POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

EXECUTIVE SUMMARY OF THE THESIS

Physics-based Neural Network modelling, Predictive Control and Lifelong Learning applied to District Heating Systems

LAUREA MAGISTRALE IN AUTOMATION AND CONTROL ENGINEERING - INGEGNERIA DELL'AUTOMAZIONE

Author: LAURA BOCA DE GIULI
Advisor: PROF. RICCARDO SCATTOLINI
Co-advisor: PROF. ALESSIO LA BELLA
Academic year: 2021-2022

---

1. Introduction

The growing issue of climate change calls for ever-increasing advanced technological solutions capable of dealing with it. In particular, District Heating Systems (DHSs) are pointed out as important tools to reach energy transition targets. A DHS is in fact an energy system used to deliver the heat generated in a centralized station through insulated water pipelines, commonly named District Heating Network or DHN, both for residential and for commercial purposes. In the state-of-the-art DHSs literature, the vast majority of control strategies proposed are based on the physical model equations. This thesis, instead, aims at identifying a DHN model through a machine learning technique, namely Recurrent Neural Networks, which are particularly suitable to identify the non-linearities of such a complex thermo-hydraulic system. However, in order to exploit both data-driven methods and physical knowledge, a Physics-based Recurrent Neural Network (PB-RNN) implementation is proposed. In particular, the main challenge of the thesis consists in developing a neural network that is capable of representing the physical interactions between the main elements of the system under investigation. This objective is achieved by making the PB-RNN resemble the physical system modular structure. After the identification of a performing model, the latter is exploited in a Model Predictive Control (MPC) strategy to optimize the district heating system operation by minimizing its electrical cost. Finally, a lifelong learning algorithm is presented in order to monitor the DHS in the long run. In detail, the algorithm is intended to continually supervise the plant so as to detect and solve potential anomalies. The overall target is to refine the existing system model by acquiring continuous information, while preventing the new knowledge from significantly interfering with past data.

2. District Heating Systems

A district heating system consists of four main parts: a *forward-flow part*, which provides hot water to consumers (the "supply" network), *consumers*, that use hot water for heating, a *backward-flow part*, which transports the cooled water back to the heating station (the "return" network) and the *heating station*, where the warming of the cooled water takes place [3].

In particular, the fundamental components of a district heating system are here briefly described, whereas the mass, momentum and energy balance equations that govern the various elements are thoroughly deepened in [3]. First, the gas boiler is a heater which provides heat to the network flowing water, i.e. it sets the desired supply temperature. Another crucial element is the pressure pump, used to impose a constant differential pressure between its inlet and outlet port without losses and at any mass flow rate. Moreover, an expansion vessel is modelled to impose a certain pressure reference in the backward-flow part of the DHS. Clearly, water is delivered to users through a pipeline network (DHN): pipes are the elements that introduce transport delay effects on temperature profiles. Lastly, a DHS is made of several consumers or thermal loads: actually, each user is an aggregation of loads that could be houses, industrial facilities or commercial buildings.

In the thesis, all the experiments are carried out on a specific system: a referenced DHN named AROMA network, whose physical governing equations are described in [3]. A schematic representation is reported in Figure 1, where the five users composing the system are highlighted in green.

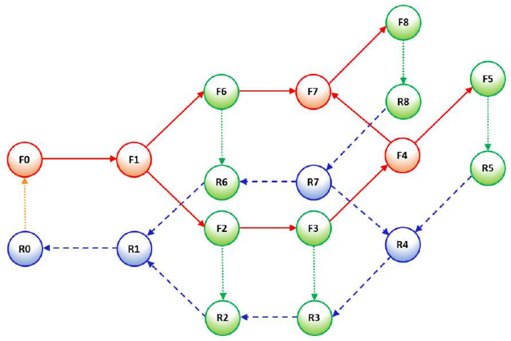


Figure 1: AROMA network schematic representation: forward-flow arcs are plotted in solid red, backward-flow arcs in dashed blue, loads in green.

The aforementioned DHS is developed and simulated through the Modelica environment, subsequently paired with MATLAB and Simulink for control and identification purposes. Finally, the main network variables that are going to be studied and identified in the remaining of the work are the load supply and return temperatures, together with the mass flow rates, plus the overall return temperature and mass flow rate.

3. Model Identification

For complex large-scale processes with countless phenomena occurring simultaneously, an accurate physics-based dynamic model may be impossible to develop. Consequently, the first goal of the thesis is to describe the AROMA DHN through data-driven techniques. The latter deal with the problem of building mathematical models of dynamical systems based on observed data from the plant itself. It is worth recalling that the system under analysis is fed with inputs (plus external disturbances) and it produces outputs, that are exactly what identification aims to get. Therefore, the procedure to follow is quite straightforward: input and output signals from the system are collected and processed by a data analysis technique so as to infer a model [4]. In order to quantitatively evaluate the identification results it is useful to define an assessment rule, such as the FIT index, i.e. $FIT = \left(1 - \frac{||\mathbf{y}_{real}-\mathbf{y}_{id}||_2}{||\mathbf{y}_{real}-y_{avg}||_2}\right) \cdot 100$ and the coefficient of determination, i.e. $R^2 = \left(1 - \frac{\sum_{i=1}^{T}(y_{i,real}-y_{i,id})^2}{\sum_{i=1}^{T}(y_{i,real}-y_{avg})^2}\right) \cdot 100$, where $\mathbf{y}_{id}$ is the vector of identified outputs, $\mathbf{y}_{real}$ the vector of the real ones and y_{avg} is its average.

After collecting a huge number of samples through a simulation in which many pseudorandom binary signals are fed to the system, various identification techniques can be exploited. In particular, classical linear models such as State-space, Autoregressive with external input and Output-error models are tested in order to identify the main AROMA network variables. However, these methods turn out to have poor predictive accuracy (low FIT values). Indeed, since the system under control is characterized by a non-linear behaviour, it is evident that standard linear model structures are not appropriate. For this reason, it is convenient to exploit other black-box identification techniques, such as Recurrent Neural Networks (RNNs), particularly suitable to process time series data.

In general, RNNs are stateful neural networks that can be described as a dynamical state-space model:

$$\begin{cases} x_{k+1} = f(x_k, u_k; \Phi) \\ y_k = g(x_k, u_k; \Phi) \end{cases} \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector, $u \in \mathbb{R}^{n_u}$ is the input, $y \in \mathbb{R}^{n_y}$ is the output and Φ is the set of parameters (*weights* and *biases*) that are computed during the training procedure [1]. The most advanced RNN architectures include the category of gated RNN, in which the internal loops are regulated by the so-called gates, that make them more suitable to learn dynamical systems [1]. To be precise, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) belong to this neural networks group and are here employed.

In detail, the RNNs training is implemented in Python, exploiting the so-called Truncated Back-Propagation Through Time (TBPTT) and testing a number of hyperparameters (hidden layers, neurons and optimizers). By comparing the performance indexes (R^2_{min}, R^2_{max}, FIT, best epoch and training time) of the different combinations of neural networks and hyperparameters, it is possible to draw some conclusions. Indeed, a GRU network made of two hidden layers with 14 neurons each and exploiting the ADAM optimizer is the most suitable to predict the main AROMA network variables behaviour. From the study of the minimum and maximum performance indexes value, it turns out that the variables associated to the closest users with respect to the heating station are the simplest to identify (high R^2), whereas the variables related to the furthest consumers are the most troublesome (low R^2) because of the major pressure losses occurring in pipes. However, standard RNNs do not allow to solve this type of local problems, because of their abstract structure. Therefore, it is intuitive to think that a neural network architecture inspired by the physical topology of the system may be helpful.

4. Physics-based Recurrent Neural Networks

When developing predictive models for control purposes, one has to balance model accuracy, complexity and robustness. In general, machine learning-only or physics-only approaches may not be sufficient for very complex and wide systems such as DHNs [2]. For this reason, the key challenge of the thesis consists in developing a Physics-based Machine Learning (PB-ML) algorithm able to improve standard RNNs performance. Because of the huge applicability range of the approach, PB-ML is having great success in the scientific community. However, the vast majority of the methods are either strongly problem-dependent or they simply include an upgraded loss function which is informed by the physical equations of the system. Unfortunately, in the AROMA network case, the RNN implementation where the loss function is guided by a physical constraint (the supply temperature must be always greater than the return one) does not bring significant improvements over standard RNNs performance. Consequently, instead of simply altering the loss function, it makes more sense to drastically modify the RNN architecture, so as to make it resemble the DHN topology. In fact, the novel idea proposed by this work is to model each load (or loads cluster) as a RNN with a single hidden layer, so that its predicted output variables are given as inputs to the subsequent consumers (again each one modelled as a RNN) that are directly affected by the former (see Figure 2).

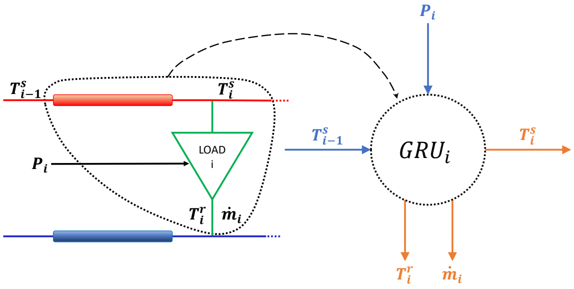


Figure 2: Representation of how the i^{th} load (green) and variables of the physical system are turned into a RNN model. Its inputs are depicted in light-blue, the outputs in orange.

Specifically, each forward pipe connected to the i^{th} user and the consumer itself are hard-coded as a GRU network whose input is, besides power, the supply temperature(s) of the preceding load(s). In detail, from several experiments it turns out that each GRU in a *meshed* network must be fed with the supply temperatures of the just preceding loads, regardless of the mass flow rate direction. In this way, each load is informed about what happens, in terms of temperature, just before its location and not only in the heating station, as in the original RNN. Moreover, as far as power is concerned, it is convenient to feed the i^{th} load (GRU) with its associated power consumption plus the summation of the other load profiles, in order to give each GRU the information regarding the overall situation of the heating system. Finally, the return network is implemented as a single-layer GRU receiving as input each load return temperature and mass flow rate. Overall, the so-implemented PB-RNN is made by six GRU networks, each one with its specific inputs (see Figure 3).

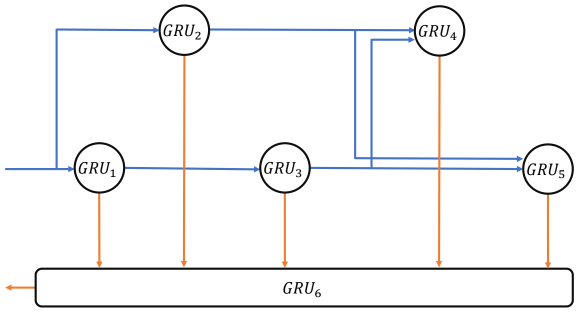


Figure 3: Physics-based RNN scheme.

The Python implementation is assisted by the graph theory: once the incidence matrix describing the physical system interconnections is defined, the training procedure can start. Actually, another important parameter, which defines how many neurons are assigned to each GRU (or GRU layer), must be chosen. In fact, thanks to the physical correspondence between a user and the associated RNN, it is possible to allocate a greater number of neurons where the identification performance is normally poor (most distant users). Ultimately, it is reasonable to claim that the further the consumer, the higher the number of neurons assigned to the corresponding GRU network.

By comparing the FIT trend of a PB-RNN made up of six GRUs and the FIT trend of a six-layer standard RNN (having the same average amount of neurons per layer), it is possible to graphically understand the strength of the physics-informed method (see Figure 4). The PB-GRU, in fact, reaches a higher FIT value in a smaller amount of time. In particular, the standard GRU with [15,15,15,15,15,15] neurons attains the maximum average FIT of 72.6% after 1495 epochs, in an overall training time of two hours and twenty-nine minutes. In addition, the associated R^2 indexes are $R^2_{min} = 69.7\%$ and $R^2_{max} = 97.5\%$. By contrast, the PB-GRU with [9,9,9,16,16,30] neurons reaches the maximum average FIT of 83.3% after 355 epochs, in an overall training time of one hour and twelve minutes. Additionally, its associated R^2 indexes are $R^2_{min} = 89.4\%$ and $R^2_{max} = 98.5\%$.

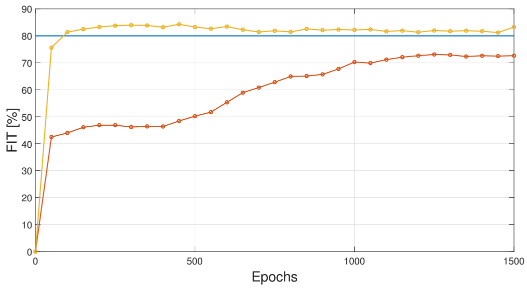


Figure 4: Comparison, over 1500 epochs, between the FIT trend of a traditional RNN (red) and of a physics-based one (yellow). The target FIT is represented in blue.

From this test and values it should be clear how the PB-GRU outperforms the standard RNN, even though the two networks are characterized by the same hyperparameters (number of layers, neurons and optimizer). Indeed, thanks to a network structure that resembles the physical system topology it is possible to place a greater number of neurons only where strictly necessary. To sum up, the PB-GRU achieves greater FIT and R^2 values in a smaller amount of training time, and, thanks to its physical interpretability, it is also easier to detect and thus solve identification issues.

5. Model Predictive Control

Once a performing system model is identified, it is possible to develop a control strategy. In particular, it is convenient to exploit the popular Non-linear Model Predictive Control algorithm because of its well-known astonishing performance. The basic concept is to transform the control synthesis problem into an optimization one, so as to achieve a time invariant control law.

In this framework, the control variable that must be manipulated to reach the optimization targets is the boiler temperature. In particular, the main objective of the finite horizon control optimization problem is to minimize the electrical cost of the boiler (time varying parameter), while fulfilling power, mass flow rate and temperature constraints. Indeed, both because of computational reasons and physical limitations, state, input and output constraints are inserted in the problem formulation. Additionally, in order to reduce the computational effort, some strategies can be adopted. First, a sampling time of five minutes (both for the MPC and for the PB-GRU model) is selected so that a prediction horizon of six hours can be chosen. Besides, along with a warm start initialization, a blocking strategy is implemented: in this way, the control variable is fixed for thirty minutes and the optimization procedure carried out by CasaADi paired with Ipopt is not overloaded.

Finally, the use of the GRU network for model predictive control purposes calls for the availability of a state estimate. To this end, an open-loop observer replicating the system dynamics by means of the PB-GRU non-linear equations is designed.

The MPC scheme is reported in Figure 5.

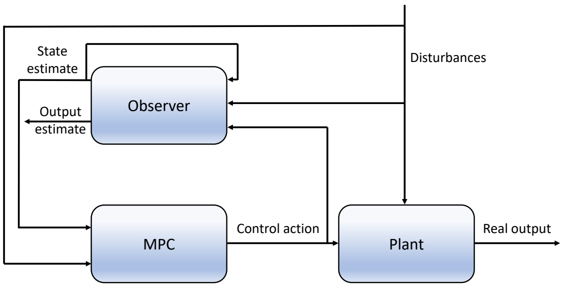


Figure 5: MPC scheme.

By carrying out a daily optimization procedure both with a 54-state PB-RNN and with a 54-state standard RNN, it is possible to observe that the most accurate model (PB-GRU) yields the best optimization and control results: a faster execution, less constraints violation and cost savings with respect to the standard GRU model.

6. Lifelong Learning

In the long run, it is very likely that throughout the lifespan of a system various changes occur, such as structural adjustments or unusual operating conditions. As a consequence, the system model initially employed does no longer constitute a precise description of the plant. Thereby, it comes the need to regularly monitor and hence adapt the original model to changes, while still preserving previously gained knowledge.

A lifelong learning system is indeed defined as an adaptive algorithm capable of learning from a continuous stream of information, with such knowledge becoming progressively available over time and where the number of tasks to be learned is not predefined. Critically, the accommodation of new information should occur without catastrophic forgetting or interference [5].

In order to deal with the problem of model change, a novel methodological algorithm is proposed. In practice, when the $RMSE$ between the measured output variables and the ones predicted by the existing PB-GRU model overshoots a certain threshold, then something in the system has changed. Therefore, from the computation of the Mahalanobis distance (T^2) of input and output variables it is possible to distinguish whether a change in the plant or in the operating conditions occurred. In fact, if the statistical distance between the current inputs and the training ones exceeds its threshold, then the actual operating conditions are statistically far from the original ones. In the opposite case, there has been a change in the plant structure as the input values are within the training set boundaries. The just explained strategy is synthesized in Algorithm 1.

Algorithm 1 Lifelong Learning Algorithm

```
Compute $RMSE_i \quad \forall i \in \{1,..,n_y\}$
if $RMSE_i \le \overline{RMSE_i} \quad \forall i \in \{1,..,n_y\}$ then
    do nothing
else if $\exists i \in \{1,..,n_y\} | RMSE_i > \overline{RMSE_i}$ then
    compute Mahalanobis distances
    if $(T^2(u) \le T^2_\alpha(u)) \wedge (T^2(y) > T^2_\alpha(y))$ then
        change in the plant: Moving Horizon Estimation
    else if $(T^2(u) > T^2_\alpha(u)) \wedge (T^2(y) > T^2_\alpha(y))$ then
        change in the operating conditions: additive uncertainty identification
    end if
end if
```

6.1. Plant Change Scenario

If a change in the plant structure or parameters is detected, then a new model must be found: the old one is no longer able to correctly capture the plant dynamics. To this end, we propose a

Moving Horizon Estimation algorithm that optimizes the neural network model parameters. Actually, differently from the traditional strategy, only the output layers weights and biases are optimized so as to significantly reduce the computational effort and to avoid catastrophic forgetting. In particular, a test is carried out on a drastically modified plant. The validation results are pretty explicative: the new model characterized by the re-optimized output parameters reaches an average R^2 value of 91.6%, outperforming the old PB-GRU model which scores $R^2_{avg} = -117.4\%$.

6.2. Operating Conditions Change Scenario

When an operating conditions shift is detected, it is not necessary to re-build a model. Indeed, the existing PB-GRU is still able to predict the system dynamics in the vast majority of working conditions. For this reason, it is convenient to simply add an incremental (or additive) neural network, once again physics-based, whose aim is to estimate the dynamics of the prediction error committed by the old PB-GRU. To this end, the additive PB-GRU must be trained from scratch with a reasonable amount of normal and abnormal operating conditions. In addition, for control purposes, the incremental network can be characterized by a smaller number of neurons with respect to the original one, being the error dynamics pretty limited. Finally, a validation test is carried out on the original plant fed with anomalous working conditions (out of the initial training range). In particular, the new overall neural network, made by the summation of the original and the incremental PB-GRU, reaches an average R^2 value of 91.9%. By contrast, the original PB-GRU model alone attains the value of $R^2_{avg} = 78.3\%$. These results confirm that when abnormal power requests occur, the initial PB-GRU must be helped by an incremental network.

7. Conclusions

To sum up, the main work challenges have been successfully tackled. The first accomplishment is the modelling of a DHN through data-driven methods, and in particular via recurrent neural networks. However, the crucial contribution of the thesis consists in the implementation of a physics-based RNN. In fact, thanks to a structure of the neural network that resembles the physical system topology and interactions, many enhancements with respect to standard RNNs are attained. In addition, a further achievement of the work is related to the NMPC strategy applied to a DHN whose physics-based GRU equations are exploited. The last contribution regards the lifelong learning issue: a novel methodological algorithm that deals with the problem of model changes over time has been proposed. Finally, the thesis is a starting point for many possible developments. Above all, it would be desirable to try out the physics-based machine learning method proposed on other and different types of system. In this way, a methodological generalization of the approach could be eventually synthesized. Furthermore, it could be convenient to implement a closed-loop observer which exploits the PB-RNN equations in the MPC problem formulation. Lastly, the lifelong learning topic is still an outstanding issue and further investigation on the two strategies proposed in the thesis is advisable.

References

[1] Fabio Bonassi, Marcello Farina, Jing Xie, and Riccardo Scattolini. On recurrent neural networks for learning-based control: recent results and ideas for future developments. *Journal of Process Control*, 114:92–104, 2022.

[2] Anuj Karpatne, Ramakrishnan Kannan, and Vipin Kumar. *Knowledge Guided Machine Learning: Accelerating Discovery using Scientific Knowledge and Data.* CRC Press, 2022.

[3] Richard Krug, Volker Mehrmann, and Martin Schmidt. Nonlinear optimization of district heating networks. *Optimization and Engineering*, 22(2):783–819, 2021.

[4] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

[5] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

POLITECNICO
MILANO 1863

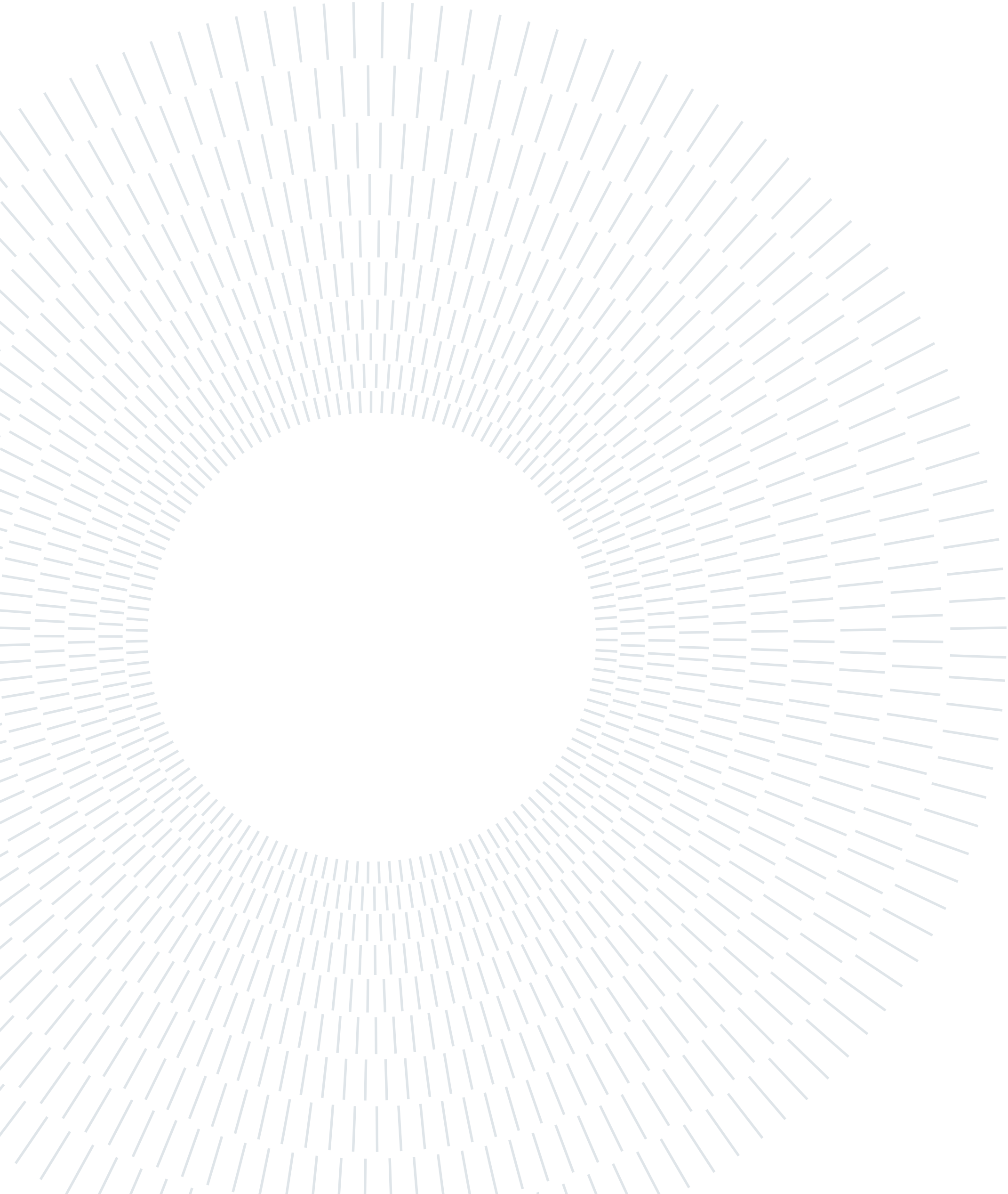
SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Physics-based Neural Network modelling, Predictive Control and Lifelong Learning applied to District Heating Systems

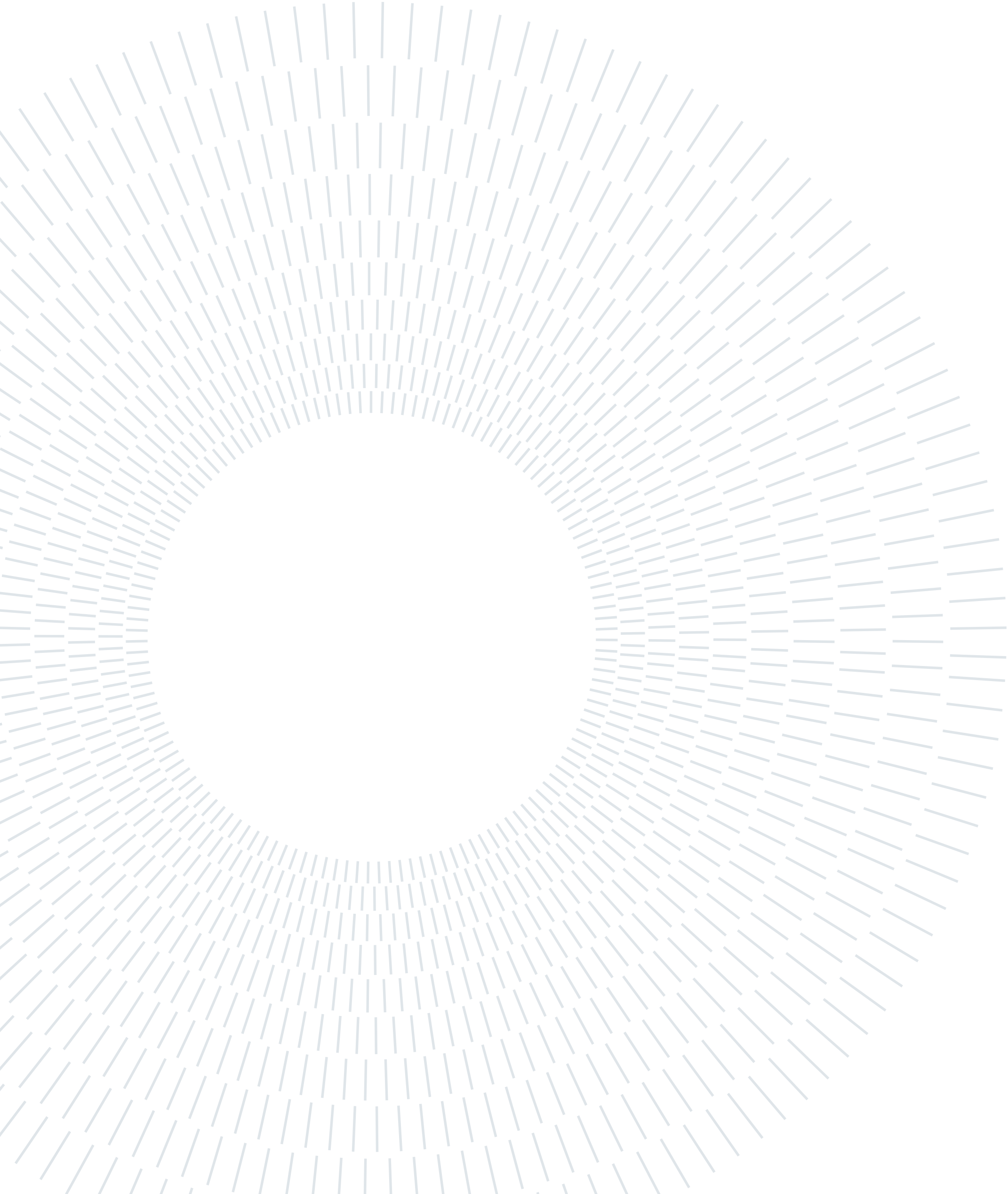
Tesi di Laurea Magistrale in
Automation and Control Engineering -
Ingegneria dell'Automazione

Author: **Laura Boca de Giuli**

Student ID: 970053
Advisor: Prof. Riccardo Scattolini
Co-advisor: Prof. Alessio La Bella
Academic Year: 2021-22



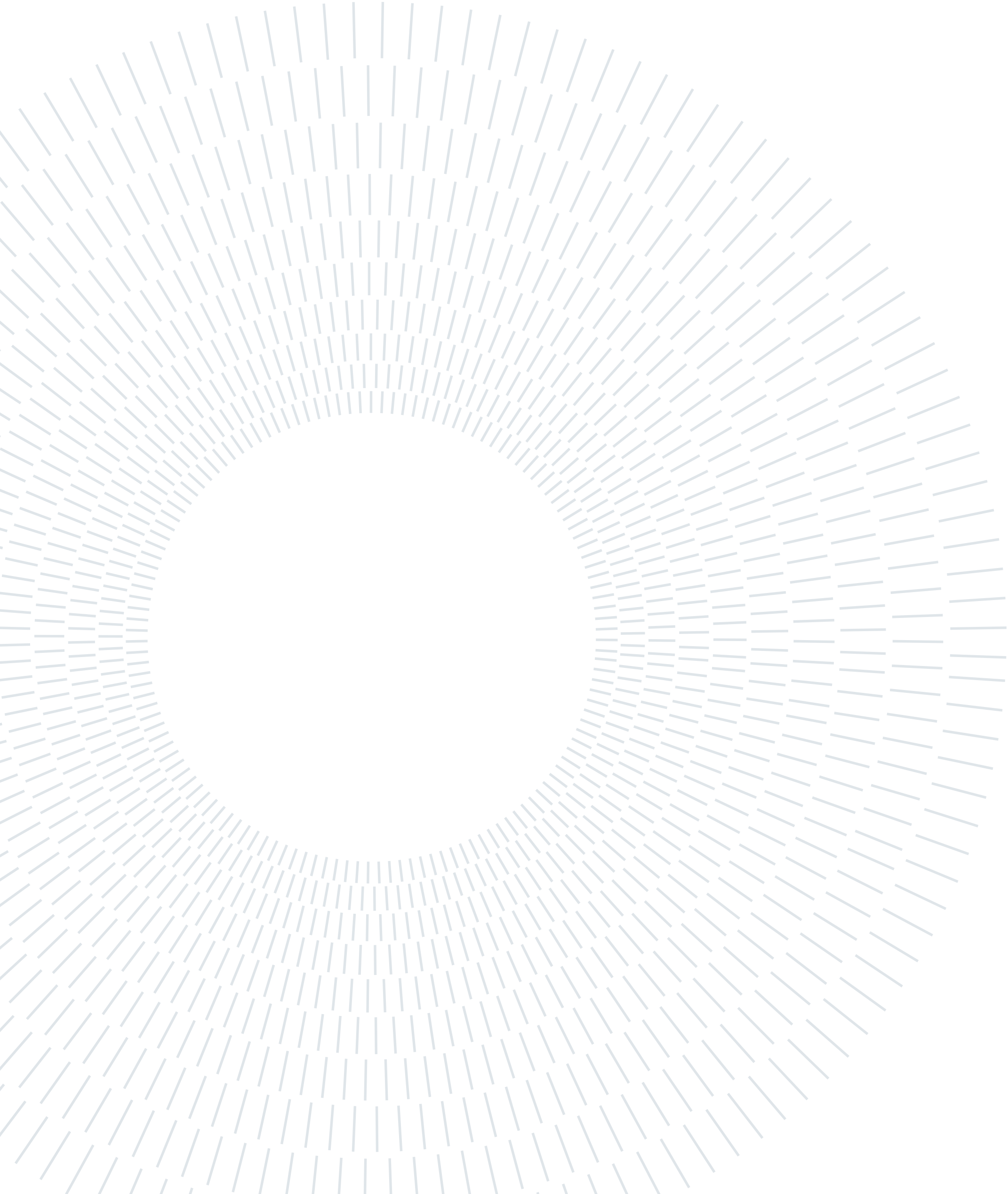
To my family



Abstract

A District Heating System (DHS) is an energy plant used to deliver heat through insulated water pipelines (District Heating Network or DHN), both for residential and for commercial purposes. This thermo-hydraulic system is particularly significant given that it is an important tool to reach energy transition targets. In the state-of-the-art DHSs literature, the vast majority of control strategies proposed are based on the physical model equations. This thesis, instead, aims at identifying a DHN model through a machine learning technique, namely Recurrent Neural Networks (RNNs), which are particularly suitable to identify the non-linearities of such a complex system. However, in order to exploit both data-driven methods and physical knowledge, a Physics-based Machine Learning approach is proposed. In particular, the key challenge of the thesis consists in developing a neural network that is capable of representing the physical interactions among the main elements of the system under investigation. This objective is achieved by making the Physics-based Recurrent Neural Network (PB-RNN) resemble the physical system topology. Ultimately, PB-RNNs turn out to improve standard RNNs in several respects: higher predictive accuracy, faster training procedure, greater interpretability and easier problem detection. After the identification of a performing system model, the latter is exploited in a Non-linear Model Predictive Control strategy to optimize the district heating system operation by minimizing its electrical cost. Finally, a lifelong learning algorithm is presented in order to monitor the DHS in the long run. In detail, the algorithm is intended to continually supervise the plant so as to detect potential anomalies. Depending on the type of scenario tracked, which can mainly be a structural plant modification or an operating conditions shift, a different solving strategy is proposed. The overall target is to refine the existing system model by acquiring continuous information, while preventing the new knowledge from significantly interfering with past data.

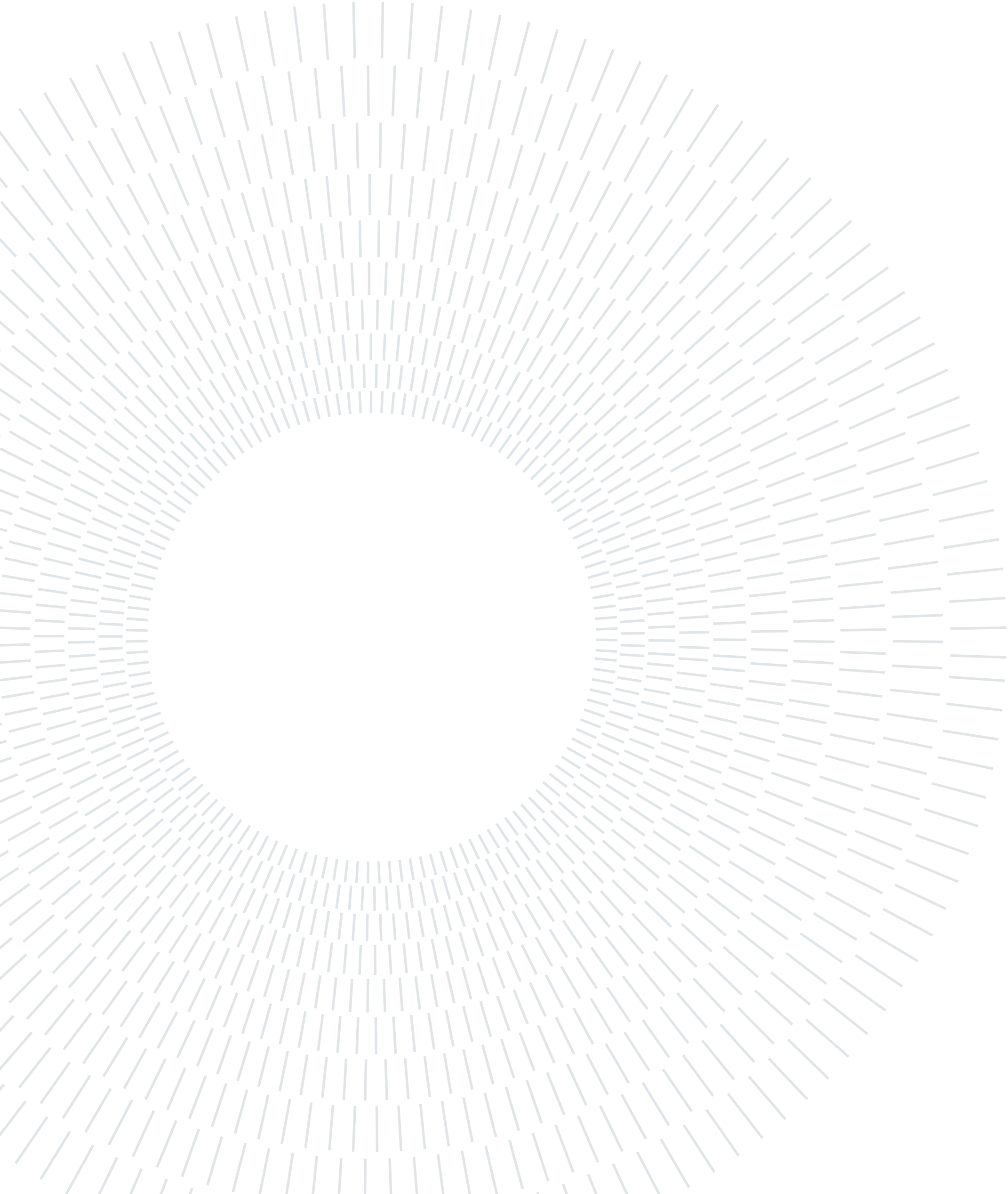
Keywords: Recurrent Neural Networks, Physics-based Machine Learning, District Heating Systems, Non-linear Model Predictive Control, Lifelong Learning.



