fact, the application of these techniques could enable a more accurate and flexible estimation of a power system's dynamic behavior, thereby enhancing the capability for real-time inertia level monitoring.

Taking everything into consideration, despite the challenges that exist today, the field of inertia estimation for modern power systems is a research area with significant prospects for further evolution. The search for new, fast, and efficient methods for the real-time monitoring of power system inertia levels is a necessity, in order to ensure the stability of modern power systems dominated by RES units.

Bibliography

[1] Bendong Tan, Junbo Zhao, Marcos Netto, Venkat Krishnan, Vladimir Terzija, and Yingchen Zhang. Power system inertia estimation: Review of methods and the impacts of converter-interfaced generations. *International Journal of Electrical Power Energy Systems*, 134:107362, 2022.

[2] Wesley Cole and Bethany Frew. Inertia and the power grid: A guide without the spin. Report NREL/TP-5C00-73856, National Renewable Energy Laboratory (NREL), Golden, CO, 2020. Accessed: [2025-01-09].

[3] Islam Uddin, Hamid Awan, Majdi Khalid, Salman Khan, Shahid Akbar, Mahidur Sarker, Maher Abdolrasol, and Thamer A. H. Alghamdi. A hybrid residue based sequential encoding mechanism with xgboost improved ensemble model for identifying 5-hydroxymethylcytosine modifications. *Scientific Reports*, 14, 09 2024.

[4] Prabha Kundur. *Power System Stability and Control*. McGraw-Hill, 1994.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[6] Ana Fernández-Guillamón, Eduard Muljadi, and Angel Molina-García. Frequency control studies: A review of power system, conventional and renewable generation unit modeling. *Electric Power Systems Research*, 211:108191, 2022.

[7] ENTSO-E. Winter outlook report 2023-2024. Technical report, European Network of Transmission System Operators for Electricity (ENTSO-E), October 2023.

[8] ENTSO-E. Frequency stability evaluation criteria for the synchronous zone of continental europe. Report, European Network of Transmission System Operators for Electricity, 2020.

[9] Hadi Alharbi, Ahmed Abu-Siada, Mohammad AlMuhaini, and Yateendra Mishra. Machine learning-based inertia estimation in power systems: A review of methods and challenges. *Electric Power Systems Research*, 236:109765, 2024.

[10] Mahdi Heidari, Lei Ding, Mostafa Kheshti, Weiyu Bao, Xiaowei Zhao, Marjan Popov, and Vladimir Terzija. A review on application of machine learning-based methods for power system inertia monitoring. *International Journal of Electrical Power Energy Systems*, 162:110279, 2024.

[11] Zhen Tang, Lihua Hao, Jiapeng Li, Qiuyi Chen, Yujun Li, and Zhao Xu. Power system inertia estimation based on long short term memory network. In *2021 IEEE Sustainable Power and Energy Conference (iSPEC)*, pages 2542–2547, 2021.

[12] Sayanee Das and Paramita Chattopadhyay. Data-driven inertia estimation of power systems using autoencoder-based unsupervised feature learning. In *2023 IEEE 3rd Applied Signal Processing Conference (ASPCON)*, pages 222–226, 2023.

[13] Vasupalli Sagar and Sachin Jain. System identification-based estimation of power system inertia using pmu data. pages 1–6, 12 2020.

[14] Lei Zhang, Zhihao Guo, Qianhui Tao, Zhizhi Xiong, and Jing Ye. Xgboost-based short-term prediction method for power system inertia and its interpretability. *Energy Reports*, 9:1458–1469, 2023. Selected papers from 2022 International Conference on Frontiers of Energy and Environment Engineering.

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, USA, 2006.

[16] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[17] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Sebastopol, CA, 2nd edition, 2019.

[18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2 edition, 2009.

[19] Rob J Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2006.

[20] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014.

[21] Alysha M De Livera, Rob J Hyndman, and Ralph D Snyder. Forecasting models and prediction intervals for the multiplicative holt-winters method. *International Journal of Forecasting*, 27(4):956–979, 2011.

[22] Nico JD Nagelkerke. A note on a general definition of the coefficient of determination. *Biometrika*, 78(3):691–692, 1991.

[23] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[24] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[25] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[26] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[27] P.M. Anderson and M. Mirheydar. A low-order system frequency response model. *IEEE Transactions on Power Systems*, 5(3):720–729, 1990.

[28] Banaja Mohanty, Sidhartha Panda, and P.K. Hota. Controller parameters tuning of differential evolution algorithm and its application to load frequency control of multi-source power system. *International Journal of Electrical Power Energy Systems*, 54:77–85, 2014.

[29] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[30] Sheldon M. Ross. *A First Course in Probability*. Pearson, 9 edition, 2014.

[31] Hêmin Golpîra and H. Bevrani. A framework for economic load frequency control design using modified multi-objective genetic algorithm. *Electric Power Components and Systems*, 42, 06 2014.

[32] D. Linaro et al. Continuous estimation of power system inertia using convolutional neural networks. *Nature Communications*, 14(1):1–10, 2023.

[33] M. Goossens, F. Mittelbach, and A. Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 9 edition, 1993.

[34] Ι. Κάβουρας. *Συστήματα Υπολογιστών*. Κλειδάριθμος, Αθήνα, 3 edition, 1991.

[35] J. Liaperdos, A. Arapoyanni, and Y. Tsiatouhas. Adjustable RF mixers' alternate test efficiency optimization by the reduction of test observables. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(9):1383–1394, Sept. 2013.

[36] I. Liaperdos, L. Dermentzoglou, A. Arapoyanni, and Y. Tsiatouhas. Fault detection in RF mixers combining defect-oriented and alternate test strategies. In *26th Conference on Design of Circuits and Integrated Systems (DCIS)*, San Sebastian, Spain, Nov. 2011.

[37] Latex project. `http://www.latex-project.org`. Ημερομηνία πρόσβασης: 13-11-2014.

[38] Ε. Ανδρουλάκη. Υλοποίηση Ενεργού Μηχανισμού σε Σύστημα Ομότιμων Βάσεων. Πτυχιακή εργασία, KDBS Lab, Εθνικό Μετσόβιο Πολυτεχνείο, Ιουλ. 2005.

[39] Ζ. Καούδη. Πρότυπο Σύστημα Αποθήκευσης και Διαχείρισης Σχημάτων rdfs. Διπλωματική εργασία, Εθνικό Μετσόβιο Πολυτεχνείο, Ιουλ. 2004.

[40] Ζ. Λάσκαρη. Κοινωνική Ανάλυση των Ταινιών της finos films. Master's thesis, Εθνικό Μετσόβιο Πολυτεχνείο, Aug. 2012.

[41] Ζ. Κουρούκλη. *Κατανεμημένα Συστήματα.* PhD thesis, ΤΕΙ Πελοποννήσου, Dec. 2013.

[42] H. Cheng J. Gao and P.-N. Tan. A framework for incorporating labeled examples into anomaly detection. Technical Report MSU-CSE-05-29, Department of Computer Science, Michigan State University, East Lansing, Michigan, 2005.

[43] P. Viswanathan, G. Winner, and P. Vyas. Convenient provisioning of embedded devices with wifi capability. US Patent 8,665,744, 2014.

[44] K. Patroumpas and T. Sellis. Subsuming multiple sliding windows for shared stream computation. In Johann Eder, Maria Bielikova, and A Min Tjoa, editors, *Advances in Databases and Information Systems*, volume 6909 of *Lecture Notes in Computer Science*, pages 56–69. Springer, 2011. doi:10.1007/978-3-642-23737-9_5.

APPENDICES

Appendix A

Τίτλος Παραρτήματος

Τα παραρτήματα περιλαμβάνουν συνοδευτικό, υποστηρικτικό υλικό (πίνακες, φωτογραφίες, ερωτηματολόγια, στατιστικά στοιχεία, αποδείξεις, περιγραφές λογισμικών προγραμμάτων, παραδείγματα, περιγραφές πολύπλοκων διαδικασιών, λίστα με πρωτογενή στοιχεία, λεπτομερής περιγραφή και προδιαγραφές εξοπλισμού, οδηγίες εγκατάστασης λογισμικού, κ.λπ.), ή αλλιώς ό,τι θεωρείται χρήσιμο να περιγραφεί, αλλά δεν συνηθίζεται να εντάσσεται μέσα στο κυρίως κείμενο της Εργασίας. Στο κυρίως κείμενο της Εργασίας πρέπει να γίνονται οι κατάλληλες παραπομπές προς τα παραρτήματα, όπου το κείμενο σχετίζεται με υλικό που περιλαμβάνεται σε αυτά. Ένα παράρτημα, αναλόγως με το περιεχόμενό του, μπορεί να είναι ενιαίο, ή να χωρίζεται σε ενότητες.

A.1 Δυνατότητες του LaTeX

Καθώς το παρόν αποτελεί ένα πρότυπο συγγραφής διπλωματικών εργασιών, στην ενότητα αυτή επιδεικνύονται ορισμένες από τις δυνατότητες το LaTeXοι οποίες μπορούν να αξιοποιηθούν στο κείμενο μιας διπλωματικής εργασίας (μια καλή πηγή για τη διερεύνηση των δυνατοτήτων του LaTeXείναι η ιστοσελίδα `https://www.overleaf.com/learn/latex/Main_Page`).