Abstract in lingua italiana

Un sistema di teleriscaldamento è un impianto energetico utilizzato per distribuire calore attraverso delle condotte d'acqua isolate (rete di teleriscaldamento), a fini sia residenziali sia commerciali. Questo sistema termoidraulico è particolarmente significativo in quanto strumento chiave per raggiungere alcuni obiettivi della transizione energetica. Nell'attuale letteratura inerente ai sistemi di teleriscaldamento, la maggior parte delle strategie di controllo proposte è basata sul modello fisico del sistema. Questa tesi, invece, ha come obiettivo l'identificazione del modello di una rete di teleriscaldamento attraverso una tecnica di machine learning, ovvero le Reti Neurali Ricorrenti (RNN), che sono particolarmente indicate per identificare le non linearità di un sistema così complesso. Per sfruttare sia metodi guidati dai dati che dalla conoscenza fisica, un approccio di Physics-based Machine Learning viene in seguito proposto. In particolare, la sfida principale della tesi consiste nello sviluppare una RNN capace di rappresentare le interazioni fisiche tra gli elementi principali del sistema in analisi. L'obiettivo viene raggiunto facendo in modo che la Rete Neurale Ricorrente basata sulla fisica (PB-RNN) ricalchi la topologia del sistema fisico. In definitiva, queste PB-RNN risultano migliorare le reti tradizionali sotto diversi aspetti: maggiore accuratezza predittiva e comprensibilità, procedura di training più veloce e rilevazione dei problemi più agevole. In seguito all'identificazione di un modello performante, quest'ultimo viene impiegato all'interno di una strategia di controllo predittivo non lineare per ottimizzare il funzionamento del sistema di teleriscaldamento minimizzando il suo costo elettrico. Viene infine presentato un algoritmo ad apprendimento continuo per monitorare il sistema sul lungo periodo. Nel dettaglio, l'algoritmo è volto a supervisionare l'impianto in maniera continua così da individuare potenziali anomalie. In base al tipo di scenario rilevato, che può consistere principalmente in una modifica strutturale dell'impianto o in un cambio delle condizioni operative, viene proposta una diversa strategia risolutiva. Il target globale è quello di affinare il modello esistente attraverso una continua acquisizione di informazioni, evitando tuttavia che queste ultime interferiscano in maniera rilevante con i dati precedenti.

Parole chiave: Reti Neurali Ricorrenti, Physics-based Machine Learning, Sistemi di teleriscaldamento, Controllo predittivo non lineare, Apprendimento continuo.



Contents

Abstract **iii**

Abstract in lingua italiana **v**

Contents **vii**

Introduction **1**

1 Modelling and Simulation of a District Heating System **9**
1.1 Chapter Overview . . . 9
1.2 Simulation Environment . . . 9
1.3 Network Components Description . . . 10
  1.3.1 Gas Boiler . . . 10
  1.3.2 Pressure Pump . . . 12
  1.3.3 Expansion Vessel . . . 12
  1.3.4 Pipes . . . 13
  1.3.5 Thermal Loads . . . 14
1.4 Case study: the AROMA network . . . 15
1.5 Daily Simulation . . . 18
1.6 Conclusions . . . 21

2 Model Identification **23**
2.1 Chapter Overview . . . 23
2.2 Identification . . . 23
  2.2.1 Data . . . 23
  2.2.2 Candidate Models . . . 28
  2.2.3 Assessment Rule . . . 28
2.3 Prediction and Simulation . . . 29
2.4 State-Space and Polynomial Models . . . 30

2.4.1 State-Space Models 30
2.4.2 Polynomial Models 31
2.4.2.1 Autoregressive with External Input Models 31
2.4.2.2 Output-Error Models 31
2.4.3 Models Comparison 32
2.5 Recurrent Neural Networks 33
2.5.1 Long Short-Term Memory 34
2.5.2 Gated Recurrent Unit 35
2.5.3 Python Implementation 35
2.5.3.1 Hyperparameters 36
2.5.3.2 Training Method 37
2.5.3.3 Results Comparison 38
2.5.4 ARX and GRU Comparison 41
2.5.5 Identification of a Daily DHS Operation 42
2.6 Conclusions 44

3 Physics-based Recurrent Neural Networks 45
3.1 Chapter Overview 45
3.2 Physics-based Machine Learning 45
3.2.1 Motivations and Objectives 45
3.2.2 Categorization of PB-ML 46
3.3 PB-RNN Implementation for a DHN 48
3.3.1 Constrained Loss Function 48
3.3.2 A Novel RNN Architecture 51
3.3.2.1 Input Choice 52
3.3.2.2 Return Network 56
3.3.2.3 Overall PB-RNN with Graph Theory 57
3.4 Results Comparison 62
3.5 PB-ML Improvements over Traditional RNNs 66
3.6 Generalization of the PB-RNN Approach 66
3.7 PB-GRU Sensitivity Analysis 67
3.8 Conclusions 73

4 Model Predictive Control using Physics-based Neural Networks 75
4.1 Chapter Overview 75
4.2 Model Predictive Control 75
4.3 Non-linear MPC for a DHS using PB-RNNs 76
4.3.1 Optimization Environment 77

4.3.2 Problem Statement . . . 77
4.3.3 Disturbance Forecasting . . . 78
4.3.4 Constraints . . . 80
4.3.5 Cost Function . . . 82
4.3.6 Computational Effort Reduction . . . 83
4.3.6.1 Initialization . . . 84
4.3.6.2 Parameters Settings . . . 84
4.3.6.3 Blocking Strategy . . . 87
4.3.7 State Observer . . . 88
4.4 Optimization Results . . . 88
4.5 Conclusions . . . 91

5 Lifelong Learning 93
5.1 Chapter Overview . . . 93
5.2 Lifelong Learning Overview . . . 93
5.2.1 Motivations and Objectives . . . 94
5.2.2 Plasticity and Lifelong Learning in Literature . . . 94
5.3 A Lifelong Learning Algorithm for the Supervision of a DHS . . . 96
5.3.1 General Algorithm Description . . . 97
5.3.2 Plant Change Scenario . . . 103
5.3.2.1 Moving Horizon Estimation . . . 106
5.3.2.2 Results . . . 108
5.3.3 Operating Conditions Change Scenario . . . 112
5.3.3.1 Model Uncertainty Identification . . . 113
5.3.3.2 Results . . . 115
5.4 Long-term Monitoring . . . 120
5.5 Conclusions . . . 121

Conclusions and Future Developments 123

Bibliography 125

List of Figures 135

List of Tables 139

List of Parameters 141

List of Variables **143**

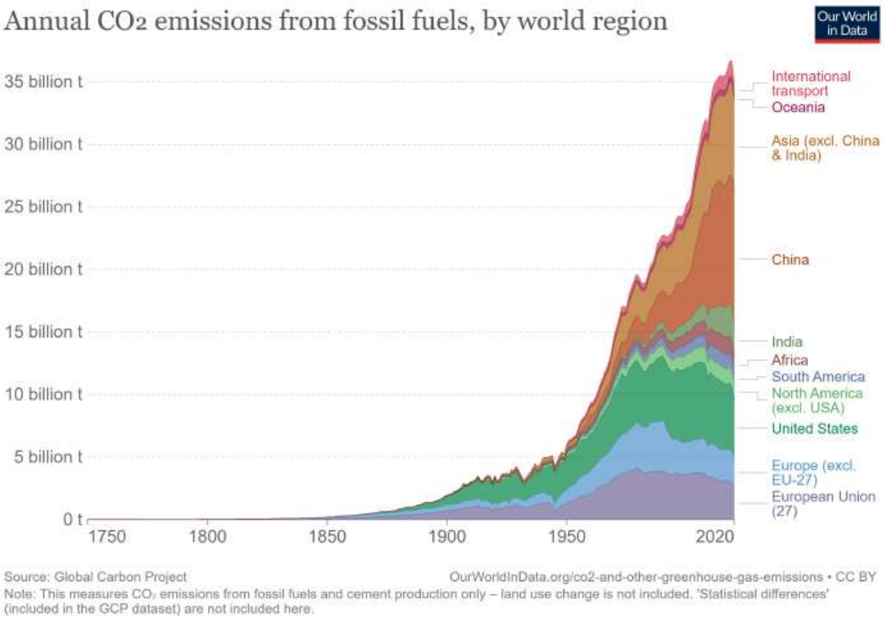
List of Acronyms **145**

Acknowledgments **147**

Introduction

Motivations

This past year has been a particularly tough period in several respects: to name one, surging energy prices have harshly hit Europe. The primary cause of the crisis is a rebound from an economic slowdown during the COVID-19 pandemic. Russia's invasion of Ukraine in February 2022 worsened the situation. Both European sanctions and Russian retaliation crimped supplies of Russian natural gas, which powers electric generators and heats buildings, pushing continental European gas prices to more than 10 times their average historical values[1]. This crisis remarked the importance of energy sources diversification, also useful to speed up energy transition. Such turnaround is indeed crucial to cope with another complex and well-established issue of nowadays, i.e. the alarming continuous growth of CO_2 emissions (Figure 1).



Annual CO2 emissions from fossil fuels, by world region

Source: Global Carbon Project

OurWorldInData.org/co2-and-other-greenhouse-gas-emissions • CC BY

Note: This measures CO_2 emissions from fossil fuels and cement production only – land use change is not included. 'Statistical differences' (included in the GCP dataset) are not included here.

Figure 1: CO_2 emissions by region.

---

[1] `www.science.org`

Fortunately, the European Union is among the leading major economies when it comes to tackling greenhouse gas (GHG) emissions. By 2019, it had cut GHG emissions by 24% compared to 1990 levels. In order to put the EU on a balanced pathway towards carbon neutrality by 2050 (in line with the Paris Agreement), in April 2021 the Commission agreed to raise the climate ambition of the GHG emission reduction target from 40% to 55% by 2030 compared to 1990 levels[2].

The path to reach these objectives is demanding and cost-intensive, but more and more solutions are now at our disposal. To mention a few, the usage of renewable and nuclear energy, the electrification and sharing of transportation and the efficiency improvement of private and industrial buildings. For instance, a recognized solution is to enhance urban district heating, increasing it from the actual value of 20% to around 50% of the total heat demand in Europe [45].
Due to all these reasons, this thesis aims at modelling, controlling and monitoring a district heating system via groundbreaking techniques.

District Heating Systems

A District Heating System (also known as teleheating and abbreviated to DHS) is a network used to distribute the heat generated in a centralized location (heating station) through insulated water pipelines (District Heating Network or DHN), both for residential and commercial requirements, such as space and water heating. District heating systems exploit various energy sources, sometimes indirectly through multipurpose infrastructure such as combined heat and power plants (CHP)[3].

At the EU level, District Heating and Cooling (DHC) with CHP production are pointed out as “important tools” to reach energy transition targets. The International Energy Agency also claims the DHC system as a leading technology in a future (2050) where a projected 6.3 billion people worldwide will live in cities [42].
District heating systems, in fact, highly contribute to primary energy savings as well as emission and air-pollution reductions in urban areas [29, 69], mainly through the use of high efficiency plants able to combine electricity and heat production, renewable energy sources [51] and waste heat recovering from industrial processes [23, 80]. In addition, along with a reduction in the maintenance and safety expenses [81], DHSs are also capable of providing flexibility by balancing supply and demand through Thermal Energy Storage (TES) systems [54].

---

[2] `www.europarl.europa.eu`

[3] `www.wikipedia.org`

Despite the aforementioned benefits of a DHS, its market share around the world is still confined: the predominant reason for neglecting such systems is the lack of suitable tools to design, analyse, and optimize them [81]. Because of that, this thesis seeks to develop identification and control methods applied to such a powerful and worthwhile system.

Generally speaking, a DHS consists of four main parts [43] (Figure 2):

- a forward-flow part, which provides hot water to consumers (the *supply* network);
- consumers, that use hot water for heating;
- a backward-flow part, which transports the cooled water back to the heating station (the *return* network);
- and the heating station, where the warming of the cooled water takes place.

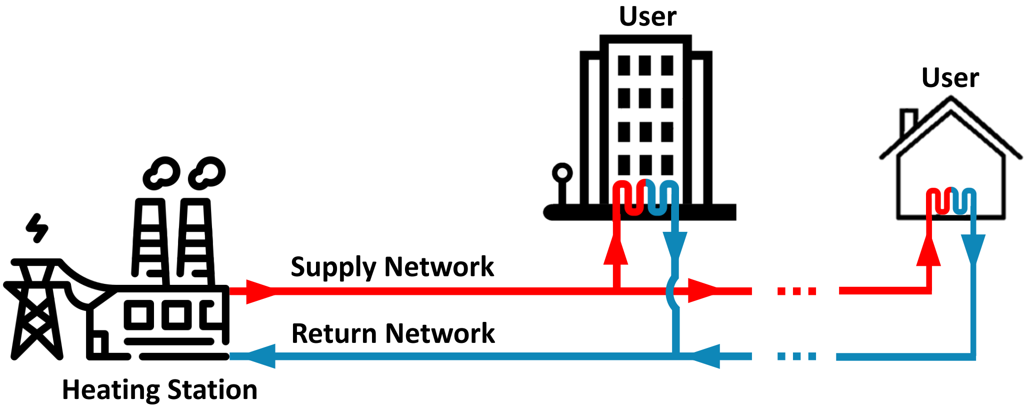


Figure 2: A schematic example of a district heating system [44].

A remark regarding nomenclature: generally speaking, a "district heating system" is referred to as the comprehensive plant composed of pipelines, generators, consumers plus other components such as storages, pumps and vessels. By contrast, for "district heating network" (DHN) the water pipelines network connecting thermal loads is typically meant. Therefore, we may say that in this thesis we are going to control a district heating system whose DHN model has been identified through different techniques.

Recurrent Neural Networks and Physics-based Machine Learning

Traditionally, DHSs, due to their modelling complexity, are not optimized for control purposes, but rather their supply temperature is kept constant. However, many efforts have been recently made in order to control district heating systems through standard model-based techniques, which involve the design and tuning of a control system based on the knowledge of the physical model (white-box methods). In particular, DHSs require advanced management and control schemes to be efficiently and optimally operated, considering various factors such as production costs, heat losses and environmental impacts [45]. Actually, the mathematical equations describing such systems are complex, highly non-linear (water transport delays are non-linear functions of the water flow [45]) and, due to the uncertainties of such a wide network, the resulting model is frequently inaccurate.

This is where the need to exploit some advanced data-driven techniques such as Neural Networks (black-box methods), with a complexity suitable for control, is born. The just mentioned approach is related to supervised Machine Learning (ML) methods where measurements of input and output variables of a process are collected, and then used to mathematically describe the system [27, 35]. In recent years Neural Networks (NNs) have spread as a powerful tool in several fields of science and engineering [9, 47], such as time series forecasting [4] and system diagnosis [25]. The reason behind this rising interest lies in the many potential advantages of black-box methods over traditional algorithms that require a physical knowledge of the plant, including the possibility to reduce the time and cost associated with model design, tuning, and adaptation to new plant operating conditions [11].

In detail, a Recurrent Neural Network (RNN) is a special type of artificial neural network particularly suitable to process time series data. They are in fact able to model long-term temporal dependencies between subsequent data samples by introducing a *hidden state* and by modelling the information flow into and out of the hidden state using the so-called *gates* [73]. The fundamental feature of an RNN is that it contains at least one feedback connection: in this way, the activations can flow round in a loop enabling temporal processing and sequences learning [13].

On the other hand, purely black-box methods may lack of physical interpretability and consistency and they typically require a huge amount of data to yield reliable results. As a consequence, there is a growing interest in the scientific community to integrate physical knowledge in machine learning frameworks in order to deliver generalizable and scientifically consistent solutions even in the scarcity of representative data [39]. The goal of the Physics-based Machine Learning (PB-ML) approach is to combine the benefits of model-based and data-driven methods, and therefore to merge scientific knowledge with machine learning techniques (see Figure 3).

Typically, leveraging physical knowledge of the system generates a model which is not only more reliable, but also less prone to over-fitting. In addition, this can be achieved with a significantly faster convergence of the training procedure [11].

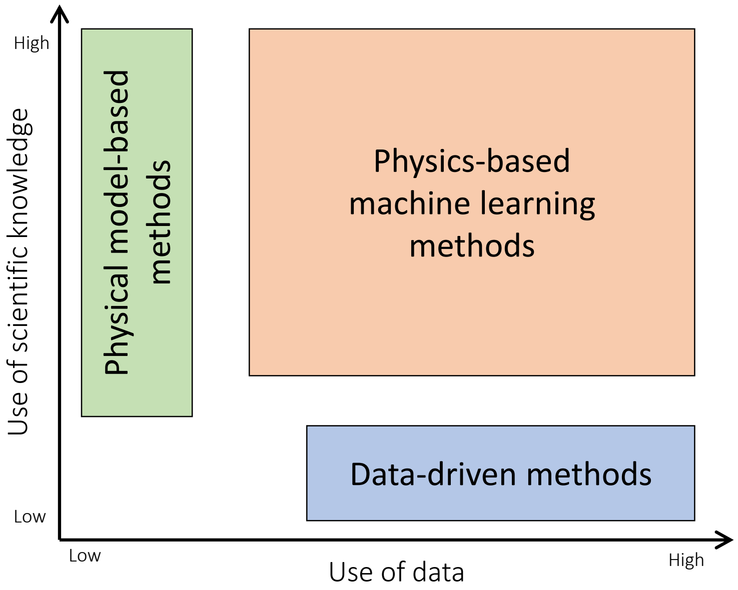

High
Use of scientific knowledge
Low
Physical model-based methods
Physics-based machine learning methods
Data-driven methods
Low
Use of data
High

Figure 3: A schematic description of physics-informed machine learning use of scientific knowledge and data with respect to standard methods [39].

Model Predictive Control

Once a plant model has been identified, a further key step consists in the optimization and control of the system under discussion, given that every thermal network requires online monitoring and feedback control.

One of the most effective control techniques is the notorious Model Predictive Control (MPC), i.e. an advanced strategy for optimizing the performance of multivariable control systems. In a nutshell, MPC produces control actions by optimizing a cost function repeatedly over a finite moving prediction horizon, based on the dynamic model of the system to be controlled, while fulfilling physical constraints. First implementations of MPC can be traced back to the late seventies. Since then, MPC research and development have grown significantly in both industries and academia [91].

Indeed, MPC requires a reasonably accurate model that captures the dynamics of the plant under control, so as to predict the optimization variables behaviour multiple steps ahead [89].

Different system models can be embedded in the design of model predictive control, ranging from highly detailed or simplified physical equations to data-driven solutions.

In particular, a key issue for an MPC implementation lies in the related online optimization: its success depends on the effectiveness and efficiency of the solution method used. For instance, neurodynamic optimization leveraging recurrent neural networks emerged in scientific literature as a very promising approach [91]. Interesting examples of MPC exploiting RNNs can be found in different fields, such as pharmaceutical manufacturing [89] or flight engineering [93].

However, district heating systems are usually controlled through MPC regulators relying on purely physics-based models [24, 67, 70, 86, 88], whereas, to the best of the author's knowledge, never through MPC based on physics-informed machine learning methods.

Lifelong Learning

After a system model is obtained and control tuning is performed, another key issue arises. In real life, in fact, processes change during time because of external (e.g. disturbances and new implemented technologies) and internal factors (e.g. wear and fouling in the equipment). As a consequence, the machine learning model that has been trained using the information from past normal operations may no longer be able to correctly predict actual process states [90]. Hence, in many practical applications it is required to adapt the model when new information is available and/or the system undergoes changes [12].

A key aspect of human intelligence is the ability, namely lifelong learning, to continually adapt and learn in dynamic environments [83]. By mimicking nature, we may extend this idea to machine learning. Actually, this represents a long-standing challenge for neural network systems, due to the tendency of learning models to catastrophically forget existing knowledge when acquiring new data. Specifically, a lifelong learning system is defined as an adaptive algorithm capable of learning from a continuous stream of information, with such knowledge becoming progressively available over time and where the number of tasks to be learned is not predefined. Critically, the accommodation of new information should occur without catastrophic forgetting or interference [63].

Thesis Contributions

By pursuing the state-of-the-art literature achievements, this thesis aims at giving a contribution under four main respects.

The first objective is the modelling of a district heating network by means of black-box identification techniques, and in particular through recurrent neural networks. To the best of the author's knowledge, in fact, this type of thermo-hydraulic systems is typically monitored and controlled via physics-based (white-box) methods.

However, the main challenge of the work consists in developing and implementing a recurrent neural network whose structure is not "blind" but rather oriented by the physics of the actual system. To this end, the interconnections among the RNN layers are intended to replicate the physical network architecture and modularity. The desired goal of such physics-based machine learning approach is to reach greater identification performance with respect to standard RNNs, hopefully in a shorter amount of training time.

Furthermore, the thesis aims at devising a non-linear model predictive control strategy for the DHS under discussion, by making use of the physics-based recurrent neural network equations both in the finite horizon control optimization problem formulation and in the state observer design.

The final contribution is related to the lifelong learning issue. In the long run, the district heating system under investigation must be continually monitored and supervised: the thesis proposes a novel algorithm to tackle the problem of model changes over time. In particular, the target is to provide an effective solution depending on the category of scenario detected.

Thesis Outline

The first chapter presents the modelling of a district heating system, along with a brief description of its major elements and of the simulation environments employed. Finally, a typical daily operation of the network under consideration is discussed.

The second chapter aims at giving a first insight on the model identification topic. An overview on some state-space and polynomial models, along with their performance, is then provided. Similarly, two types of recurrent neural networks, together with their implementation and predictive results, are explored. Finally, a typical working day of the DHS is simulated by means of the most successful identification technique spotted up to that point.

The third chapter, in addition to a short literature review on physics-based machine learning, seeks to underline the advantages of this novel approach over traditional data-driven and model-based methods. Moreover, two strategies applied to the case study under investigation are deepened. In particular, the rationale behind the second one, which turns out to be the most effective, and its implementation, i.e. a recurrent neural network that resembles the structure of the physical system, are thoroughly described. Finally, the results of this innovative technique are compared to the standard ones and a sensitivity analysis is carried out.

The fourth chapter focuses on the implementation of an optimization and control algorithm, namely model predictive control. After a general introduction and the problem statement, the main elements required to solve the finite horizon control optimization problem are investigated: cost function, constraints, disturbance forecasting and state observer. Finally, the results of the aforementioned problem are reported, together with a comparison between two MPC regulators that make use of different plant models.

The fifth chapter involves a general description of lifelong learning, complemented by its motivations, objectives and a brief literature review on the topic. Subsequently, a general supervision algorithm for the monitoring of a district heating system is proposed. In particular, the strategy articulates in two parts, depending on the category of anomaly detected. For each scenario, the corresponding solving algorithm and identification results are finally reported.

The final chapter summarizes the main conclusions and achievements of the thesis. Moreover, although the results accomplished are rewarding, the work is nonetheless a starting point for many possible reflections and future developments, which are lastly discussed.

1 | Modelling and Simulation of a District Heating System

1.1. Chapter Overview

The main goal of this chapter is to present the modelling of a District Heating System (DHS) and to describe its main components. In particular, a case study will be discussed (the AROMA network [43]), both in this and in subsequent chapters. Then, for identification and control purposes it is indeed necessary a suitable simulation environment in which a model able to capture the main dynamics of the physical system is developed. Actually, the data used in this thesis for identification scopes are not collected from a real operating system, but rather they are obtained through an accurate simulation which is here discussed. Finally, a realistic example of a daily operation will be shown.

1.2. Simulation Environment

The choice of a proper simulation environment is the first crucial step of this thesis. In very few words, it is useful to exploit a software which, given certain system inputs, such as temperature and power references, is capable of simulating the process dynamics easily and quickly.

The system taken into account is a thermal one and hence the object-oriented **Modelica** language is suitable to describe a district heating system. The Modelica Association is a non-profit organization which develops coordinated, open access standards and open source software in the area of cyber physical systems[1]. The Association also develops the free Modelica Standard Library, but for this thesis objectives another library is exploited: the *DHN4Control* developed by the Systems and Control Group of Politecnico di Milano.

Second, after the system model is implemented, it is exported from the OpenModelica Connection Editor as an FMU file and imported in **Simulink** for control purposes. The latter is a block diagram environment for multidomain simulation and Model-Based Design integrated in MATLAB[2].

---

[1] `https://modelica.org`

The reason why two simulation environments are simultaneously employed is quite straightforward. The Modelica language fits perfectly the need of describing a complex thermo-hydraulic system, whereas Simulink and MATLAB are more appropriate for identification and control purposes.

1.3. Network Components Description

In this section, the main elements composing a district heating system are concisely presented. Besides, the well-known mass, momentum and energy balance equations that govern the various components are not here entirely developed for the sake of conciseness, but they are thoroughly described in [43, 61]. Finally, the parameters and variables symbols mentioned in this chapter are listed, together with their meaning and International System (SI) unit, in Table 5.7 and 5.8.

1.3.1. Gas Boiler

In short, the gas boiler is a heater which provides heat to the network flowing water, i.e. it sets the desired supply temperature. Even though in recent years its thermal efficiency has significantly enhanced [66], it may not be always the best solution due to its emissions and human health-related problems [33, 48]. However, for this thesis purposes, simple and widely used solutions such as traditional gas boilers are entirely appropriate: they are particularly flexible and easy to control [61], which is a sufficiently fair reason to choose them.

The general idea behind a gas boiler is to transfer the chemical power stored inside natural gas into water, so that its temperature increases or a phase change occurs [22, 61].
The Modelica gas boiler block is reported in Figure 1.1.

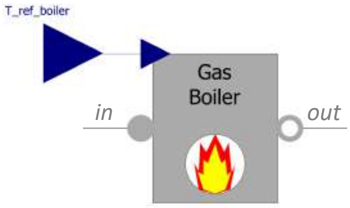


Figure 1.1: The gas boiler representation in the *DHN4Control* Modelica library.

---

[2] `https://it.mathworks.com`

As it can be noticed, the block has one input (depicted as a blue triangle), i.e. the reference temperature, which is the value the water inside the boiler must reach. In particular, this is achieved through a Proportional-Integral-Derivative (PID) with anti-windup controller, by measuring the temperature via a sensor placed in the output flow of the heater and by giving this quantity in feedback to the PID regulator.
Additionally, the power exiting from the regulator as control action is regularly monitored. It is worth recalling that it is constrained between a minimum and a maximum value (1.1), and it must always satisfy the consumers request.

$$P_{min}^{boiler} \leq P^{boiler} \leq P_{max}^{boiler} \tag{1.1}$$

In addition, many library components are characterized, as in Figure 1.1, by a flow inlet port (grey circle on the left-hand side) and an outlet one (grey ring on the right-hand side).

Second, it is useful to demonstrate how the integral time (T_i^{PID}) of the controller is computed. Actually, the water volume (V) dynamics inside the boiler is not negligible and in order to have an almost first order response of the controlled system it is convenient to cancel that slow and disturbing dynamics out through a careful choice of T_i^{PID}. By simply exploiting an energy balance (1.2), it is possible to derive the perfectly mixed volume time constant.

$$M_w c_p \frac{dT_w}{dt} = \dot{m}_w c_p (T_{in} - T_{out}) - Q_{amb} \tag{1.2}$$

In this case, the thermal conductance that characterizes the heat exchange with the external environment (Q_{amb}) can be considered zero, by fairly assuming that the boiler is well insulated. As a consequence, the water volume dominant time constant is readily found in (1.3) and it can be directly used as integral time of the PID controller.

$$\tau = \frac{\Delta T_w}{\dot{T}_w} = \frac{\rho V}{\dot{m}_w} \tag{1.3}$$

Finally, the proportional gain (k^{PID}) of the regulator is chosen so that the settling time of the outlet temperature is roughly one minute and a half [78] and neither overshoots nor oscillations occur. The complete set of equations governing the boiler dynamics is reported in [61].

1.3.2. Pressure Pump

The pump is a fundamental element used to impose a constant differential pressure between its inlet and outlet port without losses and at any mass flow rate. This feature is particularly helpful in heating applications, since thermal loads absorb an unknown amount of water in order to meet their power demands and to maintain a fixed return temperature [61].

Specifically, considering the typical power consumption profiles and mass flow rates, the value of p_{pump} reported in Table 5.7 is required. Moreover, in (1.4) the equations governing the pump dynamics are shown [61].

$$\begin{cases} p_{out}^{pump} - p_{in}^{pump} = \Delta p^{pump} \\ T_{in}^{pump} = T_{out}^{pump} \\ \dot{m}_{in}^{pump} + \dot{m}_{out}^{pump} = 0 \end{cases} \tag{1.4}$$

Lastly, in Figure 1.2 the *DHN4Control* block representing the pressure pump is displayed.

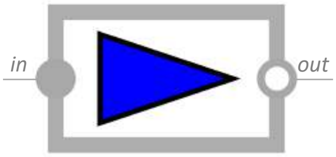


Figure 1.2: The differential pressure pump in the *DHN4Control* Modelica library.

1.3.3. Expansion Vessel

The expansion vessel (or accumulator) is an important part of any water heating system. Briefly, it is a pressure storage tank that imposes a certain p_{vess} in the backward-flow part of the DHS, avoiding fluctuations [61]. In other words, the pressure accumulator absorbs volume variations of the fluid caused by temperature changes and prevents cavitation in circulation pumps [77].

Besides, it is possible to assume a perfect insulation and negligible friction, and therefore the temperature of the volume inside the tank is equal to the return water temperature [61], i.e. T^r. As visible in Figure 1.3, the tank has a unique port which is connected to the backward-flow part only, and it imposes to the latter the pressure value p_{vess} reported in Table 5.7.

Exp.
Tank
in/out

Figure 1.3: The expansion vessel in the *DHN4Control* Modelica library.

Ultimately, since the expansion tank fixes the return pressure to a desired value, all pressures in the network are potentially known at any time [61]. As a consequence, it is possible to neglect the pressure dynamics and to focus on the temperature and mass flow rate ones.

1.3.4. Pipes

Water is delivered to users through a pipeline network (from which derives District Heating Network or DHN) that is modelled so that the metal thermal inertia, friction, flow reversal and pressure losses are taken into account: pipes are in fact the elements that introduce transport delay effects on temperature profiles [61].

In detail, it is possible to describe the physics of hot water flow in pipelines through 1D Euler equations. The thermal energy equation for each j^{th} pipe is given by (1.5) [43].

$$\frac{\partial T_j^{pipe}}{\partial t}(x,t) + u_j^{pipe}(t)\frac{\partial T_j^{pipe}}{\partial x}(x,t) + \frac{4U_j^{pipe}}{c_p \rho_j^{pipe}(x,t) D_j^{pipe}}(T_j^{pipe}(x,t) - T_{ext}) = 0 \tag{1.5}$$

Moreover, the model of each pipe can be discretised in sections to get rid of space derivatives [61, 65] (finite volume method, see Figure 1.4). In particular, since modelling complexity is out of this thesis scope, for the sake of simplicity only two sections per pipe are considered.

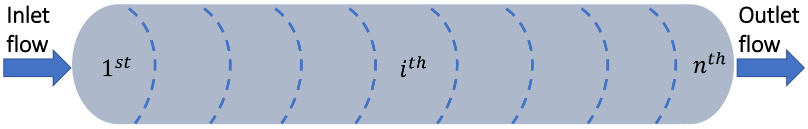


Figure 1.4: Finite volume method: discretisation of a pipe with n sections.

Overall, in the AROMA network there are nine pipes along the forward-flow part and nine along the backward-flow part: each of them has its own internal diameter and length according to [43].

For a better description of reality, a pipe insulation thickness d_{ins} and a thermal conductivity σ_{ins} are considered (Table 5.7). In particular, a thick pipelines insulation allows to mitigate the losses. In addition, the thermal conductance which describes the heat exchange between each pipe and the external environment can be found as in (1.6).

$$UA_j^{pipe} = \frac{L_j^{pipe}(2\pi\sigma_{ins})}{log\frac{D_j^{pipe}/2+d_{wall}+d_{ins}}{D_j^{pipe}/2+d_{wall}}} \tag{1.6}$$

Finally, according to the Reynolds number (Re) [61], the Fanning friction coefficient (f) is constant, as well as the nominal fluid velocity (u_{nom}).

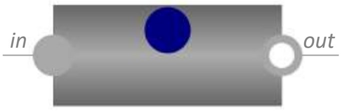


Figure 1.5: A generic pipe in the *DHN4Control* Modelica library.

1.3.5. Thermal Loads

The AROMA network involves five users that consume water for heating purposes. Actually, each consumer is an aggregation of loads that could be houses, industrial facilities or commercial buildings [61]. Each load is modelled as reported in Figure 1.6, where the valve temperature reference and the load consumption are the inputs, whereas the supply and return temperature and the mass flow rate are the outputs (internally visible).

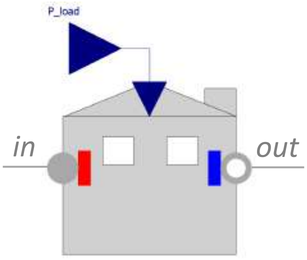


Figure 1.6: A generic consumer in the *DHN4Control* Modelica library.

Typically, the load powers can range from P_{min}^{load} to P_{max}^{load}, whereas T_{ref} is kept steady at the cold temperature of 65°C through a mere valve proportional controller, as visible in

Figure 1.7. The Modelica scheme shows also the temperature sensor block, depicted as a thermometer. Moreover, one can notice that the valve connects the user in series to the main flow line [61].

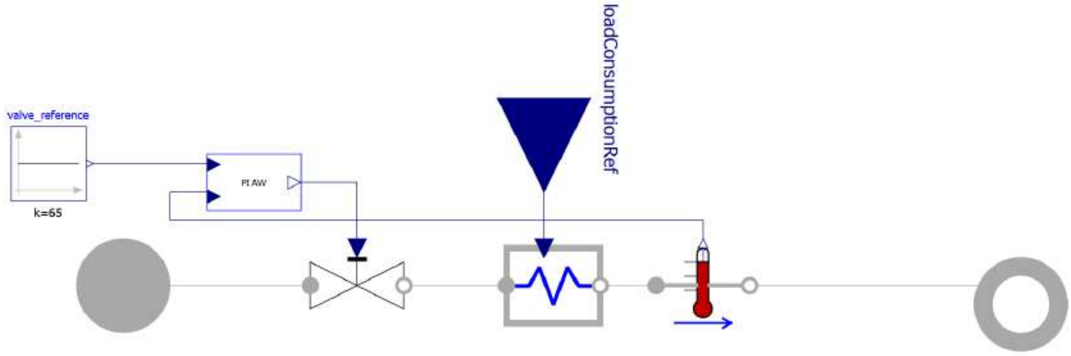


Figure 1.7: Valve proportional controller implemented in Modelica.

The valve is reasonably assumed to be linear, with constant pressure recovery coefficient, and it is necessary to regulate the mass flow rate: a key quantity is indeed its flow coefficient. In depth, the flow factor (K_v) is the metric system equivalent of the flow coefficient (C_v), and it is defined as the flow of water in cubic meters per hour ([m^3/h]) at a pressure drop of 1 bar with the temperature ranging from 5 to 30°C. It describes the relationship between the pressure drop across a valve and the corresponding flow rate[3]. Moreover, a minimum opening area of the valve (θ_{min}) is imposed in order to avoid no flow condition. Clearly, the pressure drop between the supply and the return part is the one imposed by the differential pump.

To sum up, the users demand regulates the overall mass flow rate of the network, whose relationship is given in (1.7) [45].

$$P_i^{load}(t) = c_p \dot{m}_i(t)(T_i^s(t) - T_i^r(t)) \tag{1.7}$$

1.4. Case study: the AROMA network

Now that the main elements composing a DHS have been described, a case study which includes all of them can be introduced. In fact, as explained in the Introduction, in this thesis all the experiments are carried out on a specific district heating system: a referenced DHN named AROMA network is implemented [43]. The main physical aspects of water and heat flow in such a network are governed by non-linear and hyperbolic 1D partial differential equations, thoroughly explained in [43].

---

[3] `www.wikipedia.com`

The topology of the AROMA network is rather simple: it is composed of 18 nodes, 24 arcs (1 heating station, 5 consumers, and 18 pipes), and one cycle each in the forward-flow and in the backward-flow network. Overall, its total pipe length is 7262.4 m. In Figure 1.8 a schematic representation highlighting the nodes and the thermal loads is displayed.

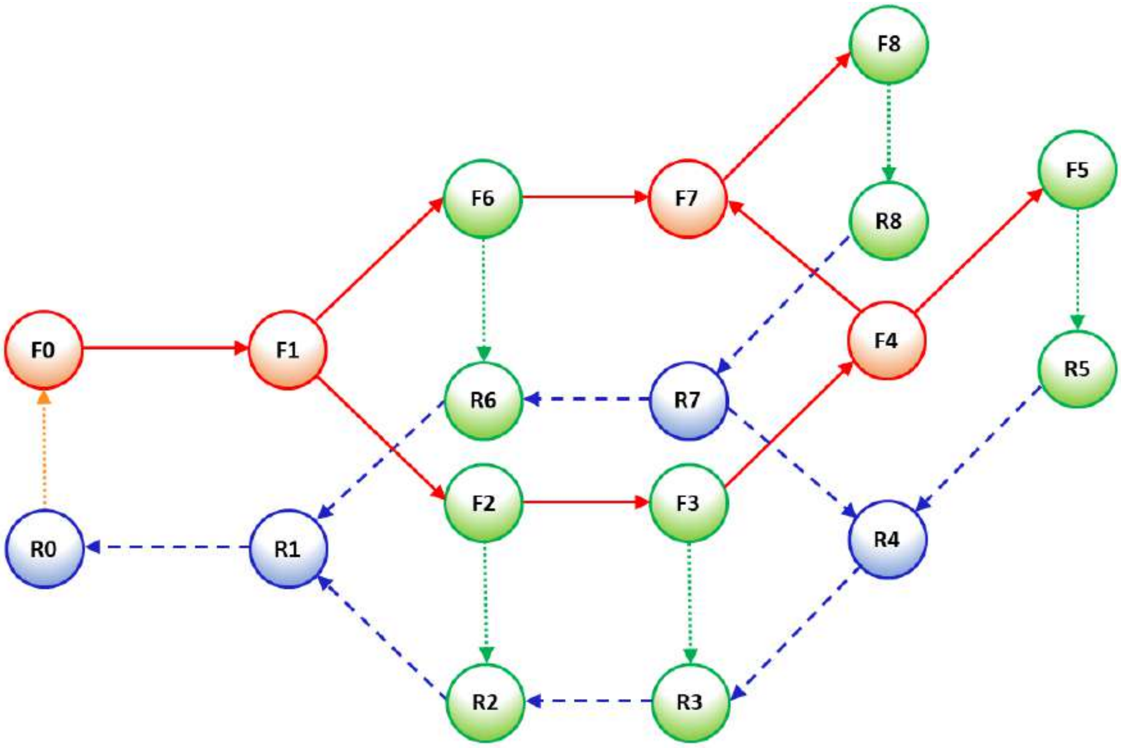


Figure 1.8: AROMA network schematic representation: forward-flow arcs are plotted in solid red, backward-flow arcs in dashed blue, users in green and the heating station in dotted yellow. Nodes in the forward part are referred to as Fi, in the return part as Ri.

Lastly, by assembling in Modelica all the components illustrated in Section 1.3 through connections that link outlet and inlet ports, the overall AROMA network is thus obtained (Figure 1.9). The loads numbering reported in the diagram is crucial and it will be used hereinafter to mention the various users. In particular, differently from the notation of [43], we number the consumers according to their distance with respect to the heating station: the first user is the closest, the fifth (last) is the furthest.

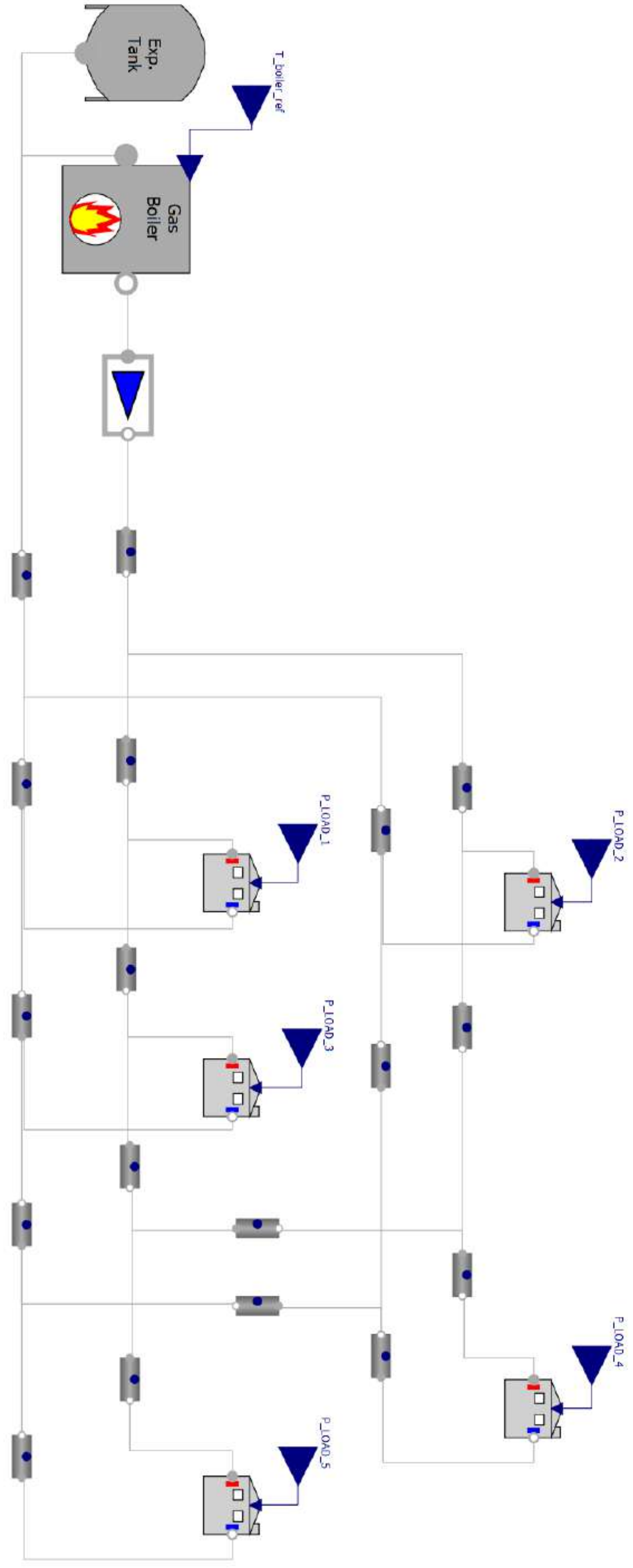

Exp. Tank
T_boiler_ref
Gas Boiler
P_LOAD_1
P_LOAD_2
P_LOAD_3
P_LOAD_4
P_LOAD_5

Figure 1.9: The complete AROMA network implemented in Modelica.

In depth, all the consumers' valve controllers have the same temperature reference, whereas the load profile inputs are separated as different users typically have different power requests. Moreover, for a better manageability, the outputs are collected into vectors. Ultimately, temperature, mass flow rate and pressure sensors are placed in key points of the network to regularly monitor its status.

1.5. Daily Simulation

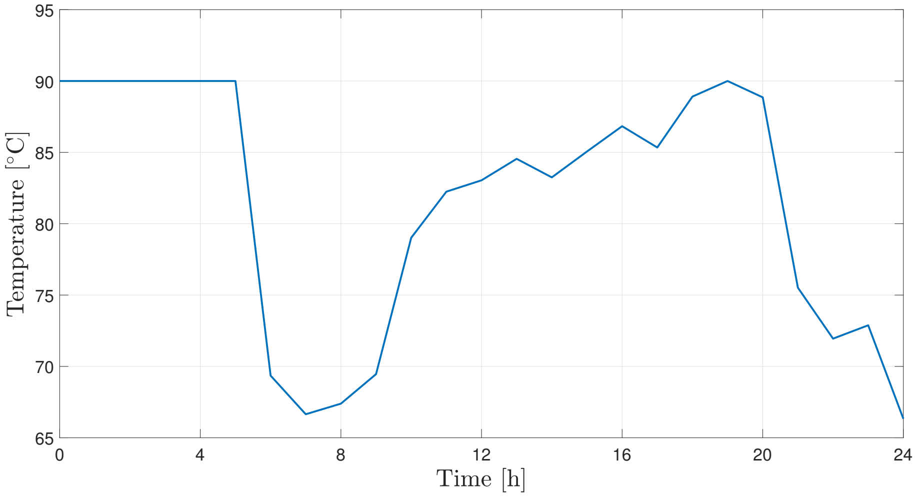
In this section, a typical operating day of the AROMA network is discussed. The data used for the simulation are the ones of a real district heating system located in Novate Milanese [45].

In detail, the system is fed with six inputs: the boiler reference temperature and the thermal loads consumptions. First, T_{ref}^{boiler} is the one obtained through the resolution of a dynamic optimization problem analysed in [61]. Then, the consumers powers reported in [45] are actually adapted to the AROMA network case according to the proper power range ($P_{min}^{load} - P_{max}^{load}$) and to the weights reported in Table 1.1 [43].

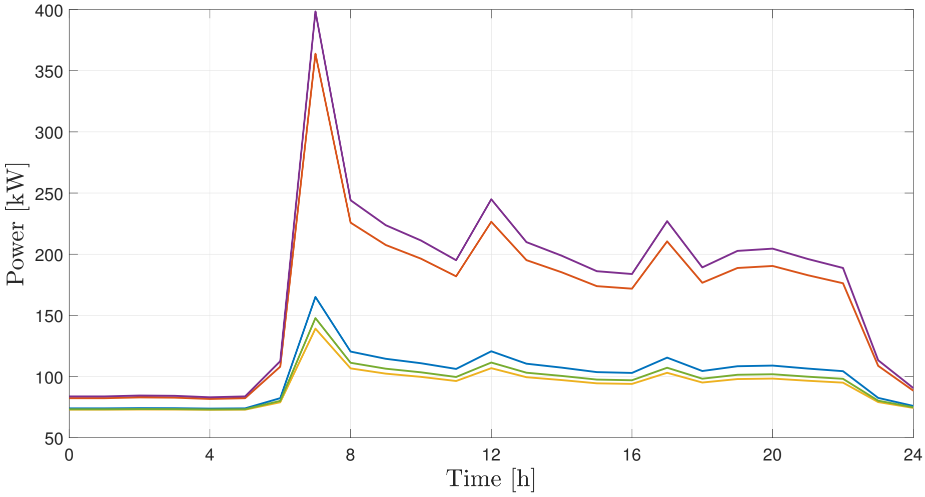
Consumer	**Weight**
Load 1	0.11
Load 2	0.38
Load 3	0.34
Load 4	0.09
Load 5	0.08

Table 1.1: Weights of the AROMA network describing how the overall power request is partitioned among the five consumers.

Ultimately, the so formed input signals are shown in Figure 1.10.



(a) Boiler temperature.

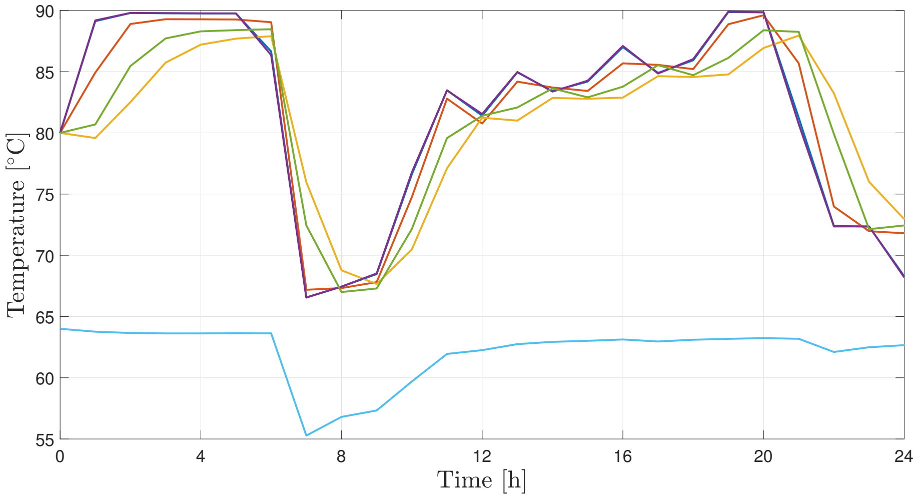


(b) Load profiles: in blue the first user's power, in purple the second's, in red the third's, in green the fourth's and in yellow the fifth's.

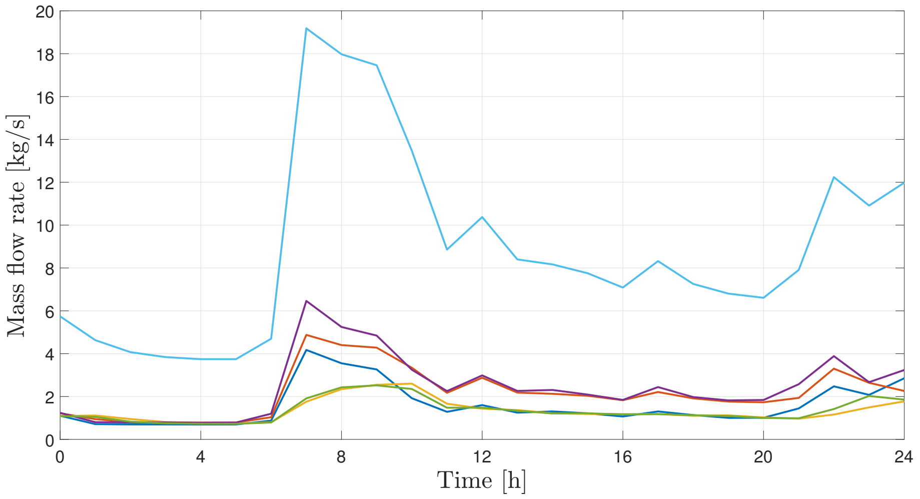
Figure 1.10: Daily simulation inputs.

It is worth noticing that the upwards peak in the load profiles and the downwards peak in the boiler reference temperature in the early morning hours are due to an obvious larger use of hot water by consumers with respect, for instance, to the night-time. Besides, the initial (night-time) high value of T_{ref}^{boiler} is intuitive and correctly computed by the optimization algorithm: in this way the network is prepared to tackle the morning demand peak.

Second, once the input quantities are defined, it is possible to run a daily simulation, whose results are shown in Figure 1.11. In depth, the output quantities of major interest are the following: the five users supply temperatures and return mass flow rates, as well as the overall return temperature and mass flow rate at the central station.



(a) Users supply temperatures and overall return temperature.



(b) Users return mass flow rates and overall return mass flow rate.

Figure 1.11: Main AROMA network outputs in a daily simulation: in blue the first user's variables, in purple the second's, in red the third's, in green the fourth's, in yellow the fifth's and in light-blue the return variables.

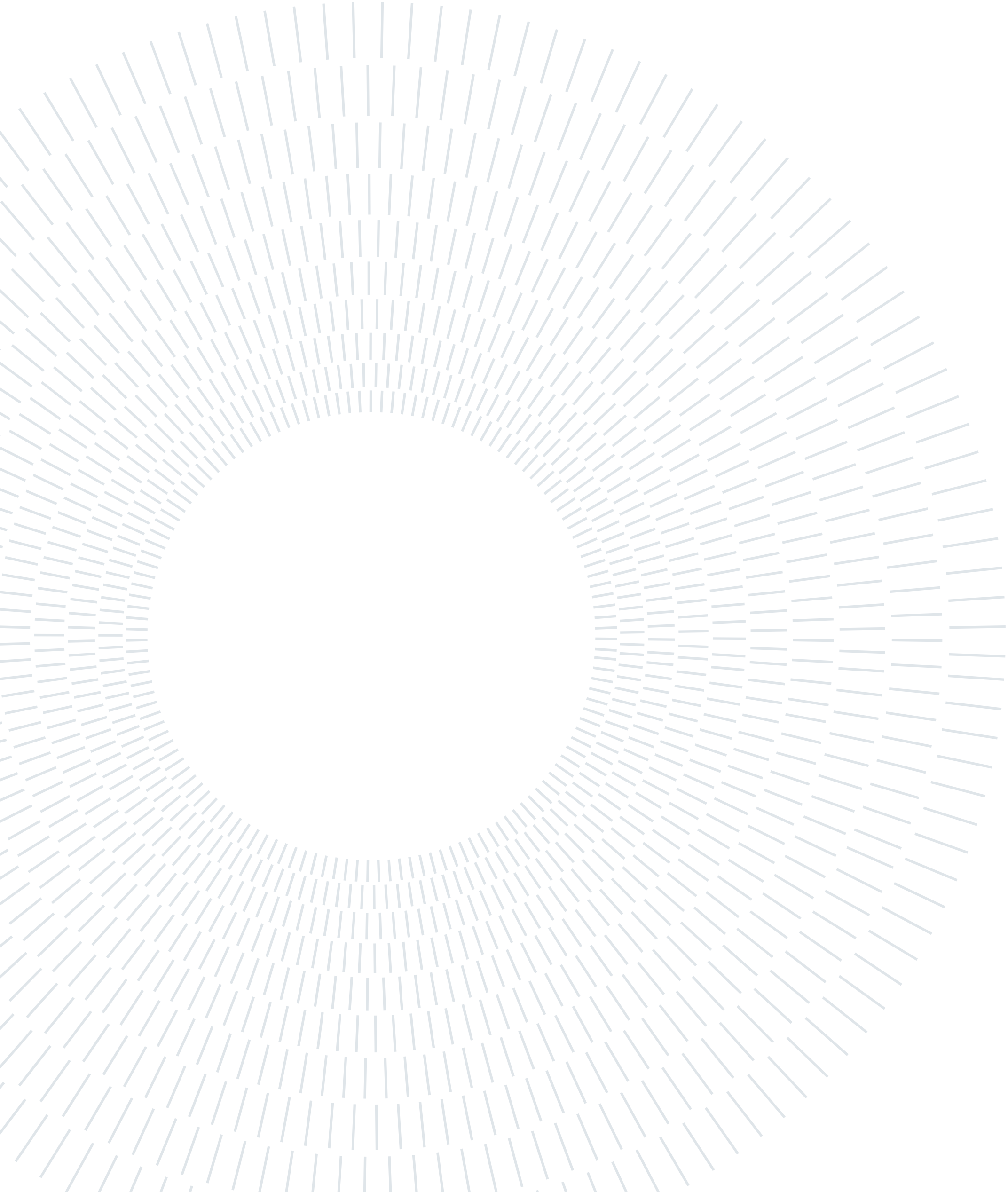
The first thing one notices is that, correctly, the return temperature, which ranges from 55°C to 65°C, is always smaller than the supply one, due to the DHS structure previously mentioned: the heating station provides hot water to the network, meanwhile the users give back cold water.

As expected, the load supply temperatures have a trend similar to the boiler reference. Actually, the farther the consumer is, the bigger the pressure and friction losses are and consequently the temperature profile is more distant from the reference one. For instance, if a simple step is given as reference to the boiler temperature (e.g. from 80 to 90°C, see the beginning of Figure 1.11a), the closest consumer has to wait roughly two hours to get the water settled at the steady-state value (90°C). By contrast, the furthest user must wait almost twice as long to not even get the requested temperature (in the same example T_5^s will settle at 87°C). This should not come as a surprise, as complex thermo-hydraulic systems such as the AROMA network have very slow transients [52], and these results confirm that the model developed in this first chapter resembles pretty well the reality.
Clearly, it could be possible to speed up a little the system dynamics by increasing the mass flow rate (achieved by raising the requested power). However, this specific network operates successfully as long as the power constraints are not violated, as in the daily simulation example, thus the total amount of power that it can be handled is approximately 2 MW.

In conclusion, as visible in Figure 1.11, $\dot{m}_r$ never exceeds 20 kg/s, and the single mass flow rates of the loads are coherent with that value (always smaller). Furthermore, these quantities follow the load profiles evolution: the higher is the power requested by consumers, the greater is the mass flow rate, and hence the faster is the overall system to reach the reference.
Moreover, the boiler power (not shown here for the sake of space) never exceeds its saturation limits (P_{min}^{boiler} and P_{max}^{boiler}), and therefore the network operation is smooth and safe.

1.6. Conclusions

In this chapter it has been proposed a simulation environment where to carry out various experiments on the referenced district heating network named AROMA. In particular, since the state-of-the-art district heating systems are already modelled through classical white-box techniques, a general and concise description regarding the fundamental system components, together with their governing non-linear equations, has been provided. Finally, a daily simulation of the system has been reported so as to clearly show the dynamics of the main variables that will be subsequently identified. Ultimately, the current chapter covered the fundamental physical aspects of the energy system under discussion, necessary to understand the succeeding chapters analysis.



2 | Model Identification

2.1. Chapter Overview

In this chapter it is first convenient to address some basic notions of model identification. Afterwards, two solving approaches are analysed: one exploits polynomial modelling techniques, the other employs neural networks. Finally, a comparison among the various methods is presented, together with a realistic data-based identification example.

2.2. Identification

As stated in the Introduction, for complex large-scale processes with countless phenomena occurring simultaneously, an accurate physics-based dynamic model may be impossible to develop [53]. Consequently, the goal of this chapter is to describe the AROMA network not through highly non-linear and often insufficient physical equations, but thanks to some data-driven identification techniques. In particular, the latter deal with the problem of building mathematical models of dynamical systems based on observed data from the system itself [49].

It is worth recalling that the system under analysis will be fed with some inputs (manipulated signals) plus some external disturbances and it will produce outputs (observable signals), that are exactly what identification aims to get. In conclusion, the procedure to follow is quite straightforward: input and output signals from the system are collected and processed by a data analysis technique in order to infer a model [49].

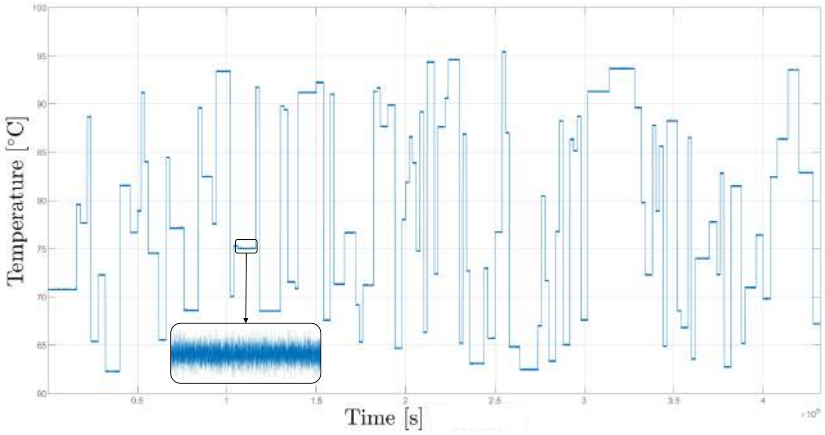
2.2.1. Data

First, it is necessary to establish which system inputs and outputs are of major interest. As introduced in Chapter 1, the AROMA network is currently excited with six inputs: the boiler temperature, i.e. the desired supply temperature value, and the five powers requested by the corresponding consumers.

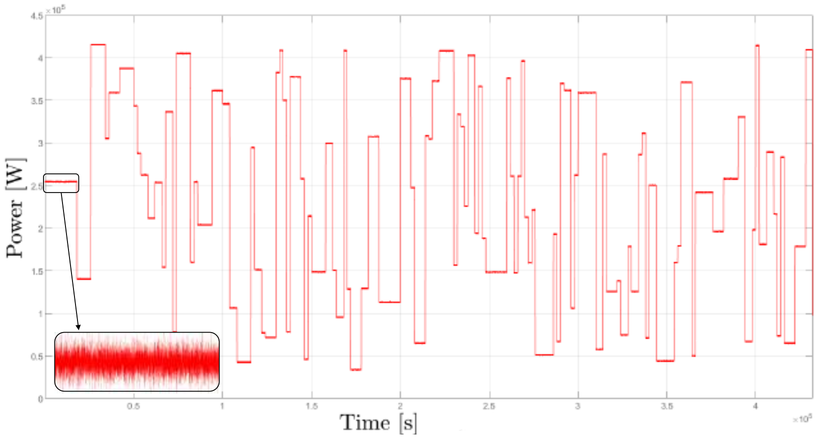
Second, the outputs that must be identified (and subsequently exploited in control schemes) are the supply temperatures of the five users and the overall return mass flow rate and temperature.

The experiment design requires a careful study and selection. In fact, in order to properly excite the system, the inputs must vary within significant ranges while never violating their limits. This is why pseudorandom binary sequences (PRBS) made of steps varying both the amplitude and the interval time size are given as input to the system. In detail, all the signals are randomly generated with some constraints:

- the boiler reference temperature can vary its amplitude from 62°C to 96°C;
- the consumers power can vary its amplitude from 30kW to 420kW;
- each consumers power is additionally divided into five subsequences (see Figure 2.2b), so that it varies not only in the whole magnitude range but also in three other smaller ranges that are useful to better catch the system dynamics. Actually, this is motivated by the typical power daily consumption (see Figure 1.10b), which is characterized by low activity periods (e.g. night-time) and high activity periods (e.g. early morning hours);
- some steps must allow all the variables to reach their steady-state values, and hence the signal frequency is selected such that the slowest settling time (around three hours of T^r) is respected. Other steps instead can have a smaller time range;
- in order to attain a meaningful experiment, a huge number of data is collected, i.e. a 10-day simulation is performed;
- a random White Gaussian Noise (WGN) is superimposed to the so-formed signals in order to mimic real external disturbances and to test the identification methods robustness. Specifically, T_{ref}^{boiler} is characterized by a WGN of peak-to-peak amplitude 0.15°C, whereas P_j^{load} is characterized by a WGN of peak-to-peak amplitude 1kW, see Figure 2.1;
- at first (Chapter 2 and 3), a sampling time of one minute is selected, so that the collected data are maximally informative. The choice is made by considering the fastest variable (T_{out}^{boiler}), which has a settling time of roughly one minute and a half. Subsequently, in Chapter 4 and 5, a sampling time of five minutes is employed because of computational reasons.



(a) Boiler reference temperature with WGN.

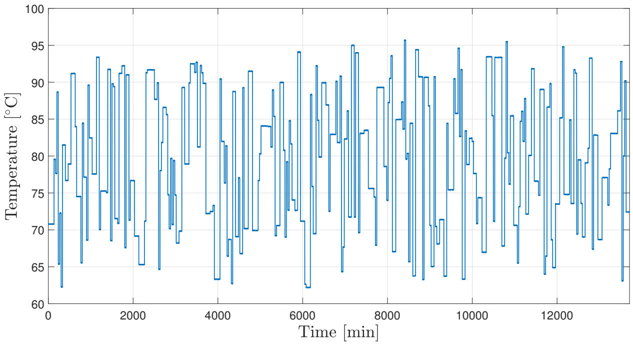


(b) First load profile with WGN.

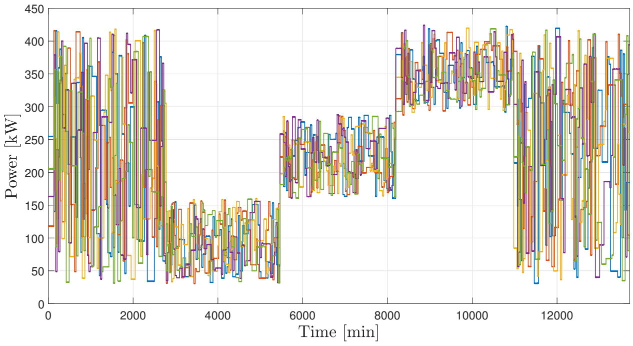
Figure 2.1: Example of noisy inputs.

Finally, in Figure 2.2 it is possible to appreciate the complete 10-day experiment: Figure 2.2a and 2.2b contain the inputs, whereas Figure 2.2c and 2.2d show the outputs.

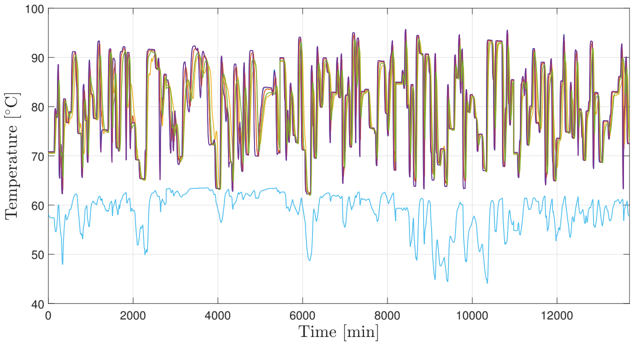
In particular, these data will be used as training (and validation, in the case of RNNs) sets, whereas the dataset displayed in Figure 2.3 will be used for testing. This splitting is crucial for a performing identification: the training set is used to build the model, which is then challenged with the validation set that contains unknown samples, and thus the model accuracy is assessed [92]. Finally, the performance is evaluated on a new, never seen before, dataset, i.e. the test set.



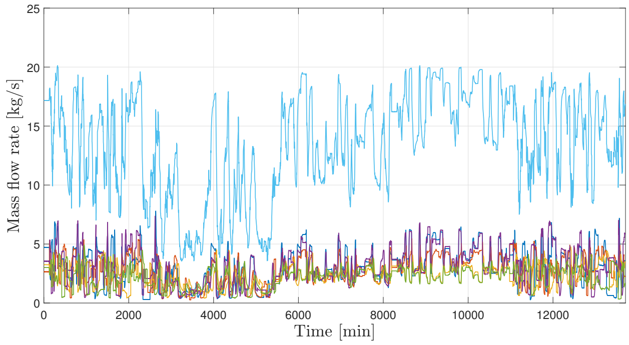
(a) Boiler reference temperature.



(b) Load profiles showing the five subsequences.

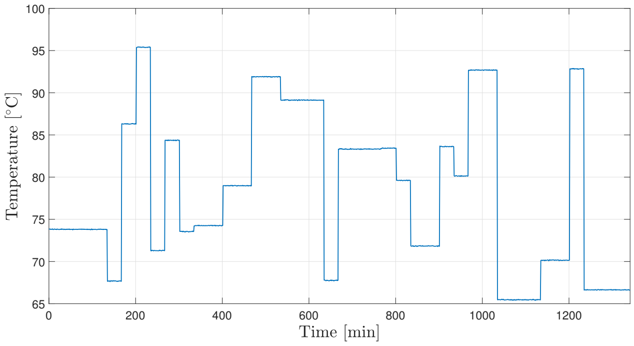


(c) Supply and return temperatures.

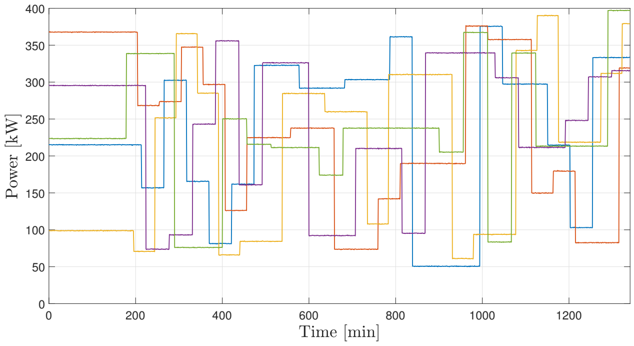


(d) Return mass flow rates.

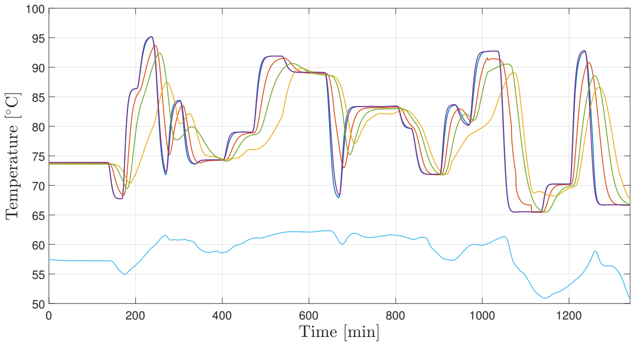
Figure 2.2: Input and output variables of a 10-day simulation, used for *training*: in blue the first user's variables, in purple the second's, in red the third's, in green the fourth's, in yellow the fifth's and in light-blue the return variables.



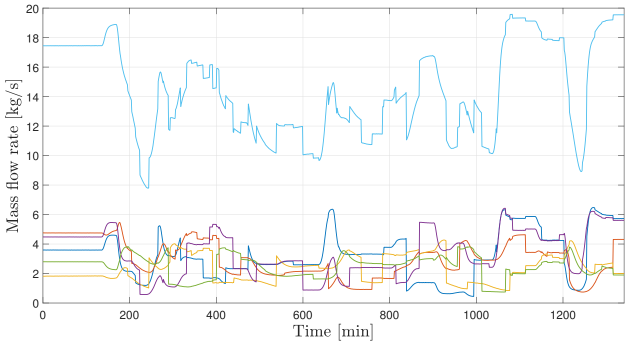
(a) Boiler reference temperature.



(b) Load profiles.



(c) Supply and return temperatures.



(d) Return mass flow rates.

Figure 2.3: Input and output variables of a daily simulation, used for *testing*: in blue the first user's variables, in purple the second's, in red the third's, in green the fourth's, in yellow the fifth's and in light-blue the return variables.

Now few comments on these plots. First, as stated above, the constraints on the inputs are satisfied and the outputs behaviour is coherent with the system dynamics.
Then, the return temperature is the slowest variable in terms of tracking speed, whereas the supply temperatures of the first and second consumer are obviously the fastest (being the closest to the heating station). Besides, the return mass flow rate follows appropriately the load profiles evolution.
Finally, the first data of any simulation are always discarded (not visible in the plots), so that the system is allowed to reach its initial equilibrium.

2.2.2. Candidate Models

In this chapter different candidate models are analysed. In particular, some black-box techniques are implemented: a black-box model of a system does not use any particular prior knowledge of the physical relationships involved, and therefore it is more a question of "curve-fitting" than "modelling" [50].
First, three standard linear models are deepened and compared: State-Space (SS) and polynomial models, i.e. AutoRegressive with eXternal input (ARX) and Output-Error (OE) models.
Second, within deep learning framework, two kinds of Recurrent Neural Networks (RNNs) are explored, namely Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), with the aim of improving the results obtained through the aforementioned classical identification methods.

2.2.3. Assessment Rule

In order to quantitatively evaluate the identification results, a performance index must be computed. In particular, the Normalized Root Mean Square Error (NRMSE or FIT) is determined so as to assess the models overall accuracy, by comparing the actual data with the predicted ones:

$$FIT = \left(1 - \frac{||\mathbf{y}_{real} - \mathbf{y}_{id}||_2}{||\mathbf{y}_{real} - y_{avg}||_2}\right) \cdot 100 \tag{2.1}$$

where, intuitively, $\mathbf{y}_{id}$ is the vector of identified outputs, $\mathbf{y}_{real}$ the vector of the real ones and y_{avg} is its average [10].

In addition, in order to assess each output performance, a suitable index is the so-called coefficient of determination, indicated as R^2 and defined in (2.2).

$$R^2 = \left(1 - \frac{\sum_{i=1}^{T}(y_{i,real} - y_{i,id})^2}{\sum_{i=1}^{T}(y_{i,real} - y_{avg})^2}\right) \cdot 100 \tag{2.2}$$

2.3. Prediction and Simulation

A key distinction in the system identification world must now be addressed so as to select the correct type of model, which must be related to the application is going to be used for [49].
On the one hand, **prediction** means projecting the model response k steps ahead into the future (prediction horizon) using the current and past values of measured inputs and outputs. By calling the output $y(t)$ and the input $u(t)$, a dynamic system described by a first-order differential equation

$$y(t+1) = ay(t) + bu(t) \tag{2.3}$$

can be predicted as

$$y_p(t+1) = ay_m(t) + bu_m(t) \tag{2.4}$$

where, intuitively, subscript p stands for predicted and m stands for measured [1].

On the other hand, **simulation** means computing the model response using input data and initial conditions. In other words, given inputs $u(t_1, .., t_N)$, the simulation generates $\hat{y}(t_1, .., t_N)$, i.e.

$$\hat{y}(t+1) = a\hat{y}(t) + bu(t) \tag{2.5}$$

Actually, the most basic use of a system description is to simulate the system response to various input scenarios [49]. To sum up, the main difference between the two methods is that the one-step predictor depends on past measurements of y and u, i.e. the prediction is based on the actual measured outputs and not on the past predicted outputs, as in simulation. This implies that, in cases like ARX (see Section 2.4.2), a model with good predictive performance is not necessarily a good simulation model [19]. As a result, in order to identify models which reliably work in simulation, Output-Error models (see Section 2.4.2) are needed [19]. Ultimately, for this thesis purposes, a simulation approach is going to be used.

2.4. State-Space and Polynomial Models

In this section, state-space and polynomial models are explored. In particular, the simulation is performed through the MATLAB System Identification Toolbox, which provides MATLAB functions, Simulink blocks, and an app for dynamic system modelling, time-series analysis, and forecasting. It is suitable to learn dynamic relationships among measured variables in order to create transfer functions, process models, and state-space models in either continuous or discrete time while using time- or frequency-domain data [1].

The goal of presenting this type of models is to highlight not only their features and benefits but also some limitations and drawbacks, especially when compared to other techniques (Section 2.5). For instance, one would expect relatively poor performance since this kind of models involves linear structures and hence a complex non-linear system such as the AROMA network is likely to be poorly identified by them.

To learn more about these models, go through the source of this section [49].

2.4.1. State-Space Models

In the state-space form the relationship between the input, noise and output signals is written as a system of first-order differential or difference equations (continuous or discrete time) using the state vector $x(t)$ [49]:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ \dot{y}(t) = Cx(t) + Du(t) \end{cases} \tag{2.6}$$

When performing identification by means of the toolbox, a discrete-time model must be selected in the structure options. Furthermore, the N4SID (Numerical algorithm for Subspace State-Space System Identification) is chosen as estimation method and the focus is set on simulation (not prediction).