A.1.1 Πίνακες

Ο Πίνακας A.1 είναι ένα παράδειγμα πίνακα σχεδιασμένου με εντολές του περιβάλλοντος `tabular`.

Table A.1: Παράμετροι πειραμάτων

Πλήθος κελιών καννάβου $c \times c$	50×50, 100×100, 200×200, $\mathbf{250} \times \mathbf{250}$, 500×500, 1000×1000
Τυπική απόκλιση σ	25m, 50m, 75m, **100m**, 150m, 200m
Αριθμός εγγύτερων γειτόνων k	1, 2, **3**, 4, 5, 10, 20
Πιθανοτικό κατώφλι θ	50%, 60%, 70%, **75**%, 80%, 90%, 99%

A.1.2 Διαγράμματα - Γραφικές παραστάσεις

Στο Σχήμα A.1, παρουσιάζεται ένα παράδειγμα γραφικής παράστασης σχεδιασμένης με το Gnuplot. Ένας άλλος τρόπος κατασκευής τέτοιων παραστάσεων/διαγραμμάτων είναι με χρήση του πακέτου pgfplots (`http://pgfplots.sourceforge.net/`).

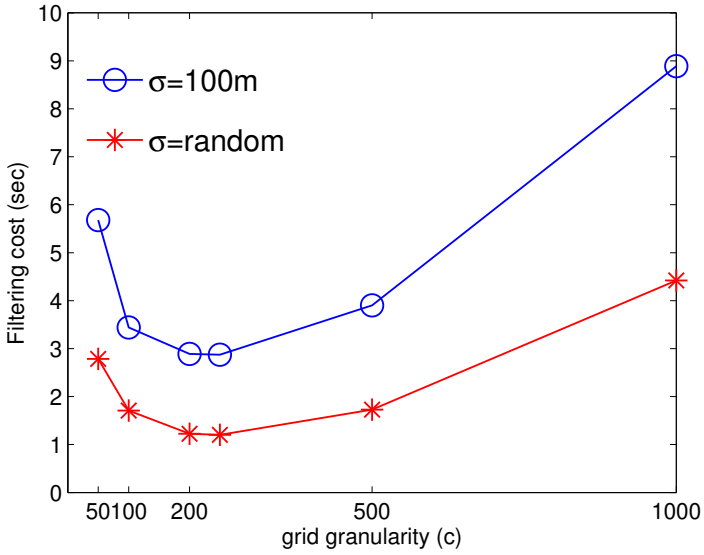

Figure A.1: Κλιμάκωση χρόνου εκτέλεσης για διάφορες υποδιαιρέσεις του καννάβου

Στο Σχήμα A.2, παρουσιάζεται ένα παράδειγμα εισαγωγής σχήματος/εικόνας που περιέχεται σε αρχείο pdf, ενώ στο Κεφάλαιο 2, στο Σχήμα **??** παρουσιάζεται ένα παράδειγμα εισαγωγής σχήματος/εικόνας που περιέχεται σε αρχείο jpg.

A.1.3 Σχήματα

Ακολουθεί στο Σχήμα A.3 ένα παράδειγμα σχήματος φτιαγμένου με εντολές του πακέτου Ti*k*Z.

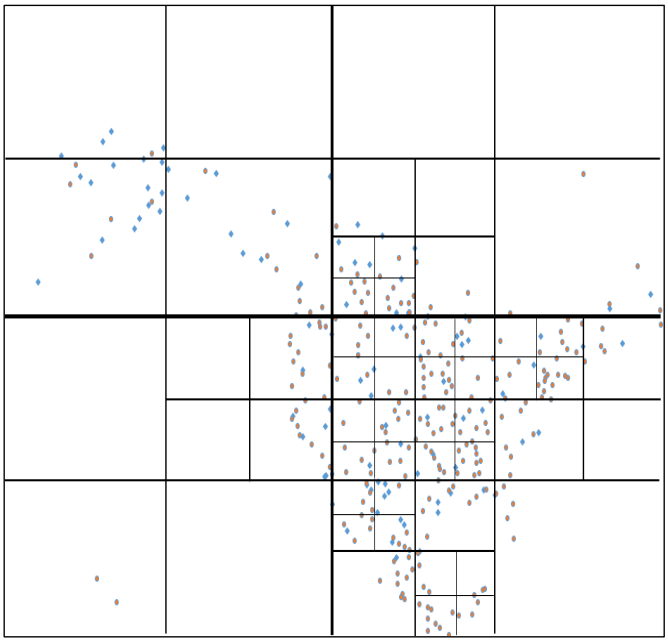

Figure A.2: Quadtree partitioning.