Then, it is necessary to pay a particular attention regarding the model structure, i.e. the model order (n) selection, which is not trivial. Actually, this is done with a trial and error approach, having as guiding metrics the FIT value of the identification result and the model zeros-poles plot: sometimes a high order is not necessary to describe the system, in particular if many zeros and poles are almost overlapping. In conclusion, it is advisable to start from a low order and then to increase it until the FIT value starts diminishing.

2.4.2. Polynomial Models

Polynomial models, which include FIR, ARX, ARMAX, ARMA, ARARX, ARARMAX, OE and BJ models, are generally described by the equation reported in (2.7) [49].

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \tag{2.7}$$

where $e(t)$ is the error and A, B, C, D, F are the polynomials describing the system. Specifically, the ARX and OE models will be considered in detail.

2.4.2.1. Autoregressive with External Input Models

The ARX model name stands for autoregressive with external input because it trivially includes an input term. In a nutshell, the ARX structure contains only two polynomials among those embedded in (2.7), A and B, i.e. it is given by the following equation:

$$A(q)y(t) = B(q)u(t - n_k) + e(t) \tag{2.8}$$

where n_k is the number of input samples that occur before the input affects the output (input-output delay), $A(q) = 1 + a_1 q^{-1} + ... + a_{n_a} q^{-n_a}$, $B(q) = b_1 + b_2 q^{-1} + ... + b_{n_b} q^{-n_b+1}$, n_a is the order of polynomial $A(q)$ (number of poles) and n_b is the order of $B(q) + 1$ (number of zeros) [1].

2.4.2.2. Output-Error Models

OE models are a special configuration of polynomial models, having only two active polynomials: B and F. These models represent conventional transfer functions that relate measured inputs to outputs while also including white noise as an additive output disturbance. The system is represented by the following equation:

$$y(t) = \frac{B(q)}{F(q)}u(t - n_k) + e(t) \tag{2.9}$$

where the orders of the polynomials are [1]:

$$n_b: \quad B(q) = b_1 + b_2 q^{-1} + ... + b_{n_b} q^{-n_b+1} \tag{2.10}$$

$$n_f: \quad F(q) = 1 + f_1 q^{-1} + ... + f_{n_f} q^{-n_f} \tag{2.11}$$

Again, for both polynomial models, in the toolbox options it is necessary to set the focus on simulation, the domain on discrete-time and to carefully select the orders of the polynomials, as explained in Section 2.4.1.

2.4.3. Models Comparison

At this point, it is appropriate to go over some identification results.

First, the experiment details are here reminded: the inputs are six, i.e. the boiler reference temperature and the five load profiles, whereas only one output (T_5^s) is considered, so as to start with a simple identification example.
In the end, almost ten days of simulation are given to the toolbox as training data (Figure 2.2), and roughly one day of testing is used to validate the model (the initial data are discarded for transient reasons), see Figure 2.3.

Second, the orders selected for the three models, using the aforementioned choice criterion, are displayed in Table 2.1. Clearly, these orders are the ones that yield the best performance of each model in terms of predictive accuracy. Finally, in Figure 2.4 it is possible to observe the identification results of the fifth consumer's supply temperature, compared to the actual value (black line):

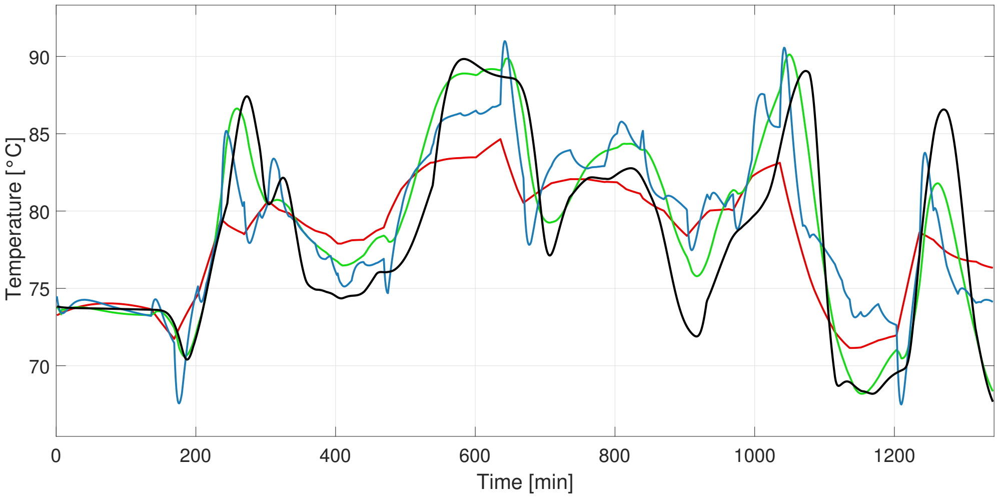


Figure 2.4: Polynomial models best identification results: in black the ground truth, in green the ARX model's result, in blue the SS' and in red the OE's.

From a quick visual inspection, OE seems to have the worst performance, whereas ARX and SS models show pretty similar trends. In order to quantitatively assess these results, it is convenient to compute the FIT, as reported in Table 2.1.

Model	**Order**	**FIT** [%]
SS	$n = 3$	34.6
ARX	$n_a = 9,\ n_b = 9,\ n_k = 1$	60.8
OE	$n_b = 1,\ n_f = 1,\ n_k = 1$	30.4

Table 2.1: Polynomial models FIT comparison.

In conclusion, the best identification performance is obtained through an ARX model, whereas the FIT values confirm the OE to be the poorest. However, generally speaking, all the results are not very satisfactory and this leads us to exploit more complex identification techniques (Section 2.5).

Actually, the variable T_5^s is one of the most critical to identify (together with T^r and $\dot{m}_r$), being the fifth user the furthest: many friction and pressure losses happen in the pipelines before the consumer's position. In fact, the same simulation procedure has been applied to T_1^s too, whose results are definitely better.
Anyway, the choice to show T_5^s results is made on purpose in order to underline the poor performance of these models, especially when compared to the neural networks ones: a reliable identification model must be able to successfully simulate all the involved variables.

2.5. Recurrent Neural Networks

The target of this section is to implement an additional identification method in order to get rid of all the system complicated non-linear equations and to try to achieve an improvement over the models analysed in Section 2.4.
A general outline regarding recurrent neural networks has been provided in the Introduction: these networks are particularly interesting when dealing with identification and control problems, owing to their ability to describe dynamical systems [9, 55]. However, their flexibility comes at the cost of computationally heavy and complex training algorithms [64], which typically suffer of the exploding gradient issues [34].

In general, RNNs are stateful neural networks that can be described as a dynamical state-space model:

$$\begin{cases} x_{k+1} = f(x_k, u_k; \Phi) \\ y_k = g(x_k, u_k; \Phi) \end{cases} \tag{2.12}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector, $u \in \mathbb{R}^{n_u}$ is the input, $y \in \mathbb{R}^{n_y}$ is the output and Φ is the set of parameters (*weights* and *biases*) that are computed during the training procedure [11].

The most advanced RNN architectures include the category of gated RNN, in which the internal loops are regulated by the so-called *gates*, that make them more suitable to learn dynamical systems [11]. To be precise, LSTM and GRU networks belong to this group.

2.5.1. Long Short-Term Memory

Recurrent neural networks with Long Short-Term Memory (LSTM) have emerged as an effective and scalable model for several learning problems related to sequential data [28]. In particular, they have been recently proposed to solve the aforementioned issues (vanishing and exploding gradient problem).
The central idea behind the LSTM architecture is a memory cell, which can maintain its state over time, and non-linear gating units, which regulate the information flow into and out of the cell [28]. This means that given the input at every time step, the LSTM model generates hidden representation/embeddings at every time step, which are then used for prediction. Essentially, the LSTM model defines a transition relationship for the hidden representation through an LSTM cell, which takes the input of features at the current time step and the inherited information from previous time steps [37].

In more detail, it is possible to describe an LSTM network in the following state-space form [9, 11] (single-layer structure):

$$\begin{cases} \chi_{k+1} = f_k \circ \chi_k + i_k \circ \phi(W_c u_k + U_c \xi_k + b_c) \\ \xi_{k+1} = o_k \circ \phi(\chi_{k+1}) \\ y_k = U_y \xi_k + b_y \end{cases} \tag{2.13}$$

where $\chi \in \mathbb{R}^{n_x}$ is the so-called hidden state and $\xi \in \mathbb{R}^{n_x}$ is the cell state.
Then, f_k, i_k and o_k are the forget, input and output gates, respectively, that rule the information flow throughout the network, i.e. they are the elements that allow to prevent the vanishing and exploding gradient problem [34, 64]. Furthermore, the gates can be described by

$$\begin{aligned} f_k &= \sigma(W_f u_k + U_f \xi_k + b_f) \\ i_k &= \sigma(W_i u_k + U_i \xi_k + b_i) \\ o_k &= \sigma(W_o u_k + U_o \xi_k + b_o) \end{aligned} \tag{2.14}$$

where the state vector of the network is $x_k = [\chi_k', \xi_k']$ and the set of weights that must be tuned during the training procedure is $\Phi = \{W_f, U_f, b_f, W_i, U_i, b_i, W_c, U_c, b_c, W_o, U_o, b_o, U_y, b_y\}$,

which is indeed a huge set and hence the training process is not trivial at all. In particular, the weight matrices are $W_f, W_i, W_c, W_o \in \mathbb{R}^{n_x \times n_u}$, $U_f, U_i, U_c, U_o \in \mathbb{R}^{n_x \times n_x}$ and the bias vectors are b_f, b_i, b_c, b_o, of proper dimensions [9].
In addition, the activation functions that introduce non-linearity in the RNN [60] are the sigmoid and the hyperbolic tangent, respectively denoted by $\sigma(x) = \frac{1}{1+e^{-x}}$ and $\phi(x) = \tanh(x)$. Lastly, the Hadamard (element-wise) product between two generic vectors u and v is indicated by $u \circ v$ [11].

2.5.2. Gated Recurrent Unit

The Gated Recurrent Unit (GRU) is a special type of optimized LSTM-based recurrent neural network [71]. The GRU internal unit is similar to the LSTM internal unit [16], except for the fact that the GRU combines the incoming port and the forgetting port in LSTM into a single update port [56].
In particular, it is a fair trade-off between architecture complexity and modelling performance [7]: GRU has a simpler structure with respect to LSTM, thanks to which the number of weights is reduced.
Finally, the state-space model of GRU can be written as [11] (single-layer structure)

$$\begin{cases} x_{k+1} = z_k \circ x_k + (1 - z_k) \circ \phi(W_r u_k + U_r f_k \circ x_k + b_r) \\ y_k = U_o x_k + b_o \end{cases} \tag{2.15}$$

where z_k and f_k are the gates, described by

$$\begin{aligned} z_k &= \sigma(W_z u_k + U_z x_k + b_z) \\ f_k &= \sigma(W_f u_k + U_f x_k + b_f) \end{aligned} \tag{2.16}$$

Again, the weights and biases of the network are $\Phi = \{W_r, U_r, b_r, W_z, U_z, b_z, W_f, U_f, b_f, U_o, b_o\}$.
In a nutshell, during model fitting the weights and biases are updated to minimize the model and data mismatch [53].

2.5.3. Python Implementation

Once the models are defined, firstly an architecture and its hyperparameters must be chosen, then one can proceed to apply the selected RNN to the available data.
It should be reminded that the dataset shown in Figure 2.2 is used for training (the first 13300 observations) and for validation (the last 410 observations), whereas the dataset shown in Figure 2.3 is used for testing (daily simulation with a total of 1340 observations).

In particular, the neural network will be characterized by six inputs (boiler temperature and load profiles) plus seven outputs (five load supply temperatures, return temperature and mass flow rate).

Additionally, as in Section 2.4, the RNN is fed with the measured inputs only, so that the identification is performed in simulation and not in prediction. Actually, the output measurements are given to the algorithm simply in order to compute the loss function during training.

Moreover, the implementation of the various networks is performed with the Python programming language (version 3.10), using the library developed in [8]. All computations are done on an Intel Core i7-1195G7 processor.

2.5.3.1. Hyperparameters

A crucial task for any deep learning network is to determine the so-called hyperparameters. Specifically, RNNs require very few hyperparameters to choose, such as the number of hidden layers, the number of nodes (or neurons) in each layer and the optimizer employed in the training process. The choice of these parameters is truly challenging due to the vast search space and unknown model behaviours: a manually tuned RNN model for a sequential dataset does not perform well for a different dataset [79].

In order to better understand these parameters, it is useful to visualize them graphically, as shown in Figure 2.5.

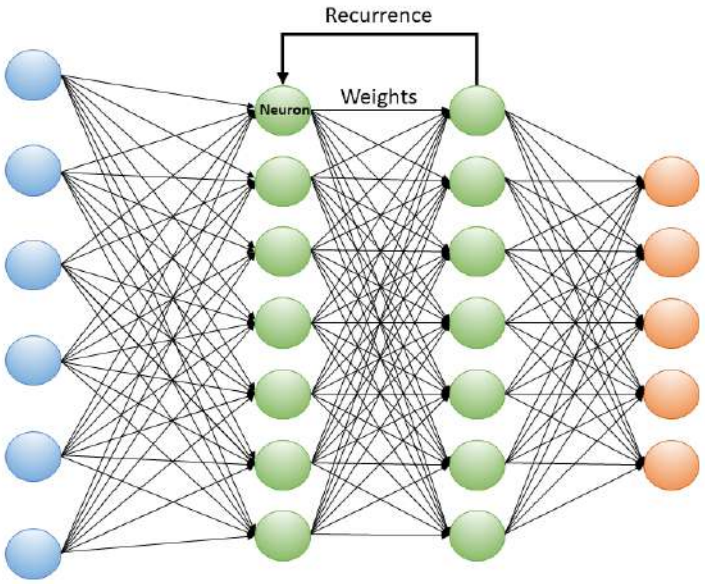


Figure 2.5: A schematic representation of an illustrative RNN. The input layer (6 inputs) is depicted in light-blue, the two hidden layers (7 neurons each) are displayed in green and the output layer (5 outputs) is depicted in orange.

First, an **hidden layer** is basically an intermediate layer between the input and the output layer of the network, and it is the collection of small neurons which transfer the data to layers [40, 84].
The choice of the number of hidden layers is critical, since under-fitting or over-fitting may occur and hence the efficiency, time complexity and overall performance of the network may be affected [84]. Specifically, over-fitting happens when a learning algorithm fits the training dataset so well that noise and data peculiarities are memorized. By contrast, under-fitting occurs when the model is incapable of capturing the variability of the data [36].
To avoid these phenomena, as reported in Table 2.2, three GRU having respectively one, two and three hidden layers are benchmarked.

Second, as far as the number of **nodes** is concerned, the tuning is made with a trial and error approach, by always considering the state dimensionality [11]: being the outputs seven, one network having 14 neurons per layer (intuitively 2 neurons per output) and one having 70 neurons per layer (10 neurons per output) are compared (see Table 2.2).

Finally, regarding the **optimizer**, many gradient-based algorithms are available to solve the training problem. In practice, when the mean square error (MSE) on the validation set stops improving, the training is stopped [11].
In particular, in this thesis two algorithms are considered:

- Root Mean Square Propagation (RMSProp), which has an excellent performance in non-line and non-stationary settings [14] and converges with a proper choice of the hyperparameters, under certain conditions [75];
- ADAptive Moment estimation (ADAM), an efficient stochastic gradient descent-like optimizer that only requires first-order gradients, little memory and which combines the advantages of AdaGrad and RMSprop, even through an easy implementation [14].

2.5.3.2. Training Method

The approach exploited for the RNN training is the so-called Truncated Back-Propagation Through Time (TBPTT) method [7], which consists in extracting shorter and partially-overlapping input-output subsequences, whose length is T_s, from the input-output data of the experiment (indicated as $(\mathbf{u}^{\{i\}}, \mathbf{y}_m^{\{i\}})$).
In practice, the network is trained by minimizing the MSE (loss function $\mathcal{L}$) between the

RNN prediction and measured output [11]:

$$\min_{\Phi}\{\mathcal{L}(\Phi) = MSE(\mathcal{I}_t; \Phi)\} \tag{2.17}$$

computing the MSE as

$$MSE(\mathcal{I}_\alpha; \Phi) = \frac{1}{|\mathcal{I}_\alpha|(T_s - T_w)} \sum_{i \in \mathcal{I}_\alpha} \sum_{k=T_w}^{T_s} (y_k(x_0, \mathbf{u}^{\{i\}}; \Phi) - y_{m,k}^{\{i\}})^2 \tag{2.18}$$

where $\mathcal{I}_\alpha$ denotes the number of subsequences in the training ($\mathcal{I}_t$), validation ($\mathcal{I}_v$) or test set ($\mathcal{I}_i$), $y_k(x_0, \mathbf{u}^{\{i\}}; \Phi)$ is the RNN output initialized in the random state x_0 and fed by the input sequence $\mathbf{u}^{\{i\}}$, and T_w is the washout period, i.e. in our example the first thirty steps are discarded because of the random initialization [11].

Another fundamental expedient to get great results, after the division of the dataset into training, validation and test sets, is to normalize the quantities with respect to their mean and standard deviation. Actually, inputs and outputs have different units of measurement and orders of magnitude, and hence it is convenient to have a standardized distribution, with zero mean and standard deviation one.

Finally, the network is trained over a period of 1500 epochs, even though the loss function usually stops decreasing much earlier (early stopping), with an optimization learning rate of 0.003, so as to get a good trade-off between convergence speed and excessive oscillations avoidance [5].

2.5.3.3. Results Comparison

In order to find the best combination of network, optimizer, number of hidden layers and of neurons, many tests are carried out. In Table 2.2 it is possible to appreciate such combinations, along with the training time required to reach the best epoch (BE), i.e. the epoch in which the FIT is at its maximum, and the performance indexes. These last quantities, in detail, are the overall FIT (computed by taking into account all the identified outputs) and the minimum and maximum R^2 values, i.e. associated to the worst and to the best predicted variable.

RNN	Optimizer	Layers	Neurons	FIT[%]	R^2_{min}[%]	R^2_{max}[%]	BE	Time
GRU	ADAM	1	14	73.8	65.2	98.4	233	5’
GRU	ADAM	2	14	82.1	88.8	98.9	296	10’
GRU	ADAM	3	14	80.6	84.3	98.8	1496	1h2’
GRU	ADAM	2	70	76.9	78.1	97.6	689	33’
GRU	RMSProp	2	14	76.2	81.7	97.6	125	3’
LSTM	ADAM	2	14	78.8	86.4	98.6	659	17’

Table 2.2: Performance comparison among different models of RNN.

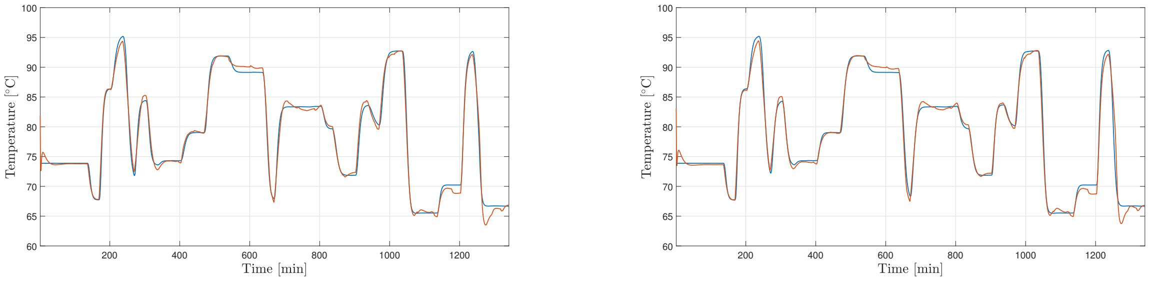
By observing the previous table, it is possible to draw some conclusions.

First, a GRU optimized with ADAM and having 14 neurons per layer, yields the best simulation performance when the hidden layers are two: in fact, both with one and with three hidden layers the FIT is smaller than with two of them. This is due to an under-fitting (1 layer) and over-fitting (3 layers) phenomenon.

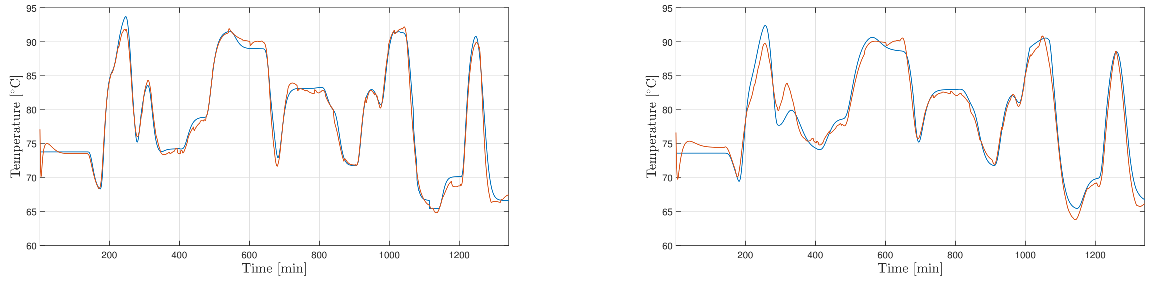
Second, if the number of nodes is increased from 14 to 70, the FIT slightly drops because of over-fitting. Furthermore, the higher is the number of weights to determine, the longer is, clearly, the training time (see the network with three hidden layers or the one with 70 neurons per layer), which is, however, still reasonable.
All these considerations lead to consider the GRU network containing 2 hidden layers, with 14 neurons each and optimized through the ADAM algorithm the best analysed so far.

Third, by having the just mentioned network as benchmark, it is possible to compare it with others slightly modified. As expected, the GRU performs better when the optimizer is ADAM, if compared to RMSProp, even though the latter achieves a slightly faster procedure. In addition, by keeping again all the same hyperparameters, but this time exploiting the LSTM network, it is trivial to conclude that the latter is worse both in terms of FIT and of time to reach the best epoch.

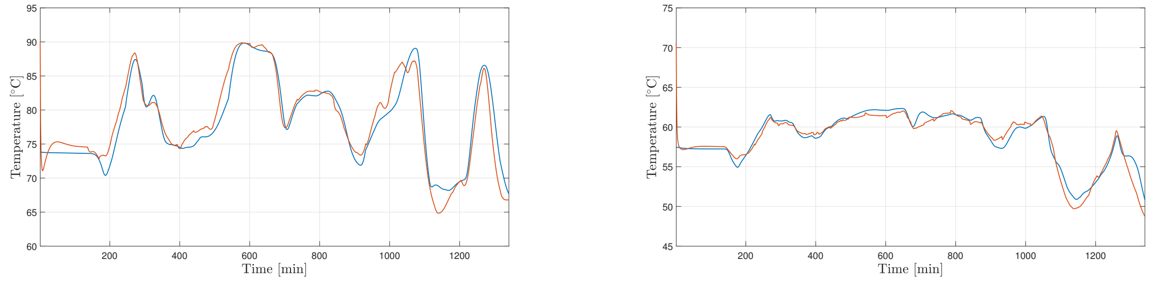
In conclusion, from now on, the GRU network highlighted in Table 2.2 will be used to identify the outputs of the AROMA DHN. In Figure 2.6 the identification performed by such a network on the test set is finally shown.



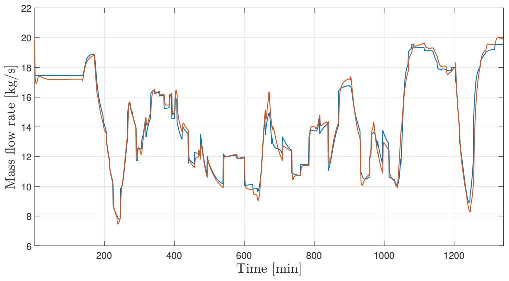
(a) First load supply temperature (b) Second load supply temperature



(c) Third load supply temperature (d) Fourth load supply temperature



(e) Fifth load supply temperature (f) Return temperature



(g) Return mass flow rate

Figure 2.6: Variables identified by the best GRU (red) compared to the real ones (blue).

As visible from the previous plots, generally speaking, the identified outputs are clearly very close to the measured ones, in some cases almost overlapping.
However, the prediction is more accurate when the variables identified are closer to the

heating station (e.g. T_1^s and T_2^s), and in fact their R^2 value is always higher than the one of farther quantities (e.g. T^r or T_5^s, whose R^2 typically coincides with the minimum). Indeed, this performance index is needed to spot where are the major identification issues in order to locally intervene.
Besides, the identification is even robust with respect to the WGN added intentionally as exogenous disturbance to the inputs, since the algorithm does not fit and memorize the noise. Overall, the GRU performance is definitely powerful and effective, even if, it goes without saying, there is still room for improvement.

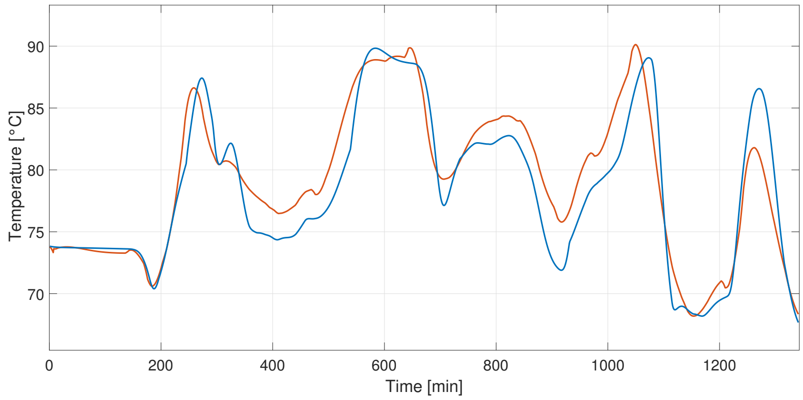
2.5.4. ARX and GRU Comparison

Now that both polynomial models and recurrent neural networks have been implemented and investigated, a comparison between the finest of the two categories is here reported. Specifically, to be fair, the identification is carried out on a MISO (Multiple Input Single Output) system: the usual six inputs and the fifth load supply temperature as output. Moreover, the GRU is optimized through the ADAM algorithm and it is made of two hidden layers having each 5 nodes (since only one output is considered).

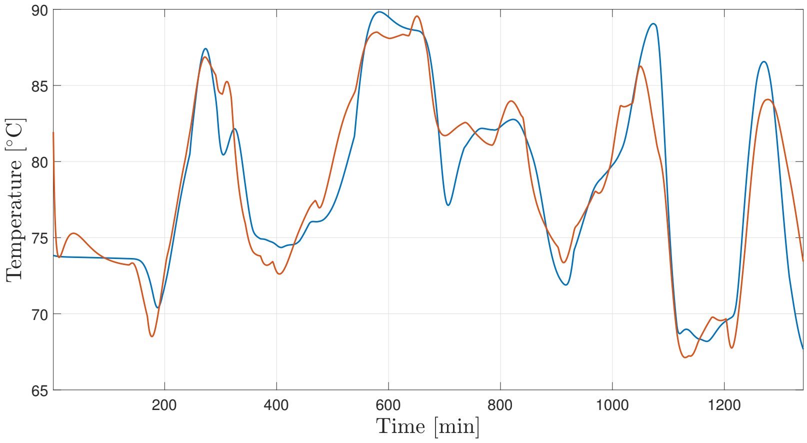
In particular, in Figure 2.7 the identified T_5^s is compared with its measured value, whereas in Table 2.3 the identification FIT values of the ARX model and the GRU network are placed side by side. Needless to say, the GRU network outperforms the ARX model. On the other hand, this comes at a cost. Great performance and high accuracy result in a complex and computationally expensive training procedure. For instance, the outcome obtained through the ARX model is almost instantaneous, the one achieved through the RNN, instead, requires approximately one hour of training (1500 epochs). In addition, the ARX identification is pretty straightforward, also thanks to the toolbox ease of use, whereas the GRU identification takes time and efforts for the programming part. However, since the system under control is characterized by a non-linear behaviour, it is evident that standard linear model structures, such as ARX and OE models, are not appropriate [11, 74]. Ultimately, for control purposes it is better to have a precise and effective simulation tool, and therefore the GRU network is the most suitable for that target.

Model	**FIT [%]**
ARX	60.8
GRU	68.8

Table 2.3: ARX and GRU FIT comparison.



(a) Fifth load supply temperature identified by ARX (red) compared to the real one (blue).



(b) Fifth load supply temperature identified by GRU (red) compared to the real one (blue).

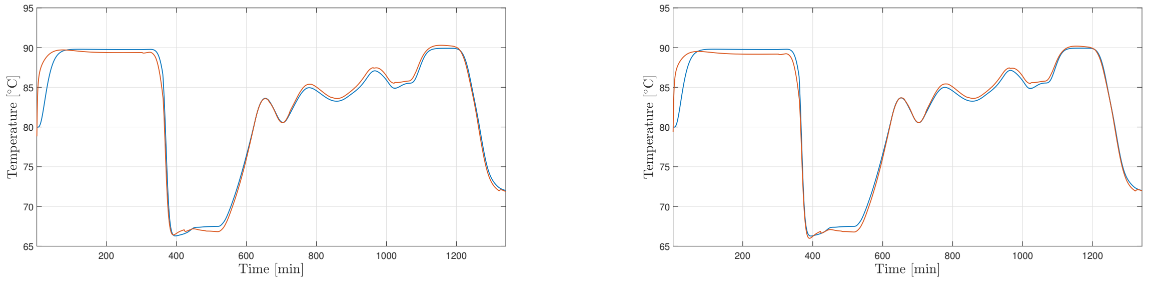
Figure 2.7: ARX and GRU results comparison.

2.5.5. Identification of a Daily DHS Operation

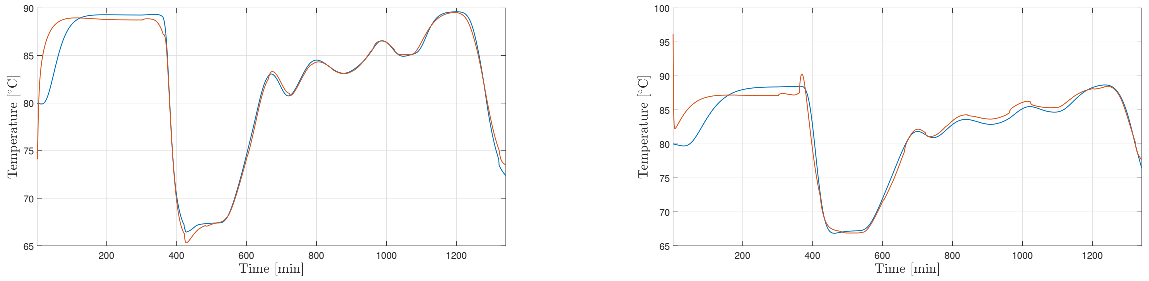
Up to now, the identification methods have only been tested on some PRBS signals specifically built to successfully excite the system. At this stage, however, it is good to use the selected technique (GRU) to identify the usual seven outputs in a realistic case. Basically, once the network is trained with the 10-day exciting signals (data used as training and validation set), it is possible to exploit it in order to predict some realistic trends. In practice, the inputs shown in Figure 1.10 and the outputs in Figure 1.11 are given as test set (typical DHS working day) to the GRU.

In Figure 2.8 the simulation results can be appreciated. Once again, the performance is quite satisfactory: as a matter of fact, the FIT is 83.1%, R^2_{min} is 82.2% (corresponding to

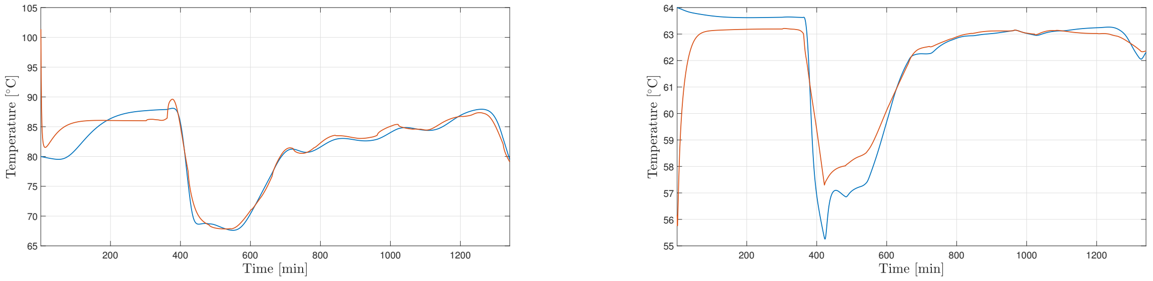
T^r) and R^2_{max} corresponds to T^s_2 with a value of 98.2%.



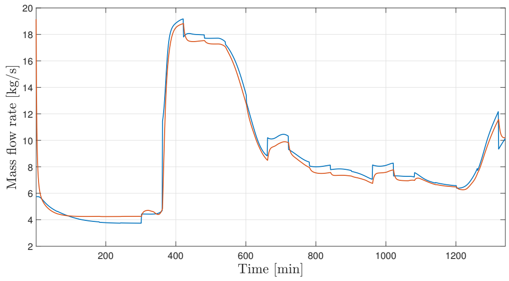
(a) First load supply temperature (b) Second load supply temperature



(c) Third load supply temperature (d) Fourth load supply temperature



(e) Fifth load supply temperature (f) Return temperature



(g) Return mass flow rate

Figure 2.8: Daily simulation: variables identified by a GRU (red) compared to the real ones (blue).

2.6. Conclusions

In this chapter we presented different types of black-box techniques used to identify the AROMA network model. Actually, proving how performing and efficient could be some machine learning techniques was the very first challenge of the thesis. Indeed, the vast majority of modelling methods employed in literature to describe district heating networks is based on physical principles and not on data. In particular, it has been pointed out how recurrent neural networks outperformed state-space and polynomial models, thanks to their recursive structure fully capable of describing the system non-linearities. Moreover, among the possible categories of RNNs, GRU networks turned out to be slightly better than LSTMs, concerning training time and predictive accuracy, mainly because of their simpler architecture. In conclusion, we proved how neural networks are perfectly suitable to describe a DHN model, particularly because they allow to avoid the use of complicated and often insufficient first principles non-linear equations. However, there is definitely still room for improvement.

3 | Physics-based Recurrent Neural Networks

3.1. Chapter Overview

This chapter seeks to underline the main drawbacks of traditional data-driven and model-based methods, and accordingly to point out the advantages of a novel approach, namely physics-based machine learning. After a short literature review on the topic, two physics-based approaches are discussed and implemented, one "soft" and the other "hard-code". In particular, the latter, which ends up to be the most effective and innovative, is thoroughly deepened and compared to standard identification techniques. Finally, a sensitivity analysis on the proposed physics-based neural network approach is carried out.

3.2. Physics-based Machine Learning

When developing predictive models for control purposes, one has to balance model accuracy, complexity and robustness [21]. In general, machine learning-only or physics-only approaches may not be sufficient for knowledge discovery in complex scientific and engineering applications [37]. This is the reason why in this chapter a novel method to identify the variables of a district heating network is proposed.

3.2.1. Motivations and Objectives

In Chapter 2 some traditional black-box models have been analysed, so as to identify the main variables of the AROMA network. Sometimes, however, using this type of algorithms is not the best solution owing to four main problems. First, state-of-the-art black-box machine learning methods require a large supervision in the form of labelled data that is not always available in scientific problems. In addition, these data often cover only a confined spectrum of the actual data distribution. Third, in some cases, black-box models produce inconsistent solutions with physics-based knowledge. Finally,

these methods are unable to discover novel scientific insights from data, because of the very nature of their design [39].
In conclusion, black-box models suffer from poor interpretability being not explicitly designed to represent physical relationships and to provide mechanistic insights.

On the other hand, model-based methods, whose solution structure is deeply rooted in the system scientific knowledge, may not exhaustively capture the dynamics underlying the process, being often incomplete or imperfect [39]. In particular, for complex large-scale systems such as district heating, obtaining the overall model from first principles is a time-consuming and impractical task [21].

In response to these drawbacks, in the scientific community there is a growing interest to embed scientific knowledge in machine learning frameworks, in order to produce physically consistent solutions despite the data paucity. These novel models, which aim at combining physical knowledge with data-driven methods, are referred to as "Physics-based" Machine Learning (PB-ML), as well as "Physics-informed" (PI-ML), "knowledge-guided", "physics-guided" or "theory-guided data science" [39].

Additionally, three fundamental objectives motivate PB-ML research. First, given observations of X and Y as training data, the improvement of predictive accuracy in estimating the output variables over classical data-driven methods and model-based techniques. Second, the goal is to increase the computational efficiency too. Third, PB-ML aims at attaining the capability to learn new scientific knowledge from data. To sum up, hybrid-physics-data learning uses scientific knowledge as an additional source of supervision to learn generalizable ML models, both optimizing model accuracy (measured on the labelled training set) and scientific consistency (evaluated even on out-of-sample distributions) [39].

3.2.2. Categorization of PB-ML

In literature, mainly four strategies are adopted to combine data-driven and science-based learning methods. The categorization and analysis of recent research on PB-ML are exhaustively discussed in [39], which is a book intended to bring together the existing works regarding knowledge-guided machine learning happening in diverse scientific communities.

First, **scientific knowledge-guided learning** is a "soft" way to incorporate scientific knowledge in ML frameworks, since additional loss functions are included in the learning objectives of neural networks, and hence the training procedure is merely modified, without an alteration in the ML architecture. Specifically, loss functions measure the inconsistency of ML solutions relative to scientific equations and laws [39].

An example can be appreciated in [38], where, in order to predict lake temperature profiles, the energy conservation law is combined in the loss function ($\mathcal{L}$) with the training objective of standard LSTM models. In addition, the authors include in $\mathcal{L}$ an extra penalty for violation of density-depth relationship, by making use of a ReLU (Rectified Linear Unit) function. Jia et al. noted that not only the predictive accuracy improved, but also it was possible to use a smaller amount of observed data (e.g. 2% of original data) because the energy conservation law and density-depth constraint regularized the model to retain physical consistency.
A similar approach is adopted in [21], where the authors add constraints in the loss function so that the model variables remain within physically realistic bounds.
Finally, in [60] Physics-based Neural Networks (PB-NNs) recover the solution to the partial differential equations (PDEs) by simply incorporating loss functions based on the PDEs, initial conditions and boundary condition residual errors and by constructing two separate networks, one for each output variable.

Second, **scientific knowledge-guided architecture** methods "hard-code" physical knowledge directly in the solution structure of ML models. For instance, they are used to encode known forms of invariances and symmetries in the system predicted outputs or to hard-code the knowledge of relationships among physical variables in the connectivity patterns of nodes in a NN architecture. Another possibility is to construct knowledge-guided architectures where the sequence of intermediate features extracted at the hidden layers of a NN are informed by scientific knowledge [39].
An example of such a strategy is analysed in [18], where novel physics-based connections are introduced among neurons in the network to capture physics-based relationships of lake temperature. Moreover, in [41] the authors propose a hierarchical deep learning architecture that explicitly models intermediate memory states and fluxes to incorporate the physical relationships among different hydrological processes. Finally, an example in a distinct topic, i.e. quantum mechanics, is provided in [3]. In this article, Anderson et al. set up the network so that each neuron corresponds to a set of physical atoms and each activation is covariant to symmetries. In such a way, the laws that individual neurons learn resemble known physical interactions.
As one can understand, incorporating science knowledge in ML architecture is quite "problem-dependent", since each application requires a different problem set-up.

Third, **scientific knowledge-guided initialization** can also help to achieve better generalization even on unlabelled samples.
In [38], in order to improve the performance given very few observed data, the authors pre-train the PB-NN using the simulated data produced by a simple physics-based model.

Furthermore, in [60] Niaki et al. use the weight matrices and bias vectors of the neural network previously trained as initial values of weights and biases for each similar problem. Similarly, in [41] the authors exploit the predicted value from the previous sample to act as initial value for the next sample.

Lastly, **hybrid-science-machine learning modelling** has the purpose of jointly using science-based and machine learning models to deliver better predictive performance than standard methods alone. Interesting examples of such an approach can be appreciated in [17, 68].

3.3. PB-RNN Implementation for a DHN

Now that scientific literature regarding physics-based machine learning has been explored, it should be clear that, at least to the best of the author's knowledge, there is a deficiency of such a learning strategy in district heating applications. Therefore, this thesis attempts to provide an implementation and analysis of different physics-based RNN techniques and to illustrate their advantages over traditional data-driven models.

Actually, the results obtained in Chapter 2 are satisfactory and do not violate physical laws, but nevertheless the predictive accuracy and the training time could be surely improved by implementing finer learning algorithms. In fact, by observing Figure 2.6, 2.8 and Table 2.2, two main issues may be noticed. First, at the beginning of the prediction, the error is always considerable due to the random initialization. Second, the minimum R^2 values (related to the toughest load variables to identify) are rather low and hence they need to be risen through a more powerful identification strategy.

Finally, in this chapter the usual identification configuration explained in Chapter 2 is exploited, but in this case with some additional outputs, for a more comprehensive identification. In fact, besides the usual six inputs (T_{ref}^{boiler} and P_i^{load}), seventeen outputs must now be identified, i.e. the supply and return temperature of each load, its mass flow rate and the overall return temperature and mass flow rate (see Figure 3.1).

3.3.1. Constrained Loss Function

As introduced in section 3.2.2, a first approach to enhance the identification performance may be to "softly" include some physical laws or constraints, properly weighted, directly in the loss function ($\mathcal{L}$). This is intended to try to raise the predictive accuracy, but, intuitively, not to reduce the training time, since more mathematical operations will be included in $\mathcal{L}$.

As far as DHNs are concerned, a simple constraint regarding load temperatures must be always verified, i.e. at any instant of time and in every load the supply temperature must be greater than the return one (3.1).

$$T_i^s(t) > T_i^r(t), \quad \forall t \in T, \quad \forall i \in \{1,..,n_{load}\} \tag{3.1}$$

This inequality can be rewritten via a slack variable, namely s_i, to indicate the value by which each constraint is violated (3.2).

$$T_i^s = T_i^r - s_i \quad \rightarrow \quad s_i = \max(0, T_i^r - T_i^s) \tag{3.2}$$

In detail, if $(T_i^r - T_i^s)$ is negative, then no penalty must be added in the loss function and indeed $s_i = 0$. By contrast, if $(T_i^r - T_i^s)$ is positive, T_i^r is exceeding the bound, resulting in a non-zero value of the slack variable. In particular, this type of constraint can be straightforwardly implemented by means of a ReLU function and included as auxiliary weighted term in $\mathcal{L}$ [21].

Specifically, the following loss function augmented with penalty terms is optimized to train the GRU:

$$\mathcal{L}_{constrained} = \mathcal{L}_{MSE} + \frac{1}{N}\sum_{i=1}^{n_{load}} \lambda_i ||s_i||_2^2 \tag{3.3}$$

where the first term computes the MSE between observed and predicted outputs over N time steps, whereas the second term penalizes violations of the aforementioned inequality constraints [21]. Finally, λ_i is the weight associated to the i^{th} slack variable, carefully selected through a trial and error approach which led to choose a value of 0.002 for all of them.

At this point, by replacing the traditional loss function with the new one, a results comparison can be made. In detail, both GRU networks are implemented with six layers and fifteen neurons each hidden layer, for a reason that will be clarified in section 3.3.2.

The so-formed RNN could be schematized as represented in Figure 3.1.

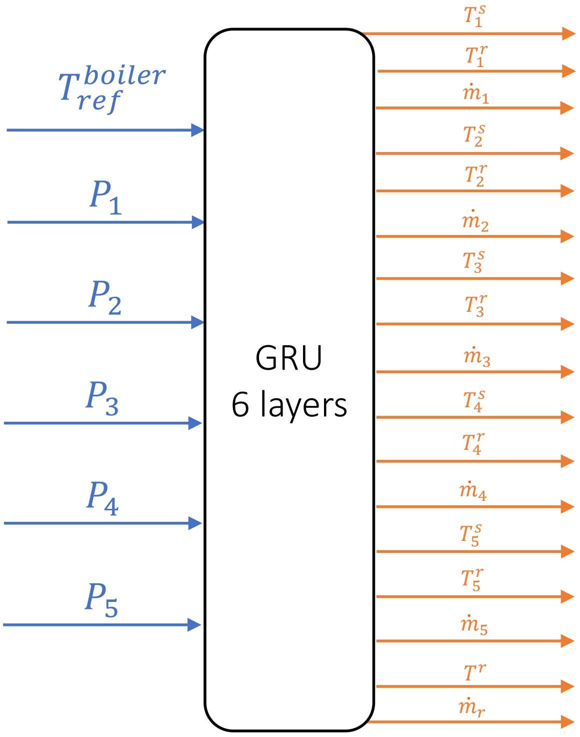


Figure 3.1: A schematic representation of the GRU implemented both with the traditional loss function and with the constrained one. Inputs are highlighted in light-blue, outputs in orange.

Finally, in Table 3.1 a comparison between the standard GRU and the one optimized with the constrained loss trained for 1500 epochs is reported. In particular, one can appreciate the FIT (2.1), the minimum and maximum R^2 value (2.2), together with the best epoch (BE) and the training time to reach it, i.e. the time to reach the FIT peak.

Model	**FIT**[%]	$\boldsymbol{R^2_{min}}$[%]	$\boldsymbol{R^2_{max}}$[%]	**Best Epoch**	**Time BE**
Standard GRU	72.6	69.7	97.5	1495	2h29’
GRU with $\mathcal{L}_{constrained}$	70.3	69.1	97.1	1333	6h

Table 3.1: Performance of a GRU with constrained loss function compared to a traditional one.

The results provided in the previous table explicitly show that there is not a performance improvement when the GRU is trained by minimizing a constrained loss function, in the case of the AROMA network. Also, the training time required to reach the highest value of FIT is dramatically high and pretty unacceptable. This is due to the additional operations that must be performed at each epoch to compute the 2-norm of the five slack

variables vectors.
In conclusion, this new physics-based method is not suitable to fulfil the previously mentioned objectives.

3.3.2. A Novel RNN Architecture

A district heating network is a complex thermo-hydraulic system characterized by a specific modular and sequential topology, as explained in Chapter 1. Such consideration leads us to think that the most effective way to include scientific knowledge in the ML algorithm is not by simply adding some physical equations in the loss function, but rather by drastically modifying the RNN architecture, so as to make it resemble the DHN one.

In practice, it is intuitive that preceding consumers influence subsequent ones and therefore their corresponding variables. Thus, the novel idea proposed by this thesis is to model each load (or loads cluster) as an RNN with a single hidden layer, so that its predicted output variables are given as inputs to the subsequent consumers (again each one modelled as an RNN) that are directly affected by the former. A schematic representation of how the i^{th} load of the AROMA network is encoded as a single GRU model is reported in Figure 3.2.

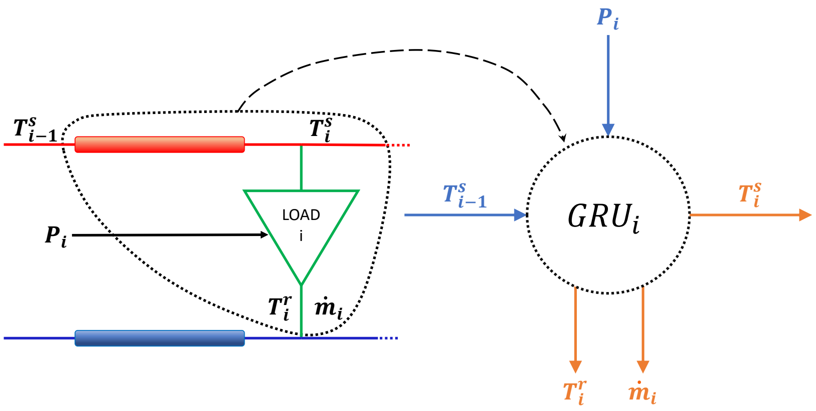


Figure 3.2: An intuitive representation of how the i^{th} load and variables of the physical system are turned into a GRU model. Its inputs are depicted in light-blue, the outputs in orange. The user is represented in green, the forward pipe in red and the return line in blue.

Specifically, each forward pipe connected to the i^{th} user and the consumer itself are hard-coded as a GRU network whose input is, besides power, the supply temperature(s) of the preceding load(s). Additionally, the outputs to be identified are its own supply temperature, return temperature and mass flow rate. Actually, the input choice is not trivial, as different and subtle variables influence the i^{th} outputs behaviour (see Section 3.3.2.1). Moreover, another assumption that must be experimentally confirmed is the possibility to enclose all the return pipelines in a unique GRU, with appropriate inputs.

3.3.2.1. Input Choice

As anticipated, a thoughtful selection of which inputs must be fed to the i^{th} GRU network is crucial to reach outstanding identification results. Another fundamental aspect to take into account is the scalability of the physics-based approach. In fact, by encoding each consumer as an RNN, it is evident that whenever the number of loads increases the number of RNNs, and hence of layers and neurons, explodes. As a result, it is essential to pick a reasonable amount of inputs, and thus of nodes per each hidden layer. Actually, the purpose of this work is to provide a physics-based identification method for the case study under discussion, but even scalable if applied to other district heating systems.

On the one hand, regarding **temperature**, it is of primary importance figuring out which T_i^s influences which load. Actually, as discussed in Chapter 2, when dealing with wide thermal systems the main identification problem in terms of accuracy is related to the furthest consumers, which is indeed intuitive, being the RNN simply fed with the overall supply temperature. Thereby, by breaking down the neural network into physics-related modules, each i^{th} GRU can identify its own T_i^s, which will be then given as input to the subsequent GRUs that are directly influenced by the i^{th} one. In such a way, each GRU is fed with the supply temperatures of the closest loads, and, hopefully, its identification performance improves. In order to understand how the consumers of the AROMA network affect each other, we may recall the forward-part scheme:

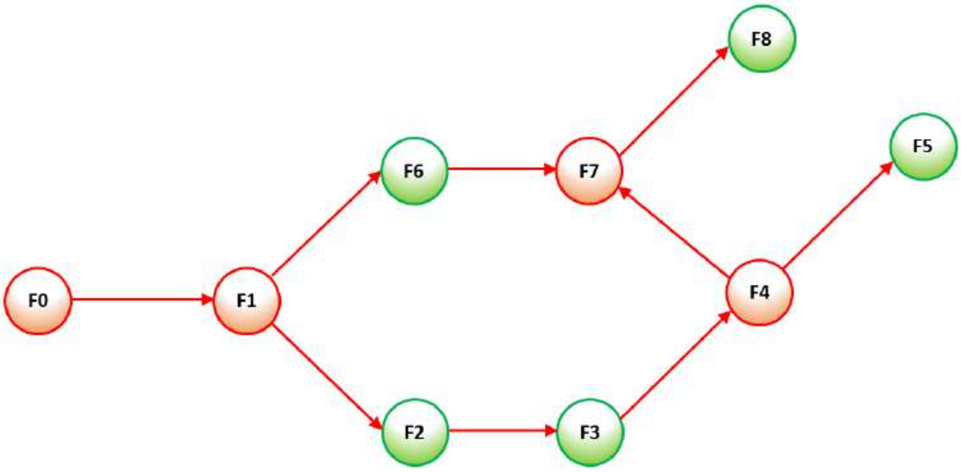


Figure 3.3: AROMA network scheme representing the forward-flow part: arcs are plotted in red, users in green and nodes are referred to as Fi.

As visible from Figure 3.3, node F2 impacts on F3 only, whereas nodes F3 and F6, being located before a ramification and a conjunction, respectively, are not that trivial. In practice, even though the graph is oriented, it has been empirically detected that in some arcs the flow direction may change according to the power demand. For instance, if the consumer positioned in node F5 requires a much larger amount of power than the user placed in node F8, then the arc between node F4 and F7 will have an opposite direction with respect to the usual one (depicted in Figure 3.3). As a result, not only F3 impacts on F5 and F8, but we shall assume that F6 influences F5 and F8 too. Ultimately, F5 and F8, being the final nodes (or *well* nodes) do not affect any other user. To sum up, it is possible to graphically view what just explained for a better grasp:

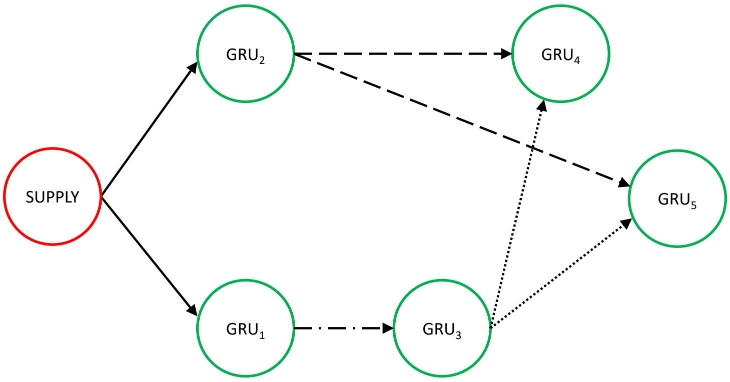


Figure 3.4: Scheme representing the impact of each load on the others in the AROMA network, in terms of supply temperature. Users (GRUs) are depicted in green, supply in red. Each GRU connection has a different line style to better visualize the dependence.

Figure 3.4 clearly shows that the first two GRUs, corresponding respectively to nodes F2 and F6, receive as input the supply temperature (T_{ref}^{boiler}). Besides, the third GRU (node F3) gets T_1^s only, whereas both the fourth and the fifth GRUs (node F8 and F5) are fed with T_2^s and T_3^s.
One final remark which is of crucial importance: the AROMA system is a *meshed* network, characterized by conjunctions and ramifications. The networks represented by this kind of topology must be carefully examined in order to learn how the water flow behaves according to different power configurations, which is usually not self-evident. By contrast, *radial* networks are typically easier to understand being characterized by ramifications only: their shape indeed is similar to a tree and each previous node affects the subsequent ones simply according to the arcs direction.

On the other hand, as far as **power** is concerned, at first glance it would be intuitive providing to the i^{th} load its associated consumption. In such a way, each GRU would have only one additional input and the approach would be easily scalable in the case of much more users. To gain a better understanding of how such an RNN is implemented, a schematic representation of the i^{th} and $(i+1)^{th}$ GRUs is reported in Figure 3.5.

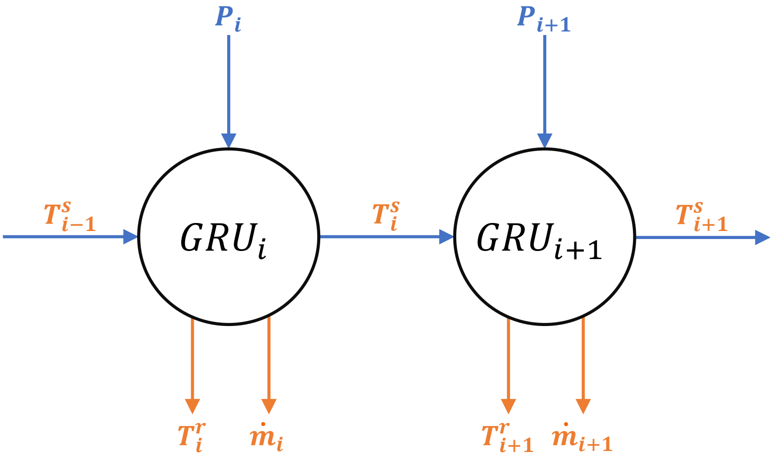


Figure 3.5: Schematic physics-based RNN having as input, besides the supply temperature, the single P_i. Inputs are depicted in light-blue, outputs in orange. Being T^s_i an output for the i^{th} load and an input for the subsequent one, it is represented both in light-blue and in orange.

An issue that clearly stands out by adopting this choice is that the model of each neural network is not considering the influence of the other loads consumption, which is obviously a core information to make the system correctly capture the dynamics of temperatures and mass flow rates. For instance, if the last consumer requires a huge amount of power, then the district heating network must adapt its mass flow rate to satisfy this demand, and hence the other users will receive a smaller volume of water.

As a consequence, a further step that could be investigated is the choice of including all load profiles as input to each GRU. In the case of the AROMA network, this does not represent a major problem, having only 5 consumers, and in fact this strategy was implemented in order to study the improvements over the single-power input RNN. Indeed, the FIT value increased when the modular GRUs were given as input all the five powers. However, actual district heating networks are characterized by far more consumers and it would be unfeasible to feed, for example, 100 GRUs with 100 load profiles each. Thereby, both approaches analysed so far are unsuccessful.

Another possibility is to compute the cumulative power consumption, by summing up all load profiles, in order to give each GRU the information regarding the overall situation of the heating system. This means that the i^{th} GRU can be fed with its own load profile and the sum of the other power consumptions, as schematically reported in figure 3.6. In this way, a DHN having 5 loads or one having 100 loads would be characterized by 5 or 100 GRUs, respectively, each one with 2 inputs associated to power, regardless of the system size.

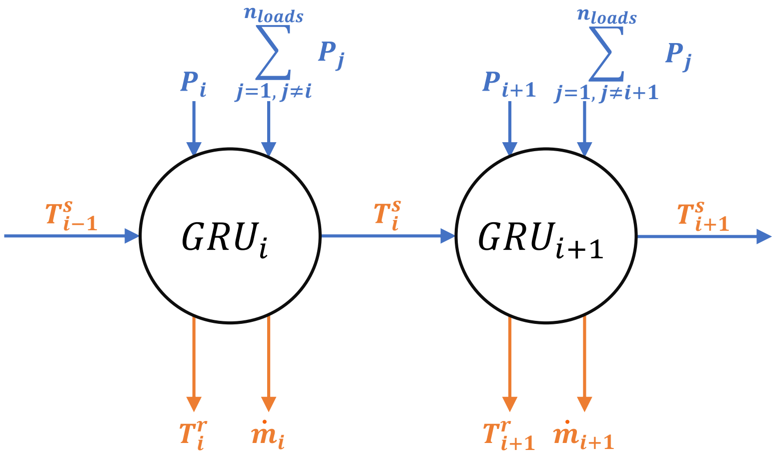


Figure 3.6: Schematic physics-based RNN having as input P_i and the summation of the other load profiles. Inputs are depicted in light-blue, outputs in orange. Being T_i^s an output for the i^{th} load and an input for the subsequent one, it is represented both in light-blue and in orange.

In order to confirm this hypothesis, it is necessary to carry out a simulation and a comparison between the approaches. In particular, two networks with 5 GRUs (i.e. 5 layers, each one associated to a load of the AROMA network) are compared in Table 3.2: the first one has the single-power input, whereas the second one the single-power and the summation as inputs. In particular, a training procedure lasting 300 epochs is performed, since at that stage it is already possible to catch the methods difference. Moreover, each layer has a number of nodes proportional to the number of inputs ($n_{neurons} = [4, 4, 4, 5, 5] \cdot n_{inp}$, see Section 3.3.2.3), while the return part is not yet considered (see Section 3.3.2.2).

Input	**Layers**	**Neurons**	**FIT**[%]	$\mathbf{R^2_{min}}$[%]	$\boldsymbol{R^2_{max}}$[%]	**BE**	**Time**
P_i	5	[8,8,8,15,15]	77.6	84.8	98.1	298	39’
P_i and $\sum P_i$	5	[12,12,12,20,20]	85.5	92.1	99.3	274	42’

Table 3.2: Comparison between the two physics-based approaches having different inputs.

As can be seen from the previous table, giving the additional input represented by the power summation brings to much better results, even if in just 300 epochs, with respect

to the single-power input case. This is particularly evident regarding the minimum R^2 value (and hence the overall FIT) which rises by roughly 8.6%, while nonetheless requiring almost the same amount of time.
To conclude, the best input choice regarding power is the one that includes, in short, P_i and $\sum P_i$ for each GRU. In fact, it allows to reach high fitting values in a reasonable time and it is even scalable in case of much bigger district heating systems.

3.3.2.2. Return Network

Now that it has been explained how to treat the forward network and the loads in this innovative physics-based RNN approach, it is appropriate to consider the backward network as well. Typically, the return temperature is characterized by a pretty limited dynamics and the overall mass flow rate is not a tough variable to identify, once the single $\dot{m}_i$ are found by the loads-associated GRUs. This consideration leads us to implement a GRU having a single hidden layer for the return network too.

First, as usual, the outputs outgoing from this last network are T^r and $\dot{m}_r$. With regard to the latter, it would be intuitive to directly compute it as $\dot{m}_r = \sum_{j=1}^{n_{loads}} \dot{m}_j$. However, in order to avoid the summation of identification errors of $\dot{m}_j$, it is more accurate to predict $\dot{m}_r$.

Second, as far as the input is concerned, it is straightforward to include all the load return temperatures since the objective is to identify the overall one. Indeed, it is not possible to select just one of them because they all influence T^r. Additionally, we can feed the GRU corresponding to the return pipelines with all the single mass flow rates identified by the loads-associated RNNs.
The so-formed return network can be schematically represented as in Figure 3.7.

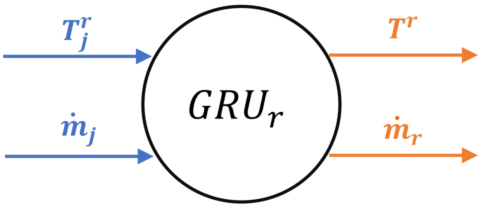


Figure 3.7: Schematic physics-based GRU used to hard-code the whole return network. Inputs are depicted in light-blue, outputs in orange. The index j stands as usual for $j = \{1, ..., n_{loads}\}$.

3.3.2.3. Overall PB-RNN with Graph Theory

Thanks to the previous explanation regarding the different parts of a physics-based model for a district heating network, it should be now clear how to assemble them. A graphic visualization helping to understand the complete PB-RNN of the AROMA network is reported in Figure 3.8.

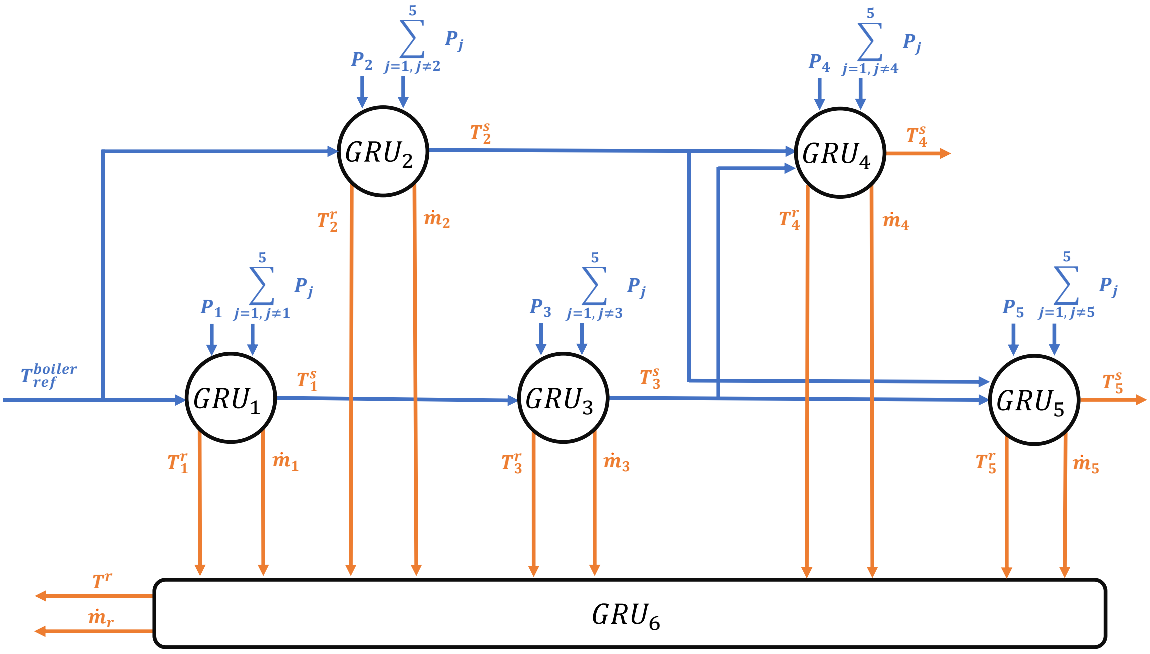

P_2
$\sum_{j=1, j\neq 2}^{5} P_j$
GRU_2
T_2^s
T_2^r
$\dot{m}_2$
P_4
$\sum_{j=1, j\neq 4}^{5} P_j$
GRU_4
T_4^s
T_4^r
$\dot{m}_4$
P_1
$\sum_{j=1, j\neq 1}^{5} P_j$
P_3
$\sum_{j=1, j\neq 3}^{5} P_j$
P_5
$\sum_{j=1, j\neq 5}^{5} P_j$
T_{ref}^{boiler}
GRU_1
T_1^s
GRU_3
T_3^s
GRU_5
T_5^s
T_1^r
$\dot{m}_1$
T_3^r
$\dot{m}_3$
T_5^r
$\dot{m}_5$
T^r
$\dot{m}_r$
GRU_6

Figure 3.8: Scheme of the implemented physics-based RNN, highlighting the six GRUs and their input and output variables. Inputs are depicted in light-blue, outputs in orange. Being T_i^s an output for the i^{th} load and an input for the subsequent one, it is represented both in light-blue and in orange.

At this stage, an insight on the implementation of this PB-RNN is suitable. In the first place, in order to train the model the usual library [8] has been used. However, a modification in the network structure has been applied.

In particular, the program receives as input (apart from the usual training, validation and test sets, the batch size, the number and the length of subsequences to extract for training/validation/testing) what we might call a "pseudo-incidence" matrix. In detail, the so-called "incidence" matrix is a well-known mathematical tool largely used in the graph theory applied to electrical networks. In fact, algebraic graph theory informs electrical network analysis, dynamics and design and it has enabled fundamental advances in the theory of the electrical networks field [20].

In detail, we say that a digraph is strongly connected if there exists a directed path from any node to any other node. The matrix which mathematically describes this type of graphs is called the incidence matrix, i.e. $B \in \mathbb{R}^{n \times m}$ of graph $\mathcal{G}$ is defined by [20]

$$B_{ie} = \begin{cases} +1, & \text{if the edge } e \text{ is } (i,j) \text{ for some } j \\ -1, & \text{if the edge } e \text{ is } (j,i) \text{ for some } j \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

We can easily extend to a district heating network the definition of the incidence matrix related to electrical networks, once the DHN oriented graph is available. In fact, by looking at Figure 3.3 it is simple to find the incidence matrix of the AROMA network:

$$B = \begin{bmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & +1 & +1 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & -1 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{3.5}$$

However, for the implementation of a PB-GRU is not necessary the entire information contained in B. First, not all nodes of the graph correspond to a load, which is exactly what we are interested in hard-coding. Second, the actual information a PB-RNN is interested in is what are the inputs and the outputs of each single GRU. For this reason, we introduce what we call the "pseudo-incidence" matrix that contains the information regarding the loads-associated network (the return part must be treated differently). Therefore, this new matrix can be defined as

$$Q_{ij} = \begin{cases} +1, & \text{if load in row } i \text{ provides information to load in column } j, \\ & \text{i.e. the input of load } j \text{ is the output of load } i \\ -1, & \text{if load in row } i \text{ receives information from load in column } j, \\ & \text{i.e. the input of load } i \text{ is the output of load } j \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

In the AROMA network case, the pseudo-incidence matrix is

$$Q = \begin{bmatrix} 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & +1 & +1 \\ -1 & 0 & 0 & +1 & +1 \\ 0 & -1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 \end{bmatrix} \tag{3.7}$$

where the rows and columns of Q correspond to the network loads, ordered as in Figure 3.8. The users order is fundamental since it is necessary to sort the physics-based GRUs according to the flow information. For instance, load 2 provides information to load 5, and hence the former must be identified before the latter. This is the reason why consumers were originally enumerated according to their distance with respect to the heating station. We could translate the Q matrix information as follows:

- the 1^{st} load provides the supply temperature identified by its corresponding GRU to the 3^{rd} load;
- the 2^{nd} load provides the supply temperature identified by its corresponding GRU to the 4^{th} and 5^{th} loads;
- the 3^{rd} load receives T_1^s as input and provides the supply temperature identified by its corresponding GRU to the 4^{th} and 5^{th} loads;
- the 4^{th} and 5^{th} loads receive T_2^s and T_3^s as inputs.

Once this matrix is defined by the user, the program computes two other matrices.

First, it calculates the "original input matrix", which contains the information regarding which GRU (in our case there are 6 of them, corresponding to rows, linked to the 5 loads and the return part) needs which "original" inputs, i.e. the inputs originally provided to the code and not computed during the training procedure (T_{ref}^{boiler}, P_1, P_2, P_3, P_4, P_5, corresponding to columns):

$$M_{orig} = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.8}$$

We could translate the matrix information as follows:

- the 1st GRU, corresponding to load 1, receives as original inputs T_{ref}^{boiler} (1 in position [1,1]), P_1 and the sum of the other load consumptions (2 in position [1,2]);
- the 2nd GRU receives as original inputs T_{ref}^{boiler}, P_2 and the sum of the other load consumptions;
- the 3rd GRU receives as original inputs P_3 and the sum of the other load consumptions;
- the 4th GRU receives as original inputs P_4 and the sum of the other load consumptions;
- the 5th GRU receives as original inputs P_5 and the sum of the other load consumptions;
- the 6th GRU, corresponding to the return network, does not receive any original input.

Second, the program computes the "auxiliary input matrix", which contains the information regarding which GRU (corresponding to rows) needs which "auxiliary" inputs, i.e. the inputs computed during the training procedure (T_1^s, T_1^r, $\dot{m}_1$, T_2^s, T_2^r, $\dot{m}_2$, T_3^s, T_3^r, $\dot{m}_3$, T_4^s, T_4^r, $\dot{m}_4$, T_5^s, T_5^r, $\dot{m}_5$, corresponding to columns):

$$M_{aux} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{3.9}$$

We could translate the matrix information as follows:

- the 1st and 2nd GRUs do not receive any auxiliary input;
- the 3rd GRU receives as auxiliary input T_1^s;
- the 4th and 5th GRUs receive as auxiliary input T_2^s and T_3^s;
- the 6th GRU receives as auxiliary input T_1^r, $\dot{m}_1$, T_2^r, $\dot{m}_2$, T_3^r, $\dot{m}_3$, T_4^r, $\dot{m}_4$, T_5^r and $\dot{m}_5$.

Moreover, the program needs to be fed with two other informative vectors: one containing the number of outputs for each layer ([3, 3, 3, 3, 3, 2]) and one containing the coefficients used to compute the number of neurons for each layer. This last idea requires an additional explanation.

In traditional RNNs, since hidden layers do not have a physical meaning, it is difficult to decide which layers should have a greater number of nodes and as a consequence, as seen in Chapter 2, they typically all have the same amount of neurons. By contrast, in PB-RNNs the single-layer GRUs are associated to a physical part of the system analysed: it is possible to understand which layers are characterized by a more difficult identification and hence which layers should be helped with a greater number of nodes. To sum up, we could conclude that, by observing traditional RNNs performance, it is possible to figure out which variables are more complex to be identified. Thus, in PB-RNNs it is convenient to set a greater number of neurons for those corresponding challenging GRUs. As usual, each identification problem requires a trial and error parameter set-up, but the rule of thumb driving the choice is the one just mentioned: in a nutshell, the farther the load, the higher the number of neurons associated with it.
For instance, in the case of the AROMA PB-RNN, the first three GRUs depicted in Figure 3.8, being the closest to the heating station, do not seem to show huge problems in the identification procedure. Instead, the last two GRUs of the same figure are the most troublesome in terms of R^2, and in fact they achieved the worst values when using traditional RNNs, especially the fifth (last) load. Hence, thanks to the matching between the physical system and the neural network structure, we may modify the number of nodes at our convenience. Specifically, another indicative rule could be to multiply the number of inputs by a certain constant, which is then the only parameter to define: $n_{nodes} = k \cdot n_{inp}$.

- GRU 1, 2 and 3 do not have particular identification problems, being the associated consumers close to the heating station, and hence their number of nodes could be restrained. These three networks have 3 inputs each, thus the number of neurons could be found as $n_{nodes} = k \cdot n_{inp} = 3 \cdot 3$;
- GRU 3 and 4, corresponding to the furthest users in the AROMA network, are the most problematic in terms of R^2 values and hence they should be helped by boosting their number of neurons, by using for example $k = 4$;
- GRU 6, which corresponds to the return network, has 10 inputs. In addition, T^r and $\dot{m}_r$ are typically characterized by quite low fitting values and hence we could set $k = 3$ in order to have a total of 30 neurons for this RNN.

In conclusion, the overall vector defining the multiplication constants by which each number of input must be multiplied to compute the amount of nodes is [3, 3, 3, 4, 4, 3], in the AROMA case. It is convenient to highlight once more that the aforementioned principle is mainly a rule of thumb useful to guide the programmer selecting a suitable number of neurons, by lowering and raising it where needed. Obviously, the higher n_{nodes} the greater the accuracy (apart from over-fitting cases) but also the longer the training time. Thereby, a small amount of neurons in some GRUs can compensate higher number of nodes in other GRUs, where strictly required, in terms of complexity and training time. Thanks to this trick, which is not possible to employ in standard RNNs, we expect to significantly enhance the R^2 values associated to the most problematic variables, and hence the average FIT.

Once these input matrices and vectors are defined, the program iterates with a *for* loop the initialization and the actual training procedure for every GRU, each one receiving its corresponding original and auxiliary inputs and number of outputs and of neurons.

What we expect to obtain by means of this novel PB-RNN is mainly an increase in the minimum R^2 value and thus in the total FIT, thanks to the physics-based structure of the overall neural network. In other words, feeding each GRU with the most appropriate inputs, found through a careful analysis of the matching between the physical system and the data-driven algorithm, should boost the performance even in terms of training time thus reaching the target FIT more quickly than traditional methods.

3.4. Results Comparison

This section is dedicated to an analysis of the implementation results of the PB-RNN, with a focus on the comparison between this novel approach and the traditional one.
In particular, a PB-GRU network with 6 hidden layers and [9,9,9,16,16,30] nodes is compared to a standard GRU having the same number of layers and 15 neurons each layer (being 15 approximately the average of the physics-based n_{nodes}). In addition, both networks are trained over a period of 1500 epochs, using the ADAM optimizer with a learning rate of 0.003. In such a way, the comparison is fair and the only difference lays in the structure of the machine learning algorithm (see Figure 3.1 and 3.8).

By plotting the FIT evolution of the two networks, a striking result can be appreciated (Figure 3.9).