A.1.4 Μαθηματικές εκφράσεις

Ακολουθούν παραδείγματα μαθηματικών εκφράσεων.

$$\hat{I}(x,u,t) \quad = \quad dist(y(t_f),\Gamma) + \int_t^{t_f} \mathcal{L}(y(s),u(s),s)\,ds \tag{A.1}$$

$$\frac{d}{dx}\left(\int_0^z f(u)\,du\right) = f(x).$$

A.1.5 Αλγόριθμοι

Ακολουθεί ο Αλγόριθμος 1, ο οποίος είναι μορφοποιημένος με τα πακέτα algorithm και algorithmic.

Algorithm 1 Probabilistic $k\theta NN$ Monitoring

1: **Procedure** *VerifyCandidate* (focal query point q, threshold θ, object o, list of auxiliary objects P, distance $kMAXDIST$)
2: **if** $\Phi(o, kMAXDIST) \geq \theta$ **and** $L_2(q,o) \leq L_2(q, P.\text{top}())$ **then**
3: P.pop(); *//Replace the most extreme element in P, since candidate o ...*
4: P.push(o); *//... has enough probability and has its mean closer to focal q*
5: **end if**
6: **End Procedure**

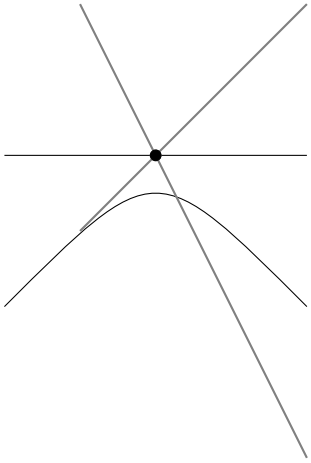

Figure A.3: Παράδειγμα σχήματος με εντολές του πακέτου Ti*k*Z

A.1.6 Θεωρήματα, Πορίσματα, Ορισμοί, κλπ.

Ακολουθεί παράδειγμα θεωρήματος από την ιστοσελίδα `https://www.overleaf.com/learn/latex/Theorems_and_proofs`

Theorem 9.1. *Let f be a function whose derivative exists in every point, then f is a continuous function.*

A.1.7 Απαριθμήσεις

Μια απαρίθμηση (itemized list) βοηθά στην παρουσίαση μιας σειράς περιπτώσεων με σαφήνεια. Ακολουθεί παράδειγμα.

Η εκπαίδευση στην Ελλάδα διακρίνεται σε:

- Πρωτοβάθμια
- Δευτεροβάθμια
- Τριτοβάθμια

A.1.8 Είδη πηγών στις αναφορές

Στο references.bib μπορεί να δει κανείς πώς γράφονται διάφορα είδη πηγών (Βιβλία Ξενόγλωσσα [33], Βιβλία Ελληνικά [34], Άρθρα σε επιστημονικά περιοδικά [35], Άρθρα σε επιστημονικά συνέδρια [36], Ιστοσελίδες [37], Πτυχιακές Εργασίες [38], Διπλωματικές Εργασίες [39], Μεταπτυχιακές Διπλωματικές Εργασίες [40], Διδακτορικές Διατριβές [41], Τε-

χνικές Αναφορές [42], Διπλώματα Ευρεσιτεχνίας [43]), Κεφάλαια σε συλλογικούς τόμους [44].

Appendix B

Τίτλος 2ου Παραρτήματος

Τα παραρτήματα περιλαμβάνουν συνοδευτικό, υποστηρικτικό υλικό (πίνακες, φωτογραφίες, ερωτηματολόγια, στατιστικά στοιχεία, αποδείξεις, περιγραφές λογισμικών προγραμμάτων, παραδείγματα, περιγραφές πολύπλοκων διαδικασιών, λίστα με πρωτογενή στοιχεία, λεπτομερής περιγραφή και προδιαγραφές εξοπλισμού, οδηγίες εγκατάστασης λογισμικού, κ.λπ.), ή αλλιώς ό,τι θεωρείται χρήσιμο να περιγραφεί, αλλά δεν συνηθίζεται να εντάσσεται μέσα στο κυρίως κείμενο της Εργασίας. Στο κυρίως κείμενο της Εργασίας πρέπει να γίνονται οι κατάλληλες παραπομπές προς τα παραρτήματα, όπου το κείμενο σχετίζεται με υλικό που περιλαμβάνεται σε αυτά. Ένα παράρτημα, αναλόγως με το περιεχόμενό του, μπορεί να είναι ενιαίο, ή να χωρίζεται σε ενότητες.