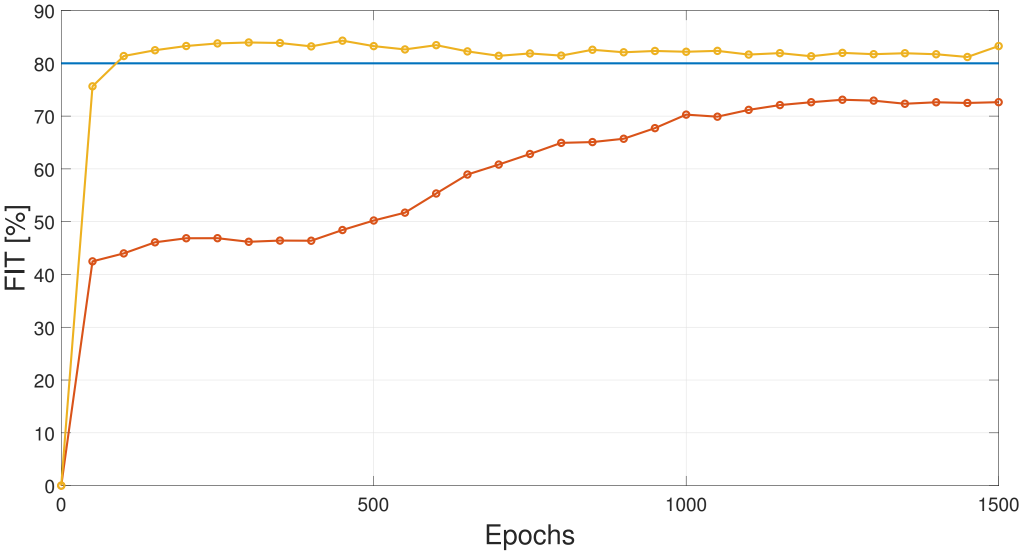


Figure 3.9: Comparison between the FIT trend of a traditional RNN (red) and the FIT trend of a physics-based RNN (yellow). The target FIT is represented in blue.

Actually, two impressive improvements are visible from the previous plot. On one side, the overall FIT value reached by the PB-RNN is definitely greater than the one achieved by the standard RNN. On the other side, the PB-RNN takes less than 100 epochs to overcome a FIT of 80%, which is a value that the standard RNN does not even reach after 1500 epochs.

In order to quantitatively compare these results, the FIT, the minimum and maximum R^2 values, the best epoch (BE) and the time to reach the latter of the two approaches are reported in Table 3.3.

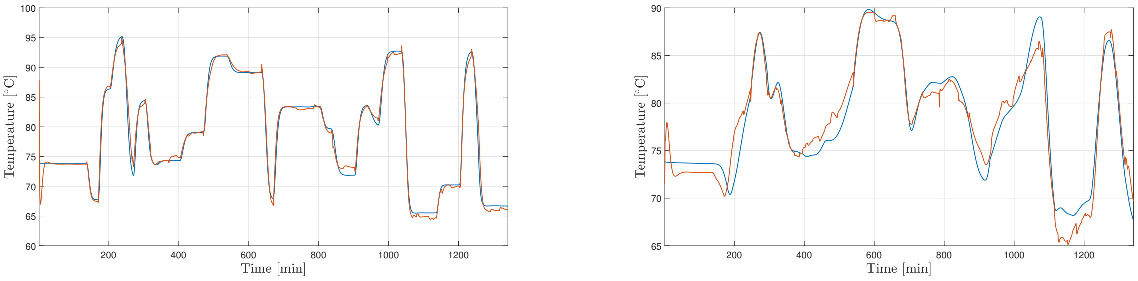
Model	**Layers**	**Neurons**	**FIT**[%]	R^2_{min}[%]	R^2_{max}[%]	**BE**	**Time**
GRU	6	[15,15,15,15,15,15]	72.6	69.7	97.5	1495	2h29’
PB-GRU	6	[9,9,9,16,16,30]	83.3	89.4	98.5	355	1h12’

Table 3.3: Comparison of the AROMA network identified with standard RNN and PB-RNN.

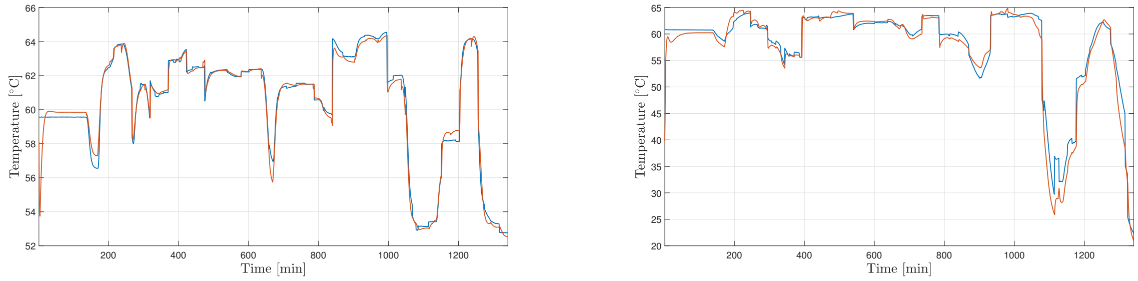
Firstly, the enhancement in the minimum R^2 value is astonishing. In detail, R^2_{min} corresponds, for both networks, to the supply temperature identified by the fifth GRU. This is coherent with Chapter 2 identification results and with physics: it is the furthest consumer and hence the most complex to predict. However, thanks to the physics-based RNN structure, the R^2 associated to this quantity increases from 69.7% to 89.4%, which is a big bump. As expected, the maximum R^2, corresponding to T_1^s, has a smaller rise. Altogether, the overall FIT has an absolute growth of 10.7%, i.e. a percentage increase of almost 15%, which is a huge accomplishment.

In addition, the time to reach the best epoch, namely the epoch where the FIT is maximum, is reduced when using a physics-based approach. An interesting remark can be drawn by observing the overall training time and the time per epoch of the two approaches. In fact, standard RNN took two and a half hours to reach the last epoch (1500), with an average time per iteration of 6 s/it, whereas PB-RNN took four hours and forty, overall, with an average time per iteration of 11.2 s/it. This means that, despite the physics-based approach is, on the whole, slower, thanks to its structure resembling the actual physical system it takes far fewer iterations to reach outstanding results.

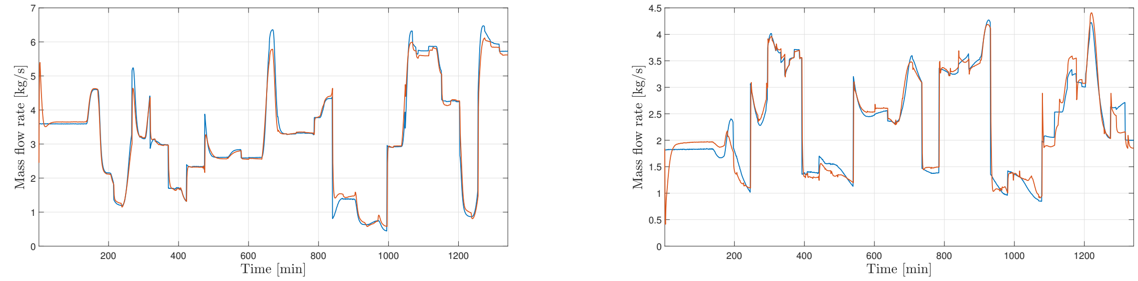
Finally, the plots of some identified variables are here reported. For the sake of brevity, not all the seventeen outputs are displayed, but just the most significant, i.e. the variables of the first and fifth GRU, plus the overall return temperature and mass flow rate.



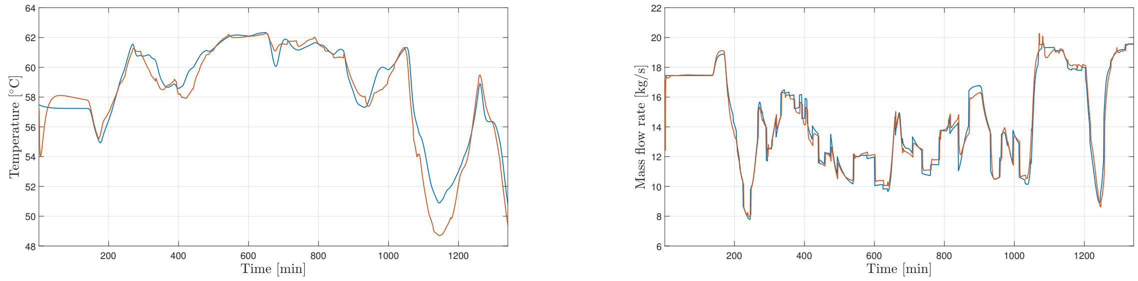
(a) First GRU supply temperature (b) Fifth GRU supply temperature



(c) First GRU return temperature (d) Fifth GRU return temperature



(e) First GRU mass flow rate (f) Fifth GRU mass flow rate



(g) Return temperature (h) Return mass flow rate

Figure 3.10: Variables identified by a PB-GRU (red) compared to the real ones (blue).

3.5. PB-ML Improvements over Traditional RNNs

The bottom line of this chapter analysis is a major improvement of the PB-GRU network over standard RNNs, which we may here summarize.

Practically speaking, a knowledge-based machine learning approach leads to higher fitting values: in other words, the goal of increasing the predictive accuracy with respect to black-box methods is largely attained. In addition, thanks to a precise structure resembling the physical system, the training time to reach great results drops with respect to standard data-driven techniques.

Conceptually speaking, another benefit of PB-RNNs is the clarity and intelligibility of such an approach. Actually, standard neural networks do not allow a practical understanding since their layers do not have a tangible equivalent. On the contrary, thanks to the matching between the physical and the computer world of PB-ML approaches, a complex structure such as the neural network one can be smoothly interpreted even by machine learning non-experts.
The other advantage that interpretability brings with it is the ease of localizing problems and thus of solving them. In district heating systems, for instance, it is well known that the most distant consumers variables are the toughest to identify. Therefore, thanks to the association of each RNN to a consumer (or, equivalently, consumers cluster) it is possible to treat different difficulty levels of identification diversely, e.g. by increasing the number of neurons only in GRUs where it is strictly needed to intervene. Obviously, this does not apply to traditional RNNs because of their abstract structure.

Finally, the only questionable drawback of physics-based techniques is the more intricate implementation and the necessity of physical knowledge of the system by the programmer. In traditional RNNs, instead, once input and output data are collected, they can be directly given to machine learning algorithms, even in absence of a system acquaintance.

3.6. Generalization of the PB-RNN Approach

So far we have presented this novel method applied to our specific case study, i.e. the AROMA network, which is a simple thermo-hydraulic system particularly suitable for this kind of experiments. This approach, in fact, is perfectly appropriate to treat district heating networks, thanks to their modular architecture and their sequential nature. Indeed, this is particularly true for radial networks, which have a well-defined flow direction (that is something not necessarily right for meshed networks, as previously explained).

Ultimately, the discussed technique could be eventually applied, under certain conditions, to much more complex and extended systems than the AROMA network. In particular, the plant under investigation must be characterized by a modular topology, which should be translated into an oriented graph and thus into an equivalent scalable neural network.

To accomplish this, each load or loads cluster must be turned into an RNN having one (or more) layer with a certain number of neurons. Similarly, the return pipelines must be associated with its corresponding RNN.
Moreover, one has to define the pseudo-incidence matrix, the original and auxiliary inputs and the outputs of the system. As previously explained, the GRUs associated to consumers have as outputs their own supply and return temperature and mass flow rate. By contrast, their inputs must be carefully selected through an empirical verification. Hence, apart from the corresponding power demand and the sum of the others, each RNN is fed with the supply temperatures of the loads which directly affect that RNN. This is obtained by studying the actual direction of the network flows. It is necessary to go backwards starting from the i^{th} user by following all the possible paths connected to it, until a new load is met: this last consumer's supply temperature will be an input for the i^{th} load. In addition, the return network is given as inputs the consumers' return temperatures and mass flow rates (or sum of $\dot{m}_i$).
Lastly, the programmer has to set a certain number of neurons (actually of multiplication factors, once the number of inputs is fixed) for each part of the modular system turned into an RNN layer (we used GRU because of Chapter 2 considerations, but other types of networks can be certainly employed).

3.7. PB-GRU Sensitivity Analysis

In this final section, some changes are introduced into the model in order to test the robustness of the previously analysed method. In particular, these adjustments are made in view of control and optimization problems. Indeed, in Chapter 4 a Model Predictive Control (MPC) algorithm will be implemented to optimally manage the AROMA network over a daily operation. Hence, for this target, it is essential to simplify the identified model so as to lighten the computational burden of such a complex algorithm.

Reduced Number of Neurons

A first convenient strategy is to reduce the overall number of network states, by minimizing the amount of neurons per hidden layer. To this purpose, a PB-GRU having 6 layers and [6,6,6,8,12,16] neurons is implemented, according to the rule of thumb discussed in Section

3.3.2.3: the farther the load, the higher the number of neurons associated with it. Indeed, the first three consumers usually do not present significant identification issues (6 neurons each). By contrast, the fifth load (fifth GRU and thus fifth layer of the network) is the most troublesome in terms of identification accuracy (being the furthest), and hence 12 neurons are associated to it. The fourth load instead seems to have less predictive issue, being slightly closer to the heating station than the fifth user, thereby 8 neurons are sufficient. Finally, the return network (sixth layer) is assigned 16 neurons, thus having altogether $n_{nodes} = [2,2,2,2,3,1.6] \circ [3,3,3,4,4,10]$.

In particular, we expect that having 54 states instead of 89 can significantly reduce the computational time of an MPC algorithm execution (see Chapter 4). Finally, it is useful to compare this new network with a standard GRU having again 6 layers and 9 neurons per layer (being 9 the average of [6,6,6,8,12,16]), as reported in Figure 3.11 and in Table 3.4.

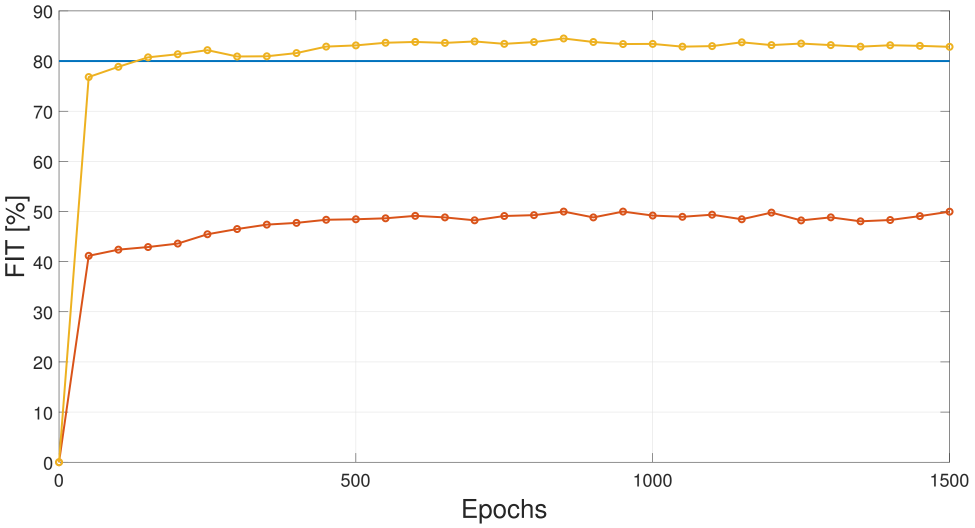


Figure 3.11: Comparison between the FIT trend of a traditional RNN (red) and the FIT trend of a physics-based one (yellow), both having 54 states. The target FIT is represented in blue. The model samples are collected every minute.

Model	Layers	Neurons	FIT[%]	R^2_{min}[%]	R^2_{max}[%]	BE	Time
GRU	6	[9,9,9,9,9,9]	49.9	9.2	95.2	1234	1h55'
PB-GRU	6	[6,6,6,8,12,16]	82.8	85.1	98.9	727	1h42'

Table 3.4: Comparison of the AROMA network identified with standard RNN and PB-RNN, using a restricted amount of neurons per layer. The model samples are collected every minute.

As expected, a smaller number of neurons may affect a bit the identification performance, which is particularly evident regarding the traditional GRU: its FIT value drops from 72.6% to 49.9% when the states are diminished. In addition, the minimum R^2 value is very low and it is associated to the fifth consumer's mass flow rate. However, the physics-based model maintains its great performance with an average FIT of 82.8% (instead of 83.3%) and a minimum R^2 value of 85.1% (instead of 89.4%). In conclusion, thanks to the physics-guided approach, it is possible to lower the model complexity while still preserving an impressive accuracy. This is especially helpful when handling much bigger physical systems that may require a higher amount of layers than this case study demands: by shrinking the total number of states, the problem complexity can be narrowed down.

Sampling Time Increase

Another possible modification to ease the control problem is to use a slightly greater sampling time. For instance, by using a $T_{sampling}$ of five minutes instead of one minute, it is possible to solve a problem with a prediction horizon five times greater (5N) than a prediction horizon (N) of a problem with a sampling time of one minute, having an equal amount of optimization variables (see Chapter 4).

The procedure adopted to identify a model whose samples are collected every five minutes is exactly the same as the one explained in Chapter 2 and 3. In detail, in order to have the same quantity of data as the one-minute-sampled system, instead of simulating the process for roughly 11 days (10 for training and validation and 1 day for testing), 55 days of simulation are tested (50 for training and validation and 5 days for testing). In this way, even though the overall period of simulation is obviously wider, the datasets of the one-minute-sampled and five-minute-sampled model are equivalent.

Finally, it is possible to graphically visualize this adjustment, so as to show that the identification performance is coherent with the one discussed so far and it is not affected by the sampling time choice. To do so, a physics-based GRU having [6,6,6,8,12,16] neurons and sampled at five minutes is compared to a standard GRU having [9,9,9,9,9,9] neurons and again sampled at five minutes are compared (see Figure 3.12 and Table 3.5).

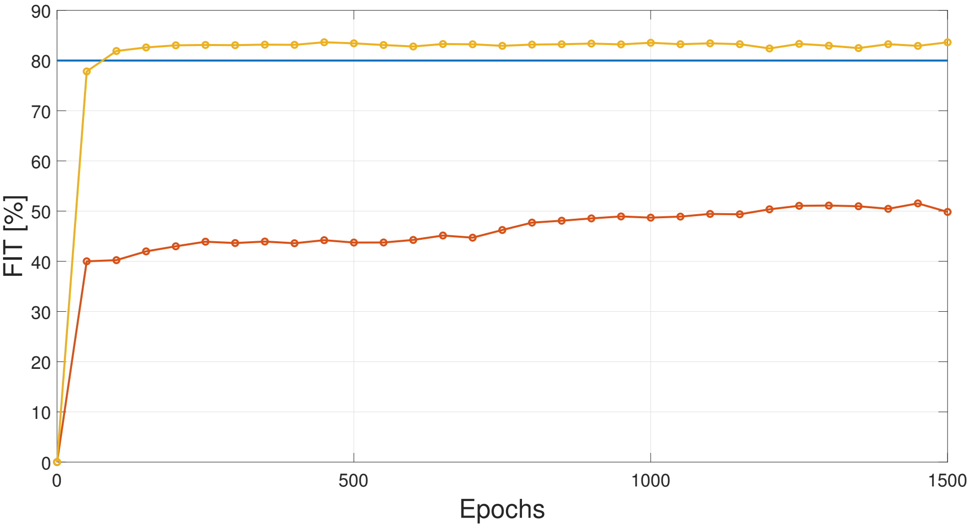


Figure 3.12: Comparison between the FIT trend of a traditional RNN (red) and the FIT trend of a physics-based one (yellow), both having 54 states. The target FIT is represented in blue. The model samples are collected every five minutes.

Model	**Layers**	**Neurons**	**FIT**[%]	R^2_{min}[%]	R^2_{max}[%]	**BE**	**Time**
GRU	6	[9,9,9,9,9,9]	49.8	36.9	95.2	1171	1h50'
PB-GRU	6	[6,6,6,8,12,16]	83.6	87.5	99.2	557	1h23'

Table 3.5: Comparison of the AROMA network identified with standard RNN and PB-RNN, using a restricted amount of neurons per layer and 15200 samples. The model samples are collected every five minutes.

As clearly visible from the previous plot and table, having a sampling time of five minutes does not impact on the identification accuracy and time, confirming that the method is robust even with respect to the sampling time. Once again, the standard GRU performance is definitely poorer than the PB-GRU one, even if under same conditions.

Reduced Number of Data

Lastly, considering the robustness of the physics-based neural networks with respect to the amount of neurons and the sampling time, it is reasonable to think that this method could also be powerful when handling a smaller amount of data. As always, this statement must be empirically verified: it is necessary to identify the AROMA network whose samples are collected every five minutes, but this time with an overall simulation lasting 11 days in place of 55. In this way, we preserve the same period of the identification procedure analysed in Chapter 2. However, being $T_{sampling} = 5$ min, we have at our disposal one fifth of the original data, i.e. 3040 samples instead of 15200.

By plotting the usual FIT-epochs graph (Figure 3.13), one can observe how the PB-GRU trained with 20% of data (depicted in purple) still performs satisfactorily. However, intuitively, its FIT trend is lower than the FIT of the PB-GRU identified with a dataset containing 55 days of samples collected every five minutes (depicted in yellow). This is even more evident in the case of standard RNNs. Indeed, as visible in Figure 3.13, when the network is identified with a much smaller dataset (depicted in blue), its performance dramatically drops leading to an unacceptable model accuracy. Finally, as reported in Table 3.6, the PB-RNN trained with the reduced dataset, not only yields to reasonably good results, but also it is able to achieve its best FIT value in a shorter period of time than the PB-GRU trained with the complete dataset (Table 3.5).

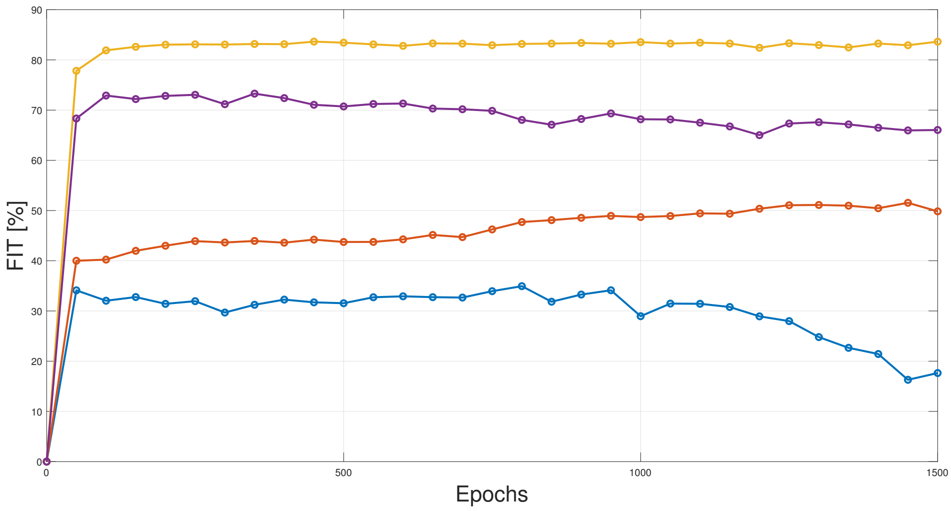


Figure 3.13: Comparison among the FIT trend of a traditional RNN trained with 15200 samples (red), of a PB-RNN trained with 15200 samples (yellow), of a traditional RNN trained with 3040 samples (blue) and of a PB-RNN trained with 3040 samples (purple). The model samples are collected every five minutes.

Model	**Layers**	**Neurons**	**FIT**[%]	$\boldsymbol{R^2_{min}}$[%]	$\boldsymbol{R^2_{max}}$[%]	**BE**	**Time**
GRU	6	[9,9,9,9,9,9]	32.3	-11.9	86.1	29	2’
PB-GRU	6	[6,6,6,8,12,16]	71.7	67.2	94.8	102	12’

Table 3.6: Comparison of the AROMA network identified with standard RNN and PB-RNN, using a smaller amount of data (3040 samples). The model samples are collected every five minutes.

For the sake of completeness, it is honest to underline that all the physics-based approaches and the standard ones have always been compared fairly: same dataset, optimizer, same number of hidden layers, states and epochs. However, it has been noted that the standard GRU does not actually require so many layers, necessary instead for the physics-informed structure itself. In fact, six hidden layers lead the traditional RNN to over-fitting. By contrast, a standard GRU having two hidden layers, each with 17 neurons, is able to improve the performance. Nevertheless, by comparing this best configuration of the traditional GRU with the PB-GRU, the latter still outperforms the standard network. In detail, in Figure 3.14 it is compared the FIT trend of a traditional RNN (red) and the FIT trend of a PB-RNN (yellow), both trained with the reduced dataset made by 3040 samples.

Even though the performance of the 2-layer standard GRU enhances with respect to the one having six layers, the physics-based one is still superior in terms of FIT and R^2_{min} (see Table 3.7).

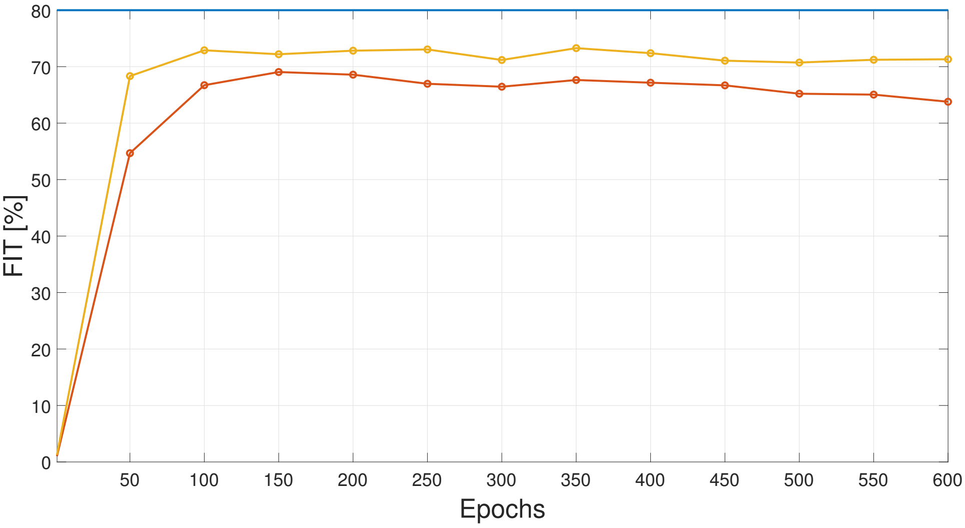


Figure 3.14: Comparison between the FIT trend of a traditional RNN (red) and the FIT trend of a PB-RNN (yellow), both trained with 3040 samples. The model samples are collected every five minutes. The training was stopped after 600 epochs because over-fitting occurred.

Model	Layers	Neurons	FIT[%]	$\boldsymbol{R^2_{min}}$[%]	$\boldsymbol{R^2_{max}}$[%]	BE	Time
GRU	2	[17,17]	68.3	61.8	95.1	162	9'
PB-GRU	6	[6,6,6,8,12,16]	71.7	67.2	94.8	102	12'

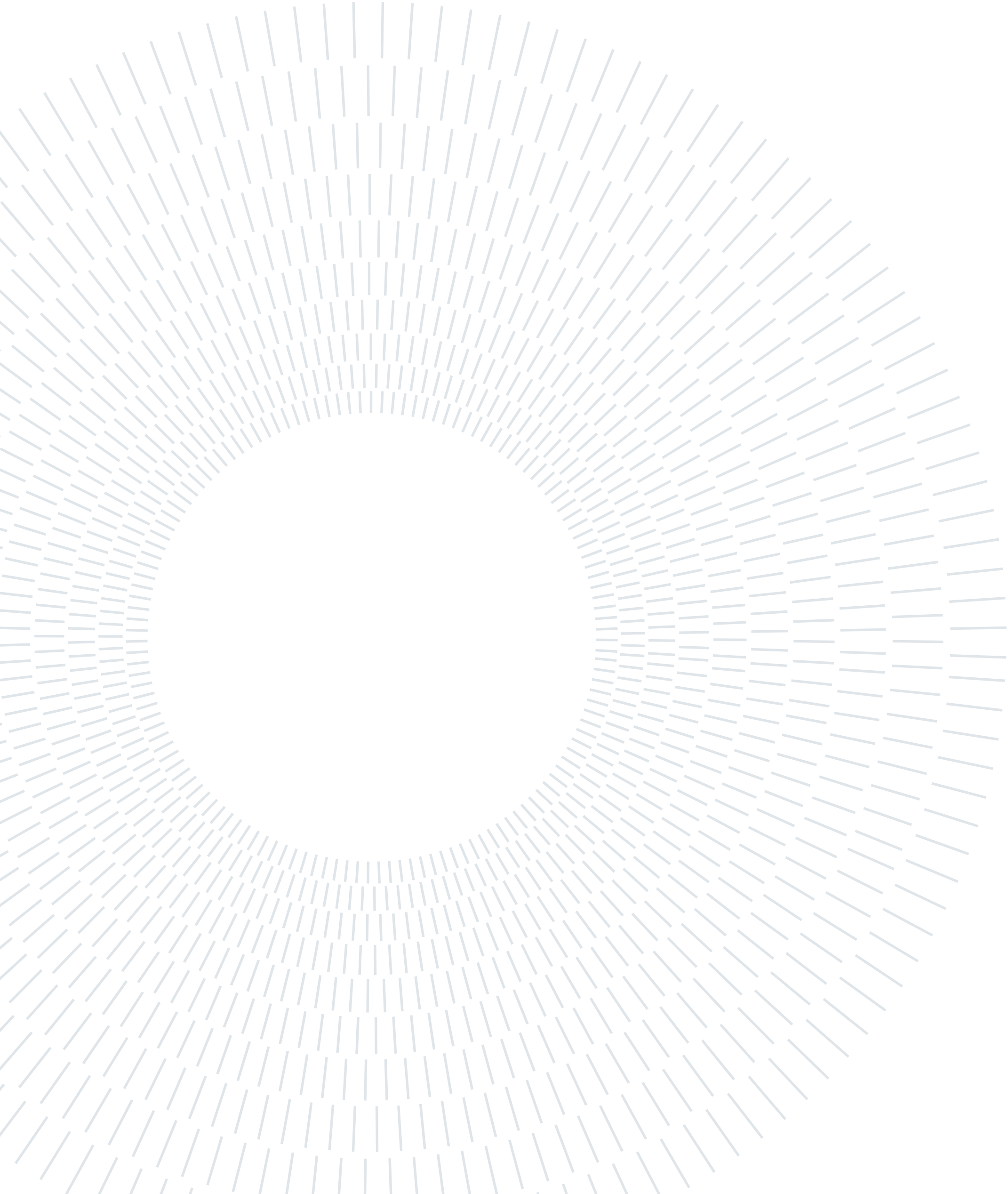
Table 3.7: Comparison of the AROMA network identified with standard RNN (best configuration) and PB-RNN, using a smaller amount of data. The model samples are collected every five minutes.

In conclusion, identifying a system model with a physics-guided machine learning approach allows to increase the model accuracy, diminish the computational time and even to use a narrow dataset, when it is not possible to collect plenty of samples.

A final remark: for control purposes the dataset shrinkage is irrelevant, given that it only affects the identification procedure and not the optimization one. In fact, in MPC the system model is exploited in the form of non-linear equations, and hence, once the network is trained, only its weights and biases are needed. Consequently, in Chapter 4 just the first two modifications are adopted, i.e. a drop in the number of states and a rise in the sampling time.

3.8. Conclusions

In this chapter the main challenge of the thesis has been tackled. In particular, it has been proposed a novel architecture for RNNs, whose structure is inspired by the physical system topology. In this way, the PB-GRU networks that mimic the interactions among physical loads are able to improve standard RNNs from different points of view: higher predictive accuracy, faster training procedure, greater interpretability and easier problem detection. Even though in literature several physics-based methods have been proposed, they are mainly based on upgraded loss functions and, above all, they are strongly problem-dependent. This is why the method proposed here is particularly effective: a generic plant with a modular structure and clear physical cause-effect relationships can be modelled through a physics-informed neural network that resembles those physical characteristics. In addition, thanks to the scientific knowledge about the system and to the help of graph theory, it is pretty straightforward to implement such neural networks.



4 | Model Predictive Control using Physics-based Neural Networks

4.1. Chapter Overview

This chapter is focused on the optimization and control of the AROMA network. After a general introduction about Model Predictive Control (MPC), the problem set-up regarding the case study under discussion is described, as well as the optimization environment. Then, its implementation using the non-linear equations of the PB-GRU model is analysed, together with a description of the cost function, constraints, observer and disturbance forecasting needed to solve the optimization problem. In addition, some strategies adopted to lighten the computational effort are outlined. Finally, some results are analysed, along with a comparison between an MPC regulator that makes use of the PB-GRU model and an MPC regulator exploiting standard GRU equations.

4.2. Model Predictive Control

Model predictive control is the most popular advanced control method in industry and in embedded applications. Indeed, all the main automation companies such as ABB and Siemens are equipped with software tools for its implementation. It was developed between the late seventies and early eighties in the process industry to cope with large-scale systems, constraints on the process variables and time varying reference signals [46].

The basic concept is to transform the control synthesis problem into an optimization one, hence a finite-horizon control problem is stated and then solved. In practice, a time invariant control law is obtained by means of the Receding Horizon principle: at time k the future sequence of control variables is computed, but only its first value is used, and at time $k+1$ the optimization procedure is repeated with the same prediction horizon (N). To sum up, MPC is based on the knowledge of a dynamic model of the system so as to compute the future evolution of the controlled variables as function of the future evolution of the control inputs. Moreover, the input sequence is computed by minimizing a cost function under state, input and output constraints [46].

The strength of this algorithm lies in a higher efficiency and tighter control with respect to traditional PID schemes: the reference signals can be set to values near the operating constraints, resulting in economic benefits too. Additionally, because of its structure, MPC can easily consider the knowledge of future external disturbances, leading to a control action improvement. For a better insight on the topic, see the source of this chapter [46].

4.3. Non-linear MPC for a DHS using PB-RNNs

Now that a performing model of the AROMA network has been identified (Chapter 3), it is appropriate to synthesize a controller using the aforementioned model. As anticipated, MPC allows to exploit the long-term prediction capabilities of the model while fulfilling input and output constraints. Typically, the feedback controller, depicted in Figure 4.1, is made by an observer that estimates the model state x_k and a finite-horizon control optimization problem which exploits such state estimation [11].

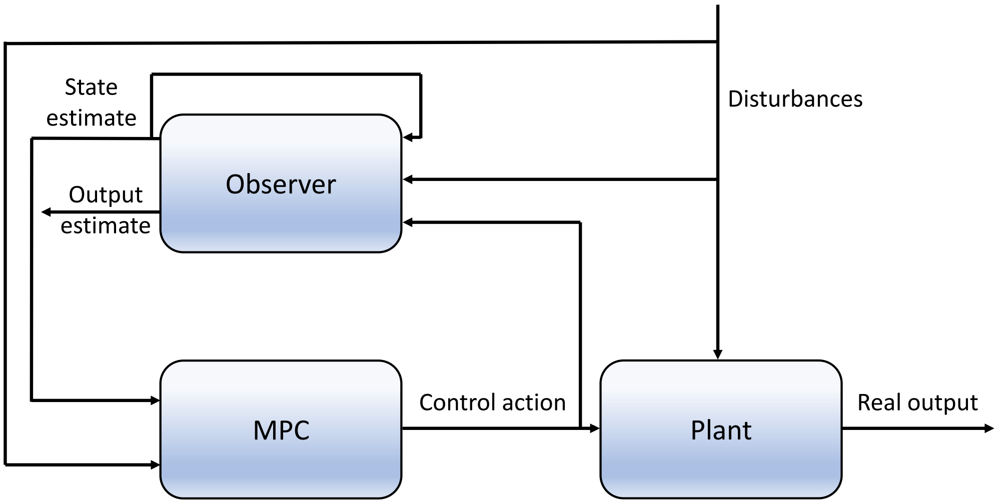


Figure 4.1: Scheme of the control algorithm, including the optimization part (MPC), the observer and the plant.

Specifically, the plant consists in the usual physical model of the AROMA network implemented through Modelica (Chapter 1). Moreover, the observer, as explained in Section 4.3.7, contains the model identified by means of a physics-based Gated Recurrent Unit, in the form of non-linear equations. Finally, the MPC block is actually far more complex than the simplified representation of Figure 4.1, and this section aims to examine it thoroughly. Precisely, it is a non-linear MPC: the system is non-linear and hence the state dynamics imposes non-linear constraints [46]. This requires the definition of the Lagrangian function and the Karush–Kuhn–Tucker (KKT) conditions, which are automatically computed by the optimization software (see Section 4.3.1). In conclusion, the design of a non-linear MPC, because of the formulation of non-convex optimization problems, is definitely a challenging task [91].

4.3.1. Optimization Environment

Before going into details of the Finite Horizon Control Optimization Problem (FHCOP), a quick introduction about the software employed is advisable. In particular, CasaADi paired with Ipopt is exploited in the MATLAB environment.

CasaADi is an open-source tool for non-linear optimization and algorithmic differentiation, which facilitates rapid and efficient implementation of different methods for numerical optimal control, both in an offline context and for non-linear model predictive control (NMPC)[1].

To solve non-linear programming (NLP) problems, characterized by a cost function and some constraints, Ipopt, which is already included in CasaADi, is exploited. The acronym stands for Interior Point Optimizer and it is an open-source software package for large-scale non-linear optimization: it implements an interior point search filter method that aims to find a local solution of NLP[2]. The mathematical details can be found in several publications such as [87].

4.3.2. Problem Statement

Now that a general overview on MPC has been provided, the optimization and control problem can be formulated. In a few words, this will regulate the boiler reference temperature (control action) minimizing its electrical cost while satisfying power, mass flow rate and temperature constraints. In particular, we aim at keeping the optimization problem as simple as possible, since its complexity is out of this thesis scope. In order to increase the problem sophistication, a cogeneration (CHP) and a thermal energy storage (TES) could be added as further degrees of freedom. However, the objective of this work is to show how a non-linear model obtained through a physics-guided machine learning procedure perfectly fits the control needs.

The FHCOP is solved online in order to find the optimal value of the optimization variables, which can be collected in the vector $\Sigma = [x, u, y, P^{boiler}, s]$, where

---

[1] `https://web.casadi.org`

[2] `https://coin-or.github.io/Ipopt`

- x is the vector containing the system states, having dimension $n_{states} \times (N+1)$;
- u is the vector containing the overall input, i.e. the control variable and the disturbances: $u = [T^{boiler}_{ref}(k); P^{load}_1(k); P^{load}_2(k); P^{load}_3(k); P^{load}_4(k); P^{load}_5(k)]_{\forall k \in \{1,..,N\}}$;
- y is the vector containing the outputs of the physical system which are strictly needed in the optimization problem, i.e. not all the seventeen outputs of the actual plant identified by the GRU:
  $y = [T^s_1(k); T^s_2(k); T^s_3(k); T^s_4(k); T^s_5(k); T^r(k); \dot{m}_r(k)]_{\forall k \in \{1,..,N\}}$;
- P^{boiler} is the vector of dimension $1 \times N$ containing the boiler power along the prediction horizon N;
- s is the vector containing the slack variables related to load constraints (see Section 4.3.4). Theoretically, it should be a matrix of dimension $n_{slack} \times N$. However, in order to reduce the computational effort we could just consider the value of the slack variables at the current instant, since their prediction along N is needless (dimension: $n_{slack} \times 1$).

To sum up, all the optimization variables are reported in Table 4.1.

Symbol	**Description**	**SI unit**
$\dot{m}_r$	Overall mass flow rate	[kg/s]
P^{boiler}	Boiler power	[W]
P^{load}_i	Load power	[W]
$P^{load}_{i,real}$	Actual load demand	[W]
$\tilde{P}^{load}_i$	Load demand forecasting	[W]
s	Slack variables	[°C]
T^{boiler}_{ref}	Boiler reference temperature	[°C]
T^s_i	Load supply temperatures	[°C]
T^r	Return temperature	[°C]
x	System states	-
x_{obs}	Observed states	-

Table 4.1: Main optimization variables.

4.3.3. Disturbance Forecasting

Closed-loop performance of model-based control algorithms is directly related to model accuracy [59], that is why we selected the best model (PB-GRU with 54 states) among the ones discussed in previous chapters. In practice, modelling errors and unmeasured disturbances can lead to poor performance and steady-state offset unless precautions are taken in the control design [59].

Typically, in DHSs the daily thermal demand is unknown, even though in some cases it could be measured in real-time. As a consequence, these disturbances must be necessarily predicted considering that MPC exploits the non-linear equations that also depend on loads consumption. Until now, in fact, they have been treated as known inputs for neural networks.
However, for MPC purposes there are mainly two strategies one can follow. First, we could measure at current time the actual power demand and assume it remains constant in the future [59]. In real district heating systems though, thanks to the availability of many historical weather and load data measured at the plant, a disturbance forecasting algorithm could be effectively implemented [6]. Actually, since the AROMA network is a theoretical case study and historical data are clearly unavailable, a thermal load forecast could be obtained by slightly modifying the usual loads consumption (see Figure 1.10b) and by adding a white Gaussian noise (WGN) to it. A comparison between the overall real power request and the "predicted" one is reported in Figure 4.2.

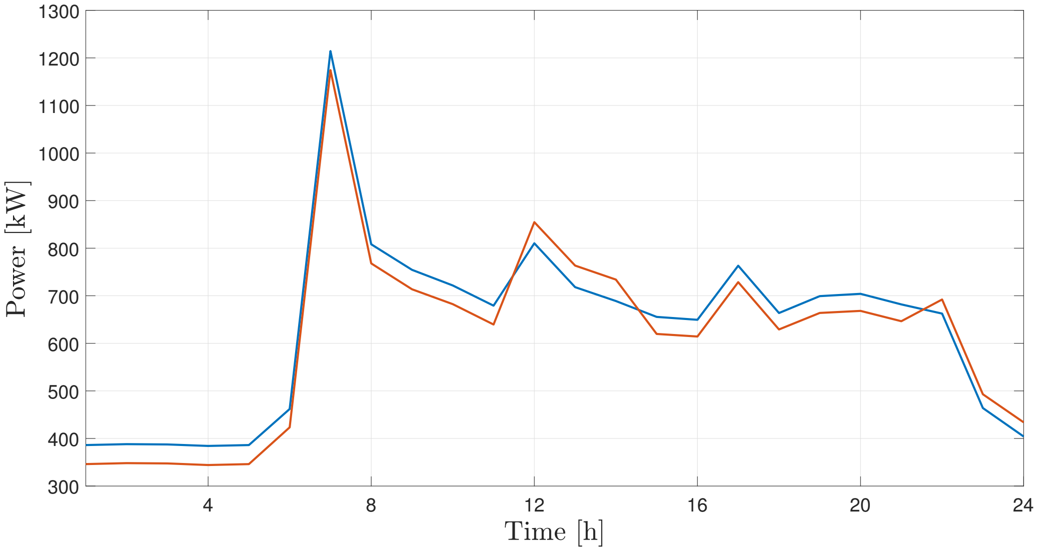


Figure 4.2: Comparison between the daily disturbance forecast (red) and the actual power demand (blue).

Ultimately, assuming a real-time measure of thermal demand is available, the initial value $P_i^{load}(k = 1)$ is set to the actual power request, while the following values along the prediction horizon (2,..,N) are constrained to be equal to the disturbance prediction (see

Section 4.3.4).

4.3.4. Constraints

As anticipated, MPC has the benefit of including several constraints, both in the form of equalities and inequalities, in the finite-horizon control optimization problem. Indeed, all the optimization variables must comply with their physical/technical limits, or, in some cases, such as the initial states, they must assume a specific value. In addition, the system model must be embedded in the FHCOP through constraints replicating the GRU equations.

Constraints can be of two types. First, "hard" constraints must be always satisfied: for instance, the control variables cannot violate their bounds since they are a design choice (the result of the optimization procedure). On the contrary, variables like system outputs can be sometimes allowed to violate their boundaries because of the effects of disturbances. These constraints are implemented as "soft" by means of suitable slack variables [46].

In the AROMA network MPC problem, at time instant $t \in T$, the following state, control variable, output, disturbance and slack variables constraints are deployed.

State constraints: the states value is constrained to stay in between a minimum and a maximum (± 1, because of the neural network normalization requirements). In addition, at the beginning of the prediction horizon, the states must be equal to the value measured by the observer, whereas subsequently they must be set to the value predicted by the GRU non-linear equation, see equations (4.1).

$$\begin{aligned}
& x_{min}(k) \leq x_l(k) \leq x_{max}(k), \qquad \forall l \in \{1,..,n_{states}\}, \forall k \in \{1,..,N+1\} \\
& x_l(k=1) = x_{obs,l}(t), \qquad \forall l \in \{1,..,n_{states}\} \\
& x(k+1) = z(k) \circ x(k) + (1-z(k)) \circ \phi(W_r u(k) + U_r f(k) \circ x(k) + b_r), \\
& \qquad \forall k \in \{1,..,N\}
\end{aligned} \tag{4.1}$$

Control variable constraints: the boiler reference temperature, being the control variable, must be "hardly" constrained between its limits. In particular, its lower bound changes over time, i.e. in the daytime (from 7 a.m. to 7 p.m.) it is stricter than at night (see Figure 4.5). In fact, during the day the water use is greater than during night, hence a higher boiler temperature is required. In addition, the latter is allowed to modify its value from one control period to another at most by ± 5°C, since in real heating systems the temperature cannot abruptly change in an extensive way. However, it can never exceed

85°C, because of pipelines safety conditions, see equations (4.2).

$$\begin{aligned}
&T^{boiler}_{min,ref}(k) \leq T^{boiler}_{ref}(k) \leq T^{boiler}_{max,ref}(k), \qquad \forall k \in \{1,..,N\} \\
&\Delta_{min} T^{boiler}_{ref}(k) \leq T^{boiler}_{ref}(k) - T^{boiler}_{ref}(k-1) \leq \Delta_{max} T^{boiler}_{ref}(k), \qquad \forall k \in \{1,..,N\}
\end{aligned} \tag{4.2}$$

Output constraints: the load supply temperatures are bounded through soft constraints, since there may be some transient periods in which the latter are slightly violated but this non-compliance is of minor relevance. Moreover, the return temperature, overall mass flow rate and boiler power can "hardly" vary between their limits, which are however not too strict because of network design, hence slack variables are not needed. Similarly to the control variable, the lower bounds of the supply temperatures and of the overall return temperature change over time (see Figure 4.5) for a better functioning of the network. To sum up, only the supply temperatures are softly constrained given that the other outputs never violate their wide bounds ($n_{slack} = n_{load}$). Obviously, it could be possible to add slack variables to all output constraints, but in order to reduce the amount of optimization variables we confine the use of soft constraints only where strictly needed. In addition, the boiler power must always fulfil the physical equation that relates it to the mass flow rate and the difference between the supply temperature and the return one. Finally, all outputs are constrained to be equal to the value predicted by the neural network model, see equations (4.3).

$$\begin{aligned}
&T^{s}_{min}(k) - s_i \leq T^{s}_i(k) \leq T^{s}_{max}(k) + s_i, \qquad \forall i \in \{1,..,n_{load}\}, \forall k \in \{1,..,N\} \\
&T^{r}_{min}(k) \leq T^{r}(k) \leq T^{r}_{max}(k), \qquad \forall k \in \{1,..,N\} \\
&\dot{m}_{min,r}(k) \leq \dot{m}_r(k) \leq \dot{m}_{max,r}(k), \qquad \forall k \in \{1,..,N\} \\
&P^{boiler}_{min}(k) \leq P^{boiler}(k) \leq P^{boiler}_{max}(k), \qquad \forall k \in \{1,..,N\} \\
&P^{boiler}(k) = c_p \dot{m}_r(k)(T^{boiler}_{ref}(k) - T^{r}(k)), \qquad \forall k \in \{1,..,N\} \\
&y(k) = U_o x(k) + b_o, \qquad \forall k \in \{1,..,N\}
\end{aligned} \tag{4.3}$$

Disturbance constraints: as anticipated, at the beginning of the prediction horizon of each optimization step, the loads power is set to the actual demand at current time $t \in T$, assuming it is measurable, while the following values along N are constrained to be equal to the disturbance forecasting, see equations (4.4).

$$\begin{aligned}
&P^{load}_i(k=1) = P^{load}_{i,real}(t), \qquad \forall i \in \{1,..,n_{load}\} \\
&P^{load}_i(k+1) = \tilde{P}^{load}_i(\tau+1), \qquad \forall i \in \{1,..,n_{load}\}, \forall k \in \{1,..,N-1\}, \\
&\qquad\qquad \forall \tau \in \{t,..,t+N \cdot T_{sampling} - 1\}
\end{aligned} \tag{4.4}$$

Slack variables constraint: as explained above, there are only five slack variables, one for each load supply temperature. Because of their definition, they must always be non-negative, see equation (4.5).

$$s_m \geq 0, \qquad \forall m \in \{1, .., n_{slack}\} \tag{4.5}$$

4.3.5. Cost Function

Control performance, apart from the model accuracy, is also dependent on the local optimization problem, which can be solved by either analytical or numerical solutions [93]. The solution depends indeed on the structure of the cost function (J), which defines what are the objectives of the optimization problem. A correct choice of J is crucial: because of the presence of non-linear constraints, the optimization problem may get stuck in a local minimum, whereas an appropriate objective function increases the likelihood of ending up in a global solution [26]. As anticipated, we aim at keeping the problem as simple as possible, and thus we could synthesize its three goals as follows:

- the first objective is to minimize the electrical cost of the boiler, so as to make it operate in efficient conditions;
- the second objective is to discourage significant variations of the boiler temperature and of the furthest user's supply temperature from a reference value, i.e. $\bar{T} = 75°C$ (terminal cost). In this way, the controller avoids changes in the control action when not needed;
- the third objective is to minimize the slack variables, so that the soft constraints are violated only when strictly necessary.

Consequently, the complete cost function reads as

$$\begin{aligned} \min_{\Sigma(1),..,\Sigma(N)} J = \sum_{k=1}^{N} (\gamma_{electric}(k) \cdot P^{boiler}(k)) + \sum_{m=1}^{n_{slack}} (\gamma_{slack,m} \cdot s_m) + \\ + \gamma_u (T^{boiler}_{ref}(N) - \bar{T})^2 + \gamma_y (T^s_5(N) - \bar{T})^2 \end{aligned} \tag{4.6}$$

Specifically, P^{boiler} is expressed in kilowatt and multiplied by a time constant to get kilowatt-hour, whereas $\gamma_{electric}$ is the vector containing the electrical cost throughout the day (€/kWh). Typically, this price varies a lot depending on the time of the day and of year. For instance, because of the 2022 energy crisis, in Italy the electrical cost has increased up to peaks of 500% with respect to 2021. However, for this illustrative MPC problem we could simply use an average daily trend of such a cost, as depicted in Figure 4.3[3]. From the plot it is evident that during the most active times of the day, such as morning and evening, the price rises. In addition, it is interesting to notice that in southern regions where renewable energies are widely exploited the cost falls, for instance, in the afternoon because of the presence of solar energy.

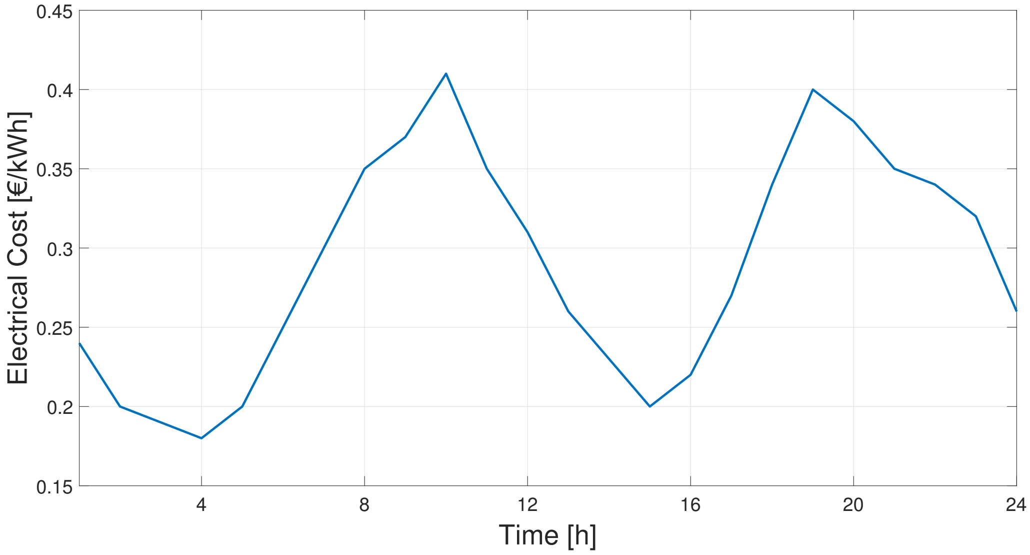


Figure 4.3: Daily electrical cost. Data extracted from an Italian average of January 2023.

Furthermore, γ_{slack}, γ_u and γ_y are tuning parameters that weigh the corresponding terms in J in order to assign them the correct importance in the cost function. They have been tuned experimentally and their values are reported in Table 4.2.

Finally, it is worth recalling that at time step k the solution of the optimization problem is $U(k|k) = [u(k|k)^T, ..., u(k+N-1|k)^T]^T$, where actually only its first element is applied to the system as control action, according to the Receding Horizon principle [82]. At the next time instant, the model is re-initialized in the observed state and the entire procedure is repeated [10].

4.3.6. Computational Effort Reduction

As anticipated, MPC is a complex yet efficient algorithm used to solve non-linear programming problems, in our case through the interior point search filter method. However, the computational cost, in particular when handling big systems such as thermal networks, is definitely expensive and the resolution may require several hours. Indeed, not only the

---

[3] `https://www.mercatoelettrico.org`

optimization problem must handle a lot of optimization variables, but the system dynamics constraints contain sigmoid and hyperbolic tangent functions, which are typically not really manageable.
In order to relieve such effort, some strategies can be adopted. For example, as already explained in Chapter 3, the PB-GRU states have been reduced from 89 to 54 (see Section 3.7). However, other tactics are embraced so as to lighten the computational burden.

4.3.6.1. Initialization

Since, as mentioned, Ipopt is a local solver, the computing time and solution optimality heavily depend on the initial conditions [61].
A simple but rather effective strategy to fasten a bit the non-linear optimization problem is to initialize at each step the optimization variables with the corresponding values predicted at the previous step (warm start). In particular, this is done for states, inputs, outputs and for the dual variables related to the constraints problem.

4.3.6.2. Parameters Settings

The first important parameter to choose when dealing with any MPC problem is the **prediction horizon** (N): it is the time window throughout which the system is simulated and the cost function is evaluated [10]. In general, it is suggested to set N to cover the settling time of the process. Moreover, it obviously depends on the adopted sampling period. However, the number of control inputs (and of optimization variables as well) is proportional to N, thus the problem can become very large and time consuming [46].
Concerning the case studied in this thesis, thermal networks are typically slow-transient systems and the longest settling time belongs to temperatures, with a value between two and three hours. Moreover, being the electrical cost highly time-varying, we aim at extending N so that the controller is able to foresee significant changes in such price.
To wrap up, considering the trade-off between problem complexity and prediction accuracy, a reasonable selection for the prediction horizon value could be six hours, i.e. 72 steps, with $T_{sampling} = 5$ min.

Another fundamental parameter is the **sampling time**, whose choice in a linear setting should be based on the Shannon theorem and on the (required) crossover bandwidth in closed-loop [46].
As described in Section 3.7, the sampling period used to collect system samples has been increased from one to five minutes, without altering the identification performance. This trick is particularly helpful when working with large MPC problems: in this way,

it is possible to raise the prediction horizon while still having a reasonable amount of optimization variables. For instance, if the desired N is 6 hours, when the sampling time is one minute the number of prediction steps is 360 (6h·60min/1min). On the other hand, when the sampling time is five minutes the number of prediction steps is 72 (6h·60min/5min). Clearly, this represents a major simplification in the optimization problem, having one fifth of optimization variables.

Finally, Ipopt allows to choose some useful settings, such as acceptable tolerances, maximum number of iterations or maximum CPU time. In detail, all these parameters are kept at their default value, whereas the maximum number of iterations is set to 10000 and the absolute tolerance on the complementarity conditions is fixed at 0.00001.

To sum up, all the main optimization parameters are reported in Table 4.2.

Symbol	Description	Value	SI unit
$\gamma_{electric}$	Electrical cost	Figure 4.3	[€/kWh]
γ_{slack}	Slack variables weight	$1 \cdot 10^3$	-
γ_u	Input weight	$1 \cdot 10^2$	-
γ_y	Output weight	$1 \cdot 10^2$	-
$\Delta_{max} T^{boiler}_{ref}$	Boiler temperature difference upper bound	+5	[°C]
$\Delta_{min} T^{boiler}_{ref}$	Boiler temperature difference lower bound	-5	[°C]
$\dot{m}_{max,r}$	Mass flow rate upper bound	25	[kg/s]
$\dot{m}_{min,r}$	Mass flow rate lower bound	2	[kg/s]
N	Prediction horizon steps	72	-
N_b	Blocking strategy steps	6	-
$n_{iterations}$	Number of iterations	10000	-
n_{load}	Amount of loads	5	-
n_{slack}	Amount of slack variables	5	-
n_{states}	Amount of states	54	-
P^{boiler}_{max}	Boiler power upper bound	$1 \cdot 10^7$	[W]
P^{boiler}_{min}	Boiler power lower bound	$1 \cdot 10^3$	[W]
T	Overall simulation time	86400	[s]
$T^{boiler}_{max,ref}$	Boiler temperature upper bound	85	[°C]
$T^{boiler}_{min,ref}$	Boiler temperature lower bound	$\{65, 70\}$	[°C]
T^s_{max}	Load supply temperatures upper bound	85	[°C]
T^s_{min}	Load supply temperatures lower bound	$\{65, 70\}$	[°C]
T^r_{max}	Return temperature upper bound	75	[°C]
T^r_{min}	Return temperature lower bound	$\{40, 45\}$	[°C]
$T_{sampling}$	Sampling time	300	[s]
x_{max}	State upper bound	+1	-
x_{min}	State lower bound	-1	-

Table 4.2: Main optimization parameters.

4.3.6.3. Blocking Strategy

An effective strategy to reduce the problem complexity is the so-called "blocking strategy".

Because of the aforementioned complexity-accuracy trade-off, we decided to set the sampling time to five minutes. As a consequence, the observer and the optimization algorithm should work every five minutes, simultaneously. However, it is well known that the temperature is characterized by a slow transient, thus it is not very cunning to change the control variable in such a short time. Therefore, if MPC is executed, for instance, every half an hour and accordingly the boiler reference temperature is kept constant in this period of time, the amount of control variables to optimize is much lower than if they were computed every five minutes.

To sum up, by fixing the control variable for thirty minutes ($N_b = 30\text{min}{\cdot}60\text{s}/300\text{s} = 6$ steps), a new constraint must be added:

$$\begin{gathered} T_{ref}^{boiler}(\kappa + (\xi - 1)N_b) = T_{ref}^{boiler}(\kappa + (\xi - 1)N_b + 1), \\ \forall \kappa \in \{1, .., N_b - 1\}, \forall \xi \in \{1, .., \frac{N}{N_b} - 1\} \end{gathered} \tag{4.7}$$

For a better understanding, an illustrative scheme to describe how the blocking strategy works is reported in Figure 4.4.

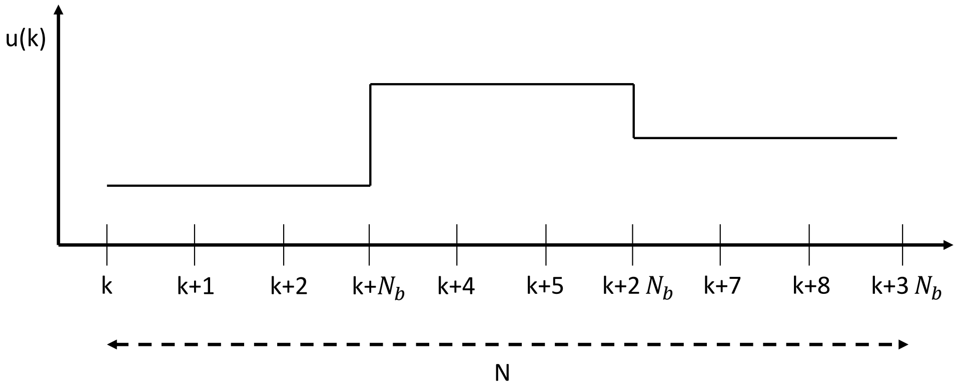


Figure 4.4: Schematic representation of the blocking strategy. N is the prediction horizon, N_b is the number of steps in which the control variable u(k) is blocked.

4.3.7. State Observer

As anticipated, the use of the GRU network for model predictive control purposes calls for the availability of a plant state estimate. Indeed, in a real case scenario, it is not possible to measure at each time instant a very large number of variables [61].

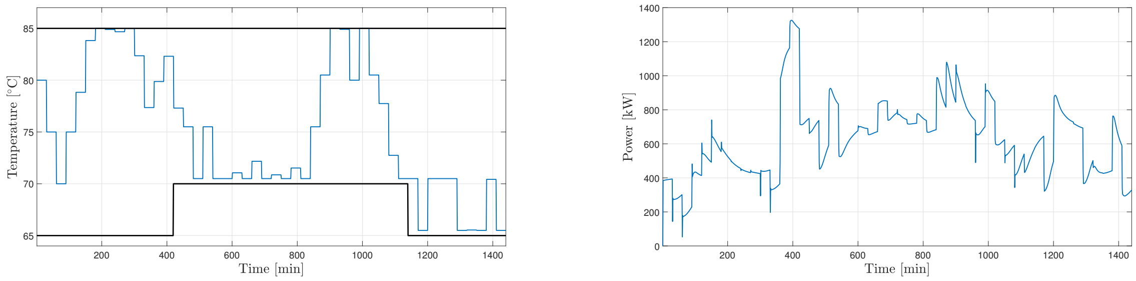
An observer is a dynamical system able to estimate state $\hat{x}$ and output $\hat{y}$ [9]. Many popular observers could be here implemented, such as Kalman Filter (KF), Extended Kalman Filter (EKF) or more simple ones replicating the system dynamics and having an innovation term. Typically, a suitable tuning of the state observer gain guaranteeing the convergence of the state estimate can be found solving the stationary Riccati equation, to learn more about it see [46].
However, since countless examples can be found in literature, such as [9–11], and the state observer complexity is out of the thesis scope, a plain open-loop observer replicating the model dynamics by means of the PB-GRU non-linear equations is appropriate. Recalling Section 2.5.2, the observer equations are given by (4.8), where actually only states are used by the MPC block as input. Ultimately, the underlying hypothesis that allows us to exploit an open-loop observer is the implicit stability property of the GRU model [10].

$$\begin{cases} \hat{x}_{k+1} = \hat{z}_k \circ \hat{x}_k + (1 - \hat{z}_k) \circ \phi(W_r u_k + U_r \hat{f}_k \circ \hat{x}_k + b_r) \\ \hat{y}_k = U_o \hat{x}_k + b_o \end{cases} \tag{4.8}$$

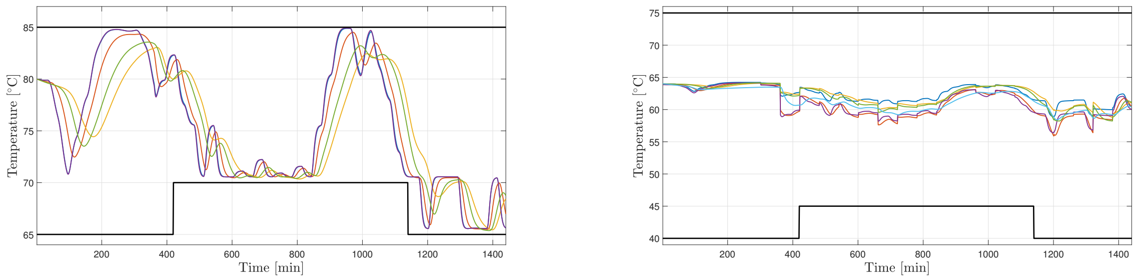
4.4. Optimization Results

Finally, the results of the above explained MPC algorithm can be examined. In detail, the system has been simulated for one day with a prediction horizon of six hours. Once again, the aim of the FHCOP is to optimally control the AROMA network through an MPC regulator so that the boiler electrical cost is minimized, while fulfilling technical constraints. This simulation, which exploited the PB-GRU equations both for the FHCOP and for the observer, required a resolution time of fifty-two minutes. The main variables behaviour is reported in Figure 4.5.

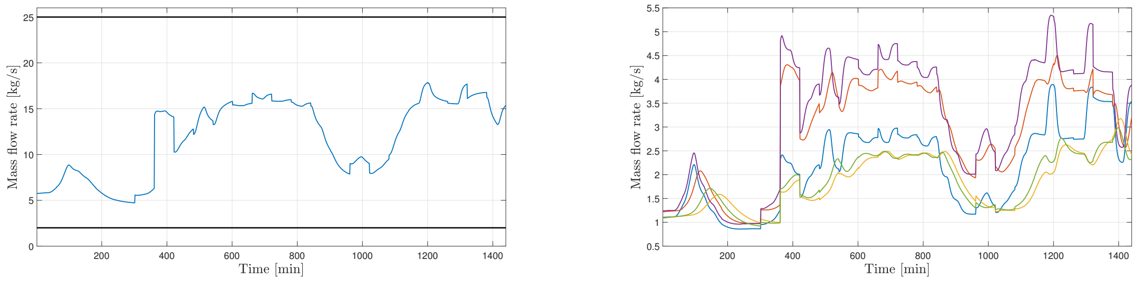
All computations are done in MATLAB R2022a using CasADi for automatic differentiation, through an Intel Core i7-1195G7 processor.



(a) Boiler reference temperature (b) Boiler power



(c) Load supply temperatures (d) Load return temperatures



(e) Overall mass flow rate (f) Load mass flow rates

Figure 4.5: PI-GRU-based MPC optimization results. When more variables are plotted in the same graph, the first user's variables are represented in blue, in purple the second's, in red the third's, in green the fourth's, in yellow the fifth's and in light-blue the overall return temperature (d). The constraints are depicted in black. The large boundaries of the boiler power (b) are not displayed to have a better close-up.

As visible from the previous plots, the boiler reference temperature never exceeds its bounds, correctly. In addition, because of the blocking strategy, it changes its value every thirty minutes. Then, it is clear that when the electric cost is expensive, the optimizer tends to minimize T_{ref}^{boiler} so that the difference between supply and return temperature is minimized in J. By contrast, when the price falls, MPC increases the boiler temperature, thanks to its predictive ability. In fact, it is able to forecast when it is more convenient to pre-charge the network, i.e. when $\gamma_{electric}$ is low, in view of subsequent periods where

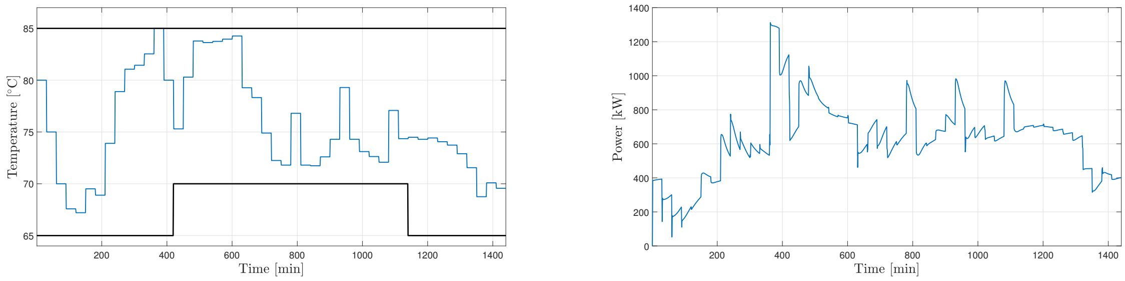
the cost will rise and hence the network will exploit the previously stored hot water.
As expected, the load supply temperatures follow the trend of the control variable. In particular, they never violate the limit, not making use of the slack variables.
As far as the return temperatures is concerned, they are always largely complying with the boundaries (hard constraints), having a pretty limited dynamics.
Similarly, the overall mass flow rate never exceeds its bounds and, correctly, the trend of the single mass flow rates follows the loads power demand.
Finally, both the boiler temperature and its power are coherent with the overall network power request (see Figure 4.2), with a peak of roughly 1.3 MW of P^{boiler}.
In the end, MPC manages to accurately achieve the cost function objectives, in a reasonable amount of time.

Lastly, it is interesting to notice how the optimization performance changes depending on the model adopted. For instance, if instead of using the 54-state PB-GRU one exploits, both in the observer and in the FHCOP, the 54-state standard GRU, the plots reported in Figure 4.6 are obtained.

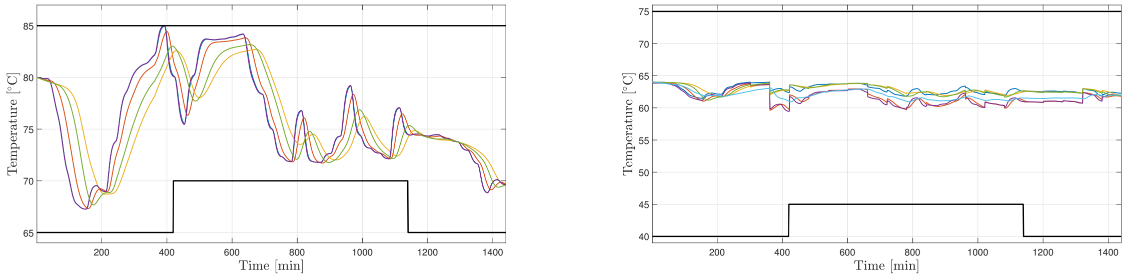
It is worth highlighting that the MPC that makes use of the standard GRU model takes more time than the one exploiting PB-RNNs. Indeed, this latter model is more accurate and hence the predictions are more reliable and precise, resulting in a faster execution. The average and maximum resolution time per iteration, together with the overall resolution time, are reported in Table 4.3. Another interesting point regards the cost function. By comparing the value of the boiler electrical cost obtained through the PI-GRU-based MPC and the one obtained through the standard GRU-based MPC, we could notice that the latter takes a higher value (see Table 4.3). In other words, a more precise model results in cost savings too.

Model	**Cost (€)**	**Average time/iter**	**Maximum time/iter**	**Total time**
PB-GRU	4378,8	1'6"	5'35"	52'
Standard GRU	4534,8	1'28"	9'27"	1h11"

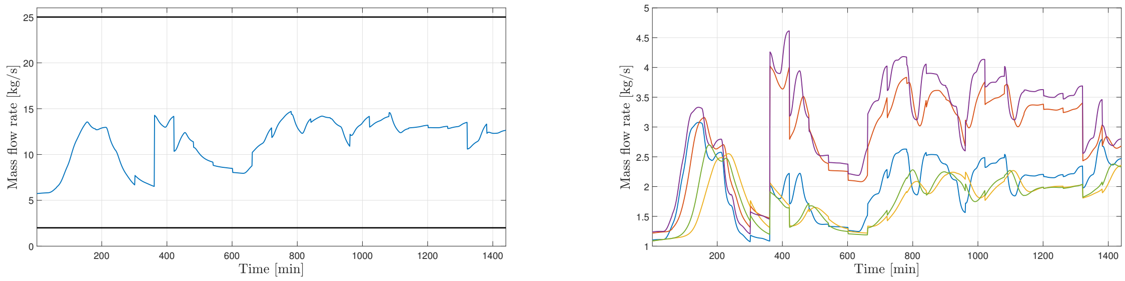
Table 4.3: Comparison between the performance of PI-RNN-based MPC and standard RNN-based MPC. The electrical cost, the average and maximum resolution time per iteration, together with the total resolution time, are reported for the two models.



(a) Boiler reference temperature (b) Boiler power



(c) Load supply temperatures (d) Load return temperatures

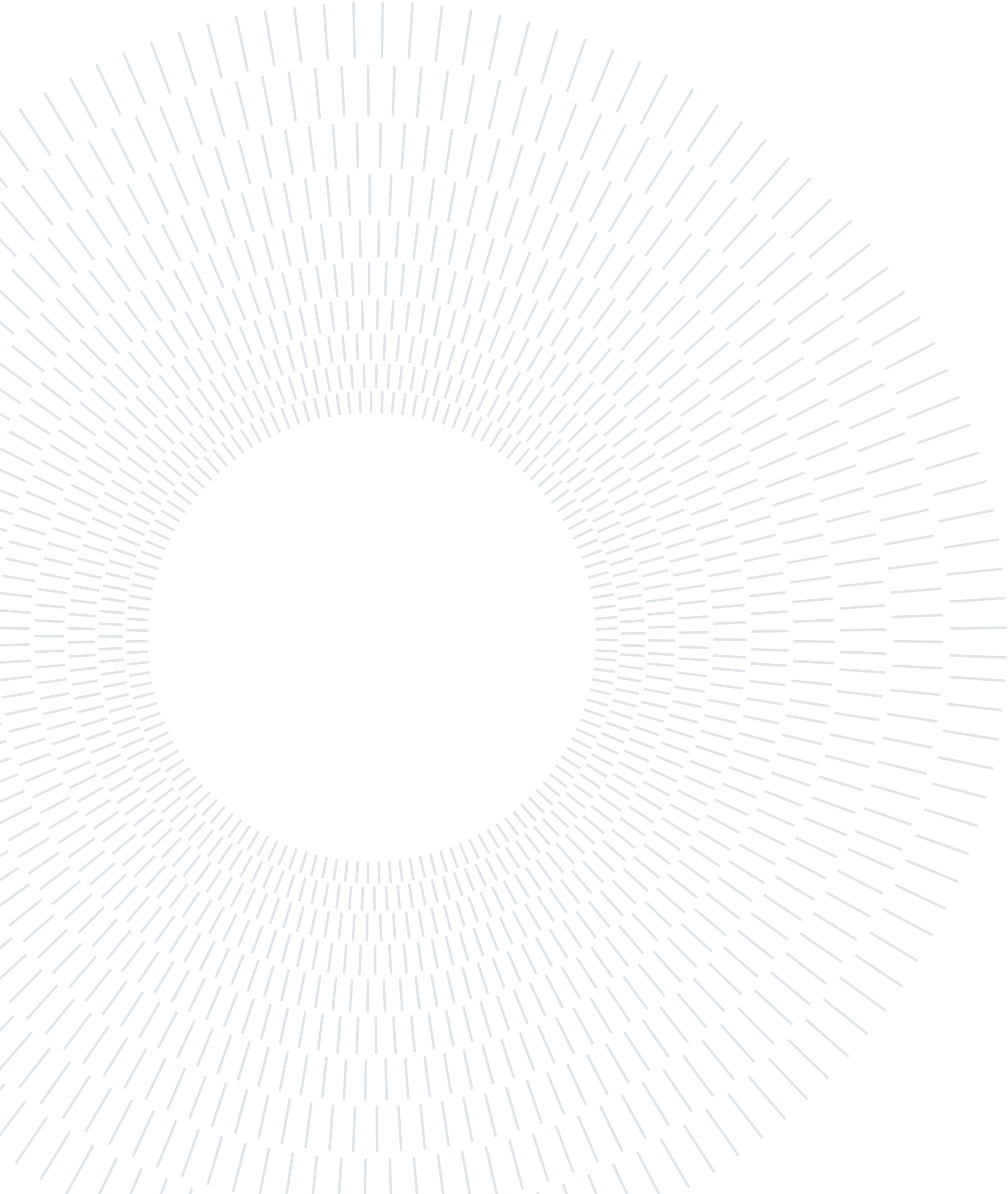


(e) Overall mass flow rate (f) Load mass flow rates

Figure 4.6: Standard GRU-based MPC optimization results. When more variables are plotted in the same graph, the first user's variables are represented in blue, in purple the second's, in red the third's, in green the fourth's, in yellow the fifth's and in light-blue the overall return temperature (d). The constraints are depicted in black.

4.5. Conclusions

In this chapter a well-established control technique has been implemented, namely model predictive control. However, differently from standard MPC algorithms, it was used a model identified through physics-based recurrent neural networks both in the formulation of the finite horizon control optimization problem and in the observer design. In particular, as expected, a more accurate model yields better optimization and control results: faster execution, less constraints violation and cost savings.



5 | Lifelong Learning

5.1. Chapter Overview

This chapter focuses on the long-term monitoring of the district heating system under investigation (AROMA network). After pointing out the motivations and objectives behind lifelong learning, a brief literature review on the topic is explored. Afterwards, an aggregate algorithm for the supervision of a DHS is proposed. The strategy actually develops into two branches, depending on which type of change has been detected: typically, it can be either a structural modification or an operating conditions shift. For each scenario, the corresponding solving algorithm and identification results are reported.

5.2. Lifelong Learning Overview

Humans and animals have the ability to continually acquire and fine-tune knowledge throughout their lifespan, thanks to a sophisticated set of neurocognitive mechanisms that collectively contribute to the development and specialization of our sensorimotor skills as well as to long-term memory consolidation. Lifelong learning is thus defined as the ability to continually learn over time by accommodating new knowledge while preserving previously learned experiences. Similarly to nature, lifelong learning capabilities are crucial for computational systems and autonomous agents interacting in the real world and processing continuous streams of information [63].

In machine learning indeed, after the training procedure is completed, the agent's knowledge is fixed and unchanging. Consequently, if the agent has to be applied to a different task it must be re-trained (fully or partially), again requiring a very large number of new training examples. By contrast, biological agents exhibit a remarkable ability to learn quickly and effectively from ongoing experience [57]: their natural adaptive skill is object of growing interest in the machine learning community.

5.2.1. Motivations and Objectives

In the long run, it is very likely that throughout the lifespan of a system various changes occur, such as structural adjustments or new operating conditions. As a consequence, the system model initially employed does no longer constitute a precise description of the plant. Thereby, it comes the need to adapt the original model to changes, while still preserving previously gained knowledge.

Lifelong learning is a well-established challenge for machine learning since the continual acquisition of incrementally available information from non-stationary data distributions generally leads to the so-called "catastrophic forgetting" or "interference", i.e. training a model with new information may interfere with previously learned knowledge [63]. Actually, once new data are available, there is the risk that the latter completely overwrite long-standing information. In order to overcome catastrophic forgetting, the neural network must, on the one hand, show the ability to acquire continuous information through which refining existing knowledge and, on the other hand, prevent the new input from significantly interfering with past data [63]. In literature, this is called the "stability–plasticity dilemma". The basic idea is that an artificial intelligence system needs plasticity to integrate new information, but also stability to prevent forgetting previous knowledge [15].

Ultimately, the lifelong learning problem is particularly relevant when the model is used to synthesize a model-based control law. Indeed, adaptation would typically be required to preserve the closed-loop stability characteristics as well as to maintain performance throughout the plant lifespan [12]. In fact, the ability to automatically fit variations in plant dynamics and environment has made adaptive controllers increasingly important for various applications [31].

5.2.2. Plasticity and Lifelong Learning in Literature

Lifelong learning is a matter of great interest in literature: almost every system whose lifespan is significantly long must be constantly monitored so as to update, if necessary, its model and to maintain an efficient control strategy.

For detailed reviews, the reader is referred to [63, 76]. In [76] Soltoggio et al. mainly describe the implementation and use of Evolved Plastic Artificial Neural Networks (EPANNs), which include both innate properties and the ability to change and learn in response to experiences in different environments and problem domains. This type of networks is inspired by a large variety of ideas from biology: plasticity is indeed an essential feature of the brain for neural malleability at the level of cells and circuits [63]. Typically, in EPANNs, learning rules are functions that change the connection weight w between two neurons. An interesting example of plastic neural networks can be found in [57, 83] where Miconi et al. mostly make use of the so-called Hebbian plasticity rule. In practice, a connection between any two neurons i and j has both a fixed and a plastic component. The fixed part is merely a traditional connection weight $w_{i,j}$, whereas the plastic part is stored in a Hebbian trace $\text{Hebb}_{i,j}$, which varies during a lifetime according to ongoing inputs and outputs. Indeed, plasticity, just like connection weights, can be optimized by gradient descent in large recurrent networks with Hebbian plastic connections. An important aspect of differentiable plasticity is its extreme ease of implementation, requiring only a few additional lines of code on top of a standard network implementation. Other examples of plasticity rules are explained in [72], such as the Oja's rule and the ABCD plasticity. To sum up, the basic idea behind plasticity is to include in the problem formulation a novel loss function so as to update the neural network weights, as discussed in detail in [15, 95].

As far as control-oriented lifelong learning is concerned, a detailed review is presented in [32], where Hewing et al., apart from describing a number of models used in literature for control purposes, the controller design and the safety issue, show various learning-based MPC schemes that aim at estimating the model uncertainty set directly from data, potentially adjusting it over time to reduce conservatism. In a nutshell, these techniques make use of an explicit distinction between a nominal system model and an additive learned term accommodating uncertainty.

Moreover, a further significant example of lifelong learning of recurrent neural networks for control design is represented by [12], where it is taken into account the case in which the plant model has constant or slowly-varying parameters. The authors propose an approach based on the Moving Horizon Estimation (MHE), i.e. an optimization-based strategy widely investigated and used by the control community.

In addition, in [5] Bemporad investigates the use of extended Kalman filtering to train recurrent neural networks with rather general convex loss functions and regularization terms on the network parameters.

Another approach used for model predictive control purposes is the one proposed in [31], where the adaptive neural network mechanism is used to compensate the steady-state error due to parameters variations and to update the weights of the neural network model online.

Additionally, some interesting examples of error-triggered online model identification can be found in [2, 90]. In practice, when the prediction error exceeds a pre-specified threshold,

a new identification is performed by using the most recently generated input/output data and by initializing the new weights with their previous values. This online identification can be used to update an empirical model when significant plant-model mismatch is detected because the region of operation shifts and the current model no longer captures the non-linear dynamics. The just mentioned method reveals both an advantage and a disadvantage: the computation time for updating an RNN model is significantly reduced due to the small size of the newly collected dataset. However, because of insufficient data in the new training dataset, the updated RNN models are not guaranteed to approximate non-linear dynamics subject to disturbances in the entire operating region. A similar issue and approach is discussed in [94], where the authors employ an error-triggered mechanism to activate the quantification of modelling uncertainties and to update the physics-informed recurrent neural network models accordingly.
Finally, a noteworthy example of continual learning that makes a distinction among three different scenarios, namely task-incremental, domain-incremental and class-incremental learning, is discussed in [85].

5.3. A Lifelong Learning Algorithm for the Supervision of a DHS

Now that the basic concepts of lifelong learning have been explored, it is possible to focus on the specific case study analysed in this thesis. The AROMA network is indeed a simplistic but rather effective example on which a number of meaningful experiments can be conducted.

In general, after an initial phase of model identification (Chapter 2 and 3) and control design tuning (Chapter 4), a district heating system is ready to work in what we may call "normal" or "nominal" conditions for a more or less extended period of time. However, in the long run (after some months or years), it is very likely that the system undergoes some changes. For instance, there may be some structural adjustments such that on pipe length or insulation, or the plant working conditions may be modified in accordance to the period of the year.

By this time, it should be clear that the initial model on which the control design was based is no longer able to correctly capture the plant dynamics and hence some adjustments must be deployed.

5.3.1. General Algorithm Description

In this section we propose a novel methodological algorithm to deal with model uncertainties. As anticipated, in the longer term, a thermo-hydraulic system such as the AROMA network can undergo mainly two types of changes. First, for instance, there may be a structural modification in the pipe length, diameter, insulation thickness, thermal conductivity, or even there may be an alteration in the number of users, in the differential pressure of the pump and of the expansion vessel. This scenario, intuitively, will result in a drastic system model modification. Second, a DHS could also exhibit a change in the operating conditions with respect to the nominal ones. Clearly, if the training procedure of the original model has been performed appropriately, the training dataset should include a large variety of working conditions, covering all the possible periods of the year. However, it may occur, for instance, that a winter is abnormally cold, i.e. the power consumption would be particularly high and out of the original training set range.

These considerations lead us to develop an algorithm through which it is possible to constantly monitor the AROMA network status and to make the proper adjustments depending on the encountered scenario.
For the sake of clarity, we might call the new data available, for instance at the end of a closed-loop daily operation, as "current data", so as to distinguish them from the "training data" originally employed to identify the primary system model.

The first step is to compute, at the end of a fixed monitoring period (e.g. day or week) and for each system output variable, the root mean square error ($RMSE$) between the measured outputs and the ones predicted by the physics-based gated recurrent unit model. If the $RMSE$ associated to each variable is lower than its corresponding threshold then the original PB-GRU model is still able to capture the system dynamics: no modification is required. By contrast, if at least one $RMSE$ exceeds that threshold, this is an alarming signal indicating that something has changed.
In particular, the $RMSE$ threshold of each variable is computed as follows. Let us suppose that we collect a weekly set of data in normal conditions, i.e. at the beginning of the system functioning where for sure neither a plant change nor a working status shift occurred. For each output variable it is necessary to compute the error committed by the existing PB-GRU model, i.e. the difference between the predicted output and the corresponding measured value. By considering the error values over time and by plotting the associated histogram, it is possible to notice that they typically have a Gaussian distribution (see Figure 5.1).

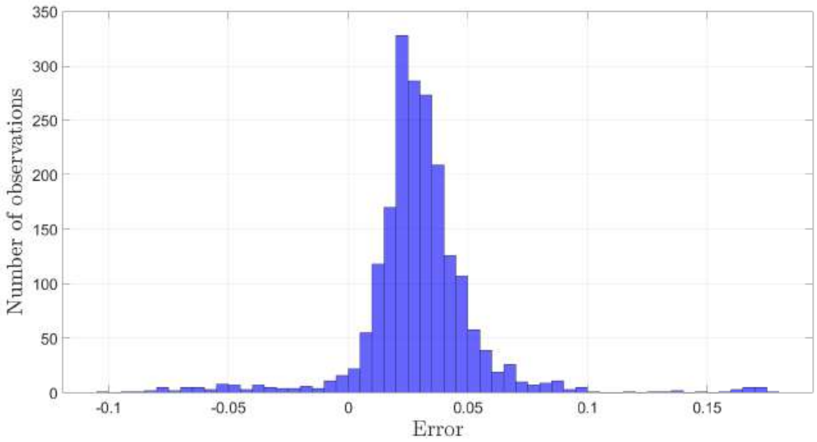


Figure 5.1: Example of the normal distribution of a system variable error: T_5^s plant-model mismatch. The number of observations is 2000, i.e. around seven days sampled every five minutes.

Therefore, for each variable, we find out the $RMSE$ threshold by computing the 90^{th} percentile of the normal distribution, i.e. the error value e where the probability of being less than e is 90%. In fact, a percentile is defined as the value in a normal distribution that has a specified percentage of observations below it. In other words, this means finding out the error value above which only the 10% of observations is distributed (histogram tails).

At this point, as soon as the $RMSE$ computation detects an anomaly, we need to figure out its cause. In order to distinguish whether a change in the plant or in the operating conditions occurred, we can exploit the so-called Mahalanobis distance (T^2). Thanks to its whitening characteristics, it is an extremely valuable measure in multivariate data analysis and has many applications including cluster analysis, outlier detection, text classification, Bayesian inference and image processing [62]. The basic idea is to verify if the current input set is "statistically close" to the input training set. In a nutshell, if this is the case, then the operating conditions are included in the original training range and hence a modification in the plant took place. By contrast, if the current input dataset is "statistically far" from the input training set, then the working status has changed.

Firstly, it is necessary to normalize each variable contained in the current and in the training dataset. In particular, for each input and output variable, the mean and the variance are computed using the training (or benchmark) dataset. These mean and variance values are therefore used to normalize both the variables contained in the training set and the ones included in the current dataset.

Secondly, considering the training set defined by the matrix X of normalized data and with covariance matrix S, it is possible to define the statistical (or Mahalanobis) distance T^2 of the vector of measurements x from the training set as $T^2 = x'S^{-1}x$. In depth, since the expected mean and variance are estimated from data, the ellipsoidal confidence region is defined by $T^2 \leq T^2_\alpha = \frac{m(n-1)}{(n-m)}F_\alpha(m, n-m)$, where α is the significance level, m is the number of independent Gaussian random variables x_i with null expected value and unitary variance, n is the number of observations and $F_\alpha(m, n-m)$ is the upper $100\alpha\%$ of the F distribution with m and $n-m$ degrees of freedom.

In conclusion, we can say that if the Mahalanobis distance between the current inputs and the training ones exceeds its threshold, then the actual operating conditions are statistically far from the original ones. In the opposite case, there has been a change in the plant structure as the input values are within the training set boundaries (see Figure 5.2). To this end, it should be evident that in order to compute the Mahalanobis distance it is necessary to divide inputs (u) and outputs (y) and then to take the average of the distances related to the different variables in the two categories. In this way, $T^2(u)$ is a clear indicator of a change of operating conditions: a new working status derives from a different power consumption whose information is contained in u (or, in our case study, in P_i^{load}). Instead, $T^2(y)$ can be seen as a simple indicator that the $RMSE$ computation is correct: both the calculation of $RMSE$ and of $T^2(y)$ is performed on the output variables and thus they must be coherent.

More details concerning the statistical concepts discussed are thoroughly parsed in [30, 58].

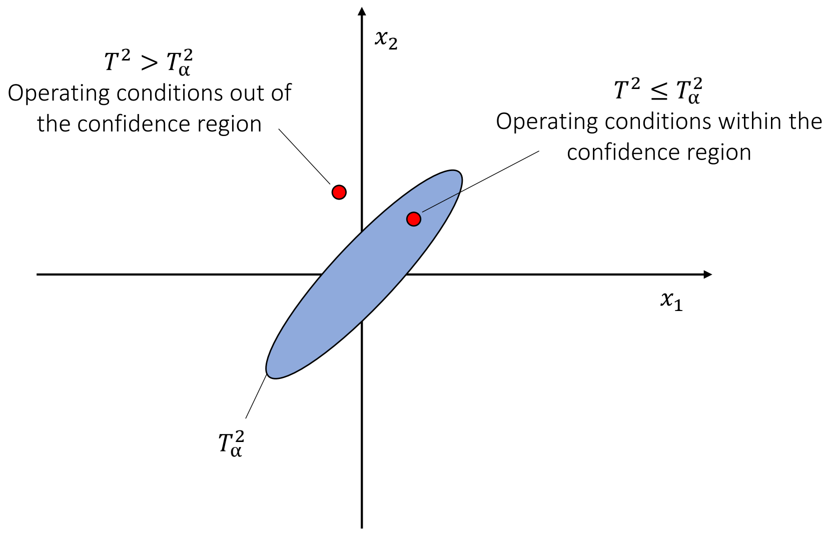

x_2
$T^2 > T^2_\alpha$
Operating conditions out of the confidence region
$T^2 \leq T^2_\alpha$
Operating conditions within the confidence region
x_1
T^2_α

Figure 5.2: Schematic representation of the ellipsoidal confidence region of T^2 in case of two variables.

Ultimately, thanks to this easy computation we can comprehend what kind of modification occurred. At this stage, one should employ a different strategy according to the type of scenario. First, if an alteration in the plant has taken place, then it is necessary to get a novel model able to capture the new dynamics (see Section 5.3.2). Conversely, if we find out that the working conditions have dramatically changed with respect to the nominal ones, then it is convenient to add this new information to the existing model (see Section 5.3.3).

To sum up, the overall procedure is reported in Algorithm 5.1.

Algorithm 5.1 Lifelong Learning Algorithm for the AROMA network

```
Compute $RMSE_i \quad \forall i \in \{1,..,n_y\}$
if $RMSE_i \leq \overline{RMSE_i} \quad \forall i \in \{1,..,n_y\}$ then
    do nothing
else if $\exists i \in \{1,..,n_y\} | RMSE_i > \overline{RMSE_i}$ then
    compute Mahalanobis distances
    if $(T^2(u) \leq T^2_\alpha(u)) \wedge (T^2(y) > T^2_\alpha(y))$ then
        change in the plant: Moving Horizon Estimation
    else if $(T^2(u) > T^2_\alpha(u)) \wedge (T^2(y) > T^2_\alpha(y))$ then
        change in the operating conditions: additive uncertainty identification
    end if
end if
```

For a better grasp, before entering into the details of each scenario, let us apply the aforementioned algorithm to our case study by means of a numerical example.

First, it is necessary to compute the triggering thresholds. To this end, at the end of a weekly closed-loop operation in nominal working conditions and without plant alterations, we collect data and we compute, for each output variable, the $RMSE$ between the normalized measured system outputs and the ones predicted by the original PB-GRU having 54 states (nominal $RMSE$). It is worth recalling that the AROMA network variables we are interested in are twenty-three: six inputs ($m_u = 6$), that is, the boiler temperature and the five power consumptions, and seventeen outputs ($m_y = 17$), i.e. T_1^s, T_1^r, $\dot{m}_1$, T_2^s, T_2^r, $\dot{m}_2$, T_3^s, T_3^r, $\dot{m}_3$, T_4^s, T_4^r, $\dot{m}_4$, T_5^s, T_5^r, $\dot{m}_5$, T^r, $\dot{m}_r$.
Starting from the nominal $RMSE$ values, we calculate, again for each output variable, the associated thresholds by means of the 90^{th} percentile. The so-obtained thresholds vector is $\overline{RMSE} = [0.0351, 0.0430, 0.0545, 0.0385, 0.0905, 0.1012, 0.0419, 0.0719, 0.0939, 0.0580, 0.0334, 0.0971, 0.0594, 0.0268, 0.0746, 0.0505, 0.1583]$.

Furthermore, as far as the statistical distance is concerned, by consulting the F distribution table, we can find out the Mahalanobis distance triggering thresholds, always distinguishing between input and output. It is reasonable to select a 90% confidence interval, i.e. $\alpha = 0.1$:

$$\begin{aligned} T^2_{\alpha=0.1}(u) &= 10.8388 \\ T^2_{\alpha=0.1}(y) &= 26.2190 \end{aligned} \tag{5.1}$$

Now that the required thresholds have been computed, at the end of each monitoring period (e.g at the end of the day) we should verify whether the $RMSE$ thresholds are exceeded or not. If this is the case, we therefore have to normalize the variables contained both in the current daily dataset and in the original training set by using the mean and the variance computed through the benchmark dataset variables. Once again, in order to compute the Mahalanobis distances we shall split inputs and outputs and then take the average of the distances related to the different variables in the two groups.

A remark: the computation of T^2 requires a much larger training (or benchmark) set than the current dataset which we need to statistically evaluate. Indeed, the training set used for this computation is made by 15200 samples (around 55 days of simulation sampled every five minutes, see Chapter 3). By contrast, if a single daily operation is considered, the current available data are 288 (again with a sampling time of five minutes).

Let us perform the just mentioned calculations in three different scenarios, over a daily dataset:

- case 1: original plant and nominal operating conditions;
- case 2: modified plant and nominal operating conditions;
- case 3: original plant and operating conditions out of the training set.

In detail, the original plant is the one discussed in Chapter 1, whereas in the modified plant a change is performed: the first pipe length is increased by 20% with respect to the original one and the diameter is decreased by 20% with respect to its initial value (see Section 5.3.2). Moreover, the nominal operating condition is represented by the AROMA network usual power consumption (see Figure 1.10b). Conversely, in order to detect an unusual working status, load profiles out of the original training set ranging from 30kW to 420kW (see Figure 2.2b) are taken into account (see Section 5.3.3).

The values of the seventeen prediction $RMSE$ are reported in Table 5.1, distinguishing among the three different aforementioned scenarios. In addition, to make a clearer comparison, the $RMSE$ thresholds are listed too. Additionally, in Table 5.2 the values of

Mahalanobis distances both for input and for output are reported.

	Threshold	**Case 1**	**Case 2**	**Case 3**
$\boldsymbol{T_1^s}$	0.0351	0.0209	0.0177	0.0247
$\boldsymbol{T_1^r}$	0.0430	0.0157	0.1692	0.0227
$\boldsymbol{\dot{m}_1}$	0.0545	0.0218	0.1021	0.0331
$\boldsymbol{T_2^s}$	0.0385	0.0205	0.0166	0.0579
$\boldsymbol{T_2^r}$	0.0905	0.0244	0.2632	0.1033
$\boldsymbol{\dot{m}_2}$	0.1012	0.0315	0.1397	0.0432
$\boldsymbol{T_3^s}$	0.0419	0.0304	0.0295	0.0315
$\boldsymbol{T_3^r}$	0.0719	0.0174	0.2333	0.1984
$\boldsymbol{\dot{m}_3}$	0.0939	0.0269	0.2118	0.0627
$\boldsymbol{T_4^s}$	0.0580	0.0448	0.0617	0.0527
$\boldsymbol{T_4^r}$	0.0334	0.0159	0.1031	0.0567
$\boldsymbol{\dot{m}_4}$	0.0971	0.0573	0.2579	0.0778
$\boldsymbol{T_5^s}$	0.0594	0.0464	0.0660	0.0583
$\boldsymbol{T_5^r}$	0.0268	0.0107	0.0846	0.0490
$\boldsymbol{\dot{m}_5}$	0.0746	0.0443	0.2111	0.0720
$\boldsymbol{T^r}$	0.0505	0.0381	0.2149	0.1088
$\boldsymbol{\dot{m}_r}$	0.1583	0.0564	0.3203	0.0630

Table 5.1: Comparison of $RMSE$ in three different cases for each output variable. The thresholds are highlighted in light-blue, whereas the values exceeding the corresponding thresholds are highlighted in red.

	Threshold	**Case 1**	**Case 2**	**Case 3**
$\boldsymbol{T^2(u)}$	10.8388	3.5477	3.5477	23.4386
$\boldsymbol{T^2(y)}$	26.2190	14.5386	37.7028	161.8426

Table 5.2: Comparison of Mahalanobis distances in three different cases. The thresholds are highlighted in light-blue, whereas the values exceeding the corresponding thresholds are highlighted in red.

We can finally explain the three case studies reported in the previous tables more specifically:

- case 1: since none of the seventeen output variables exceeds its corresponding $RMSE$ threshold, the algorithm is not triggered. Accordingly, the Mahalanobis distance thresholds are not crossed.
- Case 2: since fourteen $RMSE$ thresholds are exceeded, it is necessary to compute the Mahalanobis distance as well. In particular, being $T^2(u) \leq T^2_\alpha(u)$ and $T^2(y) > T^2_\alpha(y)$, the current inputs are within the original training range whereas the outputs are outside. In conclusion, there has been a modification in the plant.
- Case 3: since six $RMSE$ thresholds are exceeded, it is necessary to compute the Mahalanobis distance as well. In particular, being $T^2(u) > T^2_\alpha(u)$ and $T^2(y) > T^2_\alpha(y)$, the current inputs and outputs are out of the original training range. In conclusion, there has been a modification in the operating conditions.

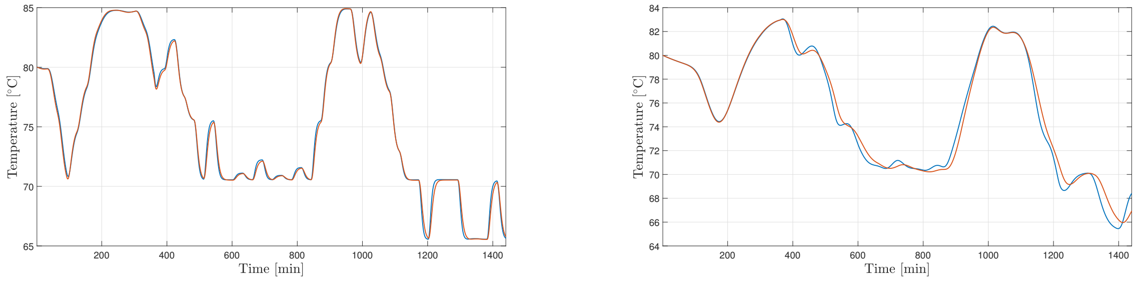
In the end, after figuring out in which case lies the actual system, it is possible to proceed following a different strategy depending on the system status. The clear separation between the two scenarios is convenient for studying their different challenges and possible solutions in isolation.

5.3.2. Plant Change Scenario

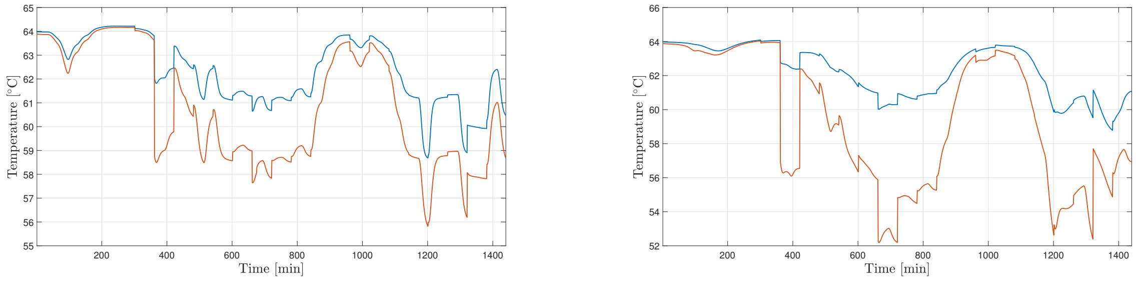
This section aims at giving an overview on how to proceed in case of plant change. As anticipated, it is highly unlikely that, after some months or years of functioning, a DHS, and, in general, every large thermo-hydraulic system, does not go through any adjustment. For instance, because of the ongoing cities growth, it may be possible that some new users are added to an existing district heating system. To this end, new pipes must be inserted, resulting in an overall pipeline longer than the original one. Likewise, because of structural reasons, there may be a modification in pipes diameter, insulation thickness or thermal conductivity, or even there may be an alteration in the differential pressure of the pump or of the expansion vessel.

As can be seen, these constructional alterations lead to a radical system modification. As a consequence, the original model that has been formerly identified through a black-box technique is no longer appropriate to describe the new plant dynamics and it must be re-tuned.

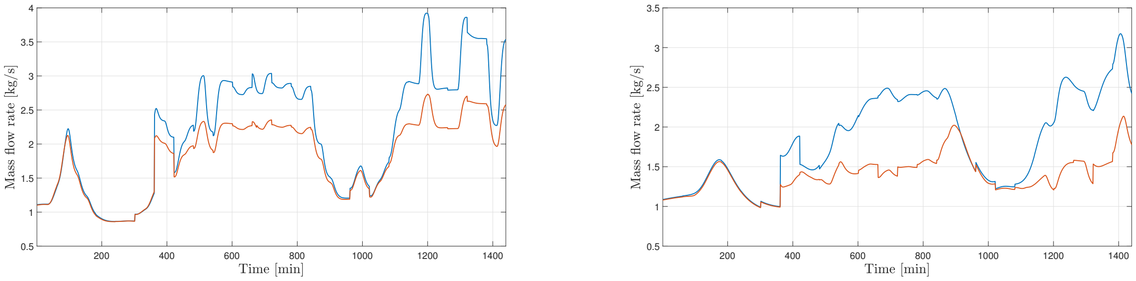
Before introducing how to solve the just mentioned problem, we may visualize how the dynamics of the system changes depending on its physical parameters. Let us consider the case study analysed in this thesis, i.e. the AROMA network. A reasonable and effective change could be to extend the first (both supply and return) pipe length by 20% with respect to its original value and to reduce the same pipe diameter by 20%. In this way, the overall system dynamics should be severely modified: an extension in the first pipe length and a shrinkage in the diameter should result in a slowed down system. This is clearly evident in Figure 5.3, where, for the sake of brevity, it is reported only the behaviour of the closest (first) and of the furthest (fifth) load variables (supply, return temperature and mass flow rate), together with the overall return temperature and mass flow rate, both in the case of old and new plant. In this daily closed-loop simulation, the control variable is, as usual, the boiler temperature computed through the model predictive control algorithm (see Chapter 4).



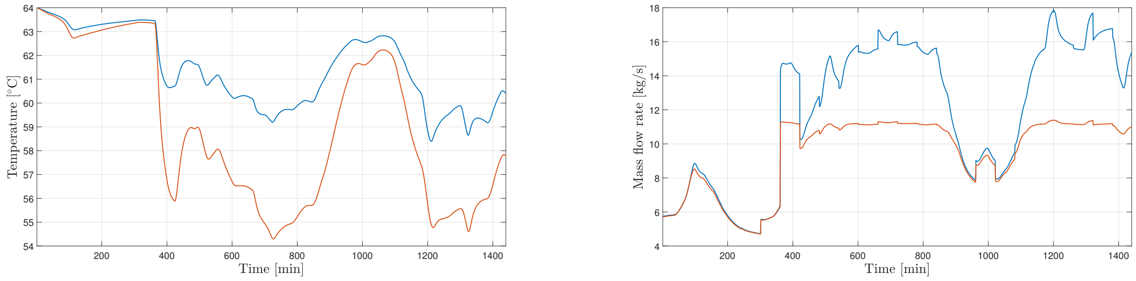
(a) First user supply temperature (b) Fifth user supply temperature



(c) First user return temperature (d) Fifth user return temperature



(e) First user mass flow rate (f) Fifth user mass flow rate



(g) Return temperature (h) Return mass flow rate

Figure 5.3: Comparison between some AROMA network variables in the case of old plant (blue) and new plant (red).

5.3.2.1. Moving Horizon Estimation

In order to capture the new plant dynamics, we exploit an adaptation algorithm inspired by Moving Horizon Estimators (MHE), as carefully described in [12].
Even though the system dynamics changes, we would like to preserve to some extent the information contained in the original model, so as not to completely throw away the results achieved so far. In fact, as already discussed, re-tuning the model when new data are available can lead to the catastrophic forgetting problem, which occurs when the knowledge previously embedded in the original model is discarded by the re-tuning procedure itself, with consequent reduced description capabilities [12].

Let us consider a generic Recurrent Neural Network model in the form

$$\begin{cases} x_{k+1} = f(x_k, u_k; \Theta) \\ y_k = g(x_k, u_k; \Theta) \end{cases} \tag{5.2}$$

where k is the discrete time index, Θ is the vector of network parameters (weights and biases) and x, u and y are, respectively, the state, input and output vectors.
The proposed algorithm is intended to be run periodically every N steps, where N is a positive integer corresponding to the length of the time-window throughout which data are collected from the system. For instance, if at the end of a daily operation and subsequent data gathering the triggering conditions are activated, the algorithm is run: $N = 24\text{h}\cdot 60\text{min}/5\text{min} = 288$, with $T_{sampling} = 5$ min. Therefore, we can periodically formulate an optimization problem which seeks the parameters update that best explains the new collected data. As a consequence, the model weights constitute an optimization variable, denoted by $\hat{\Theta}$. The underlying MHE optimization problem, at the current time instant, can be stated as

$$\begin{aligned} \hat{\Theta}^* = \arg\min_{\hat{\Theta}, \hat{x}_0} & \{J = \sum_{k=0}^{N} ||y_k - \hat{y}_k||^2 + \mu||\hat{\Theta} - \bar{\Theta}||^2\} \\ \text{s.t.} \qquad & \hat{x}_0 = \bar{x}_0 \\ & \hat{x}_{k+1} = f(\hat{x}_k, u_k; \hat{\Theta}) \\ & \hat{y}_k = g(\hat{x}_k, u_k; \hat{\Theta}) \\ & \forall k \in \{0, .., N\} \end{aligned} \tag{5.3}$$

where $\hat{\Theta}^*$ denotes the optimal solution to (5.3), whereas $\bar{\Theta}$ represents the vector of original model weights. Clearly, the underlying assumption is that the network parameters are constant over time. The convergence properties of the proposed MHE algorithm are

proved in [12].
A few relevant considerations regarding this optimization problem are here listed:

- the cost function penalizes the mismatch between the actual measured output y_k and that of the new GRU model. By contrast, the second term of J discourages significant deviations from the previously computed optimal solution $\bar{\Theta}$. Hence, the coefficient μ defines the trade-off between the need to improve the model performance (μ small) and the necessity not to forget the information previously acquired, i.e. to avoid catastrophic forgetting. In particular, in our example we can set $\mu = 0.1$.

- The optimal solution $\hat{\Theta}^*$ represents the updated set of weights of the model (5.2). Since the PB-GRU model is made of a large number of parameters, being characterized by fifty-four states and six layers, we adopt the following strategy. The optimization variable $\hat{\Theta}$ does not contain all the weights and biases characterizing the neural network, but rather only the ones related to the output layer of each GRU. This means that, given the overall GRU parameters vector $\Theta = \{W_r, U_r, b_r, W_z, U_z, b_z, W_f, U_f, b_f, U_o, b_o\}$, the optimized one is plainly $\hat{\Theta} = \{\hat{U}_o, \hat{b}_o\}$, for each GRU. This trick allows us to gain two benefits. First, the number of optimization variables is dramatically reduced than in the situation where all parameters are optimized. The original MHE algorithm discussed in [12] actually revealed a scalability problem. In fact, if all the network parameters were optimized as in the reference paper, having fifty-four states and six layers, we would probably end up in a very large computational burden or even in an intractability of the problem. However, thanks to the aforementioned simplification, the updated problem resolution typically demands less than two hours of computation. Second, since the parameters describing the interconnections among hidden layers are not optimized but are fixed to their original value ("old" PB-GRU), the catastrophic forgetting issue is effectively averted. Indeed, the information previously acquired through the original training procedure is embedded in the hidden layers parameters, whereas the information related to the current modified plant is stored in the new output parameters vector $\hat{\Theta}^*$.

- The initial state vector is constrained to be equal to the state measured at time $k = 0$ by the open-loop observer made by the original PB-GRU equations (see Section 4.3.7). In this way, the computational effort is reduced and drift issues related to state initialization are bypassed.

- $\hat{x}$ and $\hat{y}$ are randomly initialized. By contrast, $\hat{\Theta}$ is initialized using the values contained in $\bar{\Theta}$ (old model parameters). This warm start trick allows a faster optimization procedure and prevents catastrophic forgetting.

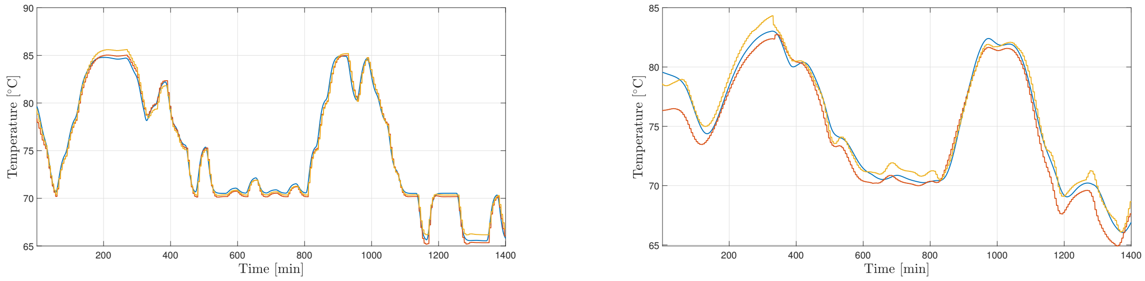
- For computational reasons, all the optimization variables are bounded within certain limits. In particular, $\hat{x}$ and $\hat{y}$, being normalized, are constrained between ±1. By contrast, $\hat{\Theta}$ is forced to stay around a neighbourhood of $\bar{\Theta}$, i.e. it can move away from the original vector at most by $\pm\Delta = \pm 0.5$ (value empirically found). Once again, these constraints ensure a faster optimization procedure and prevent catastrophic forgetting.

5.3.2.2. Results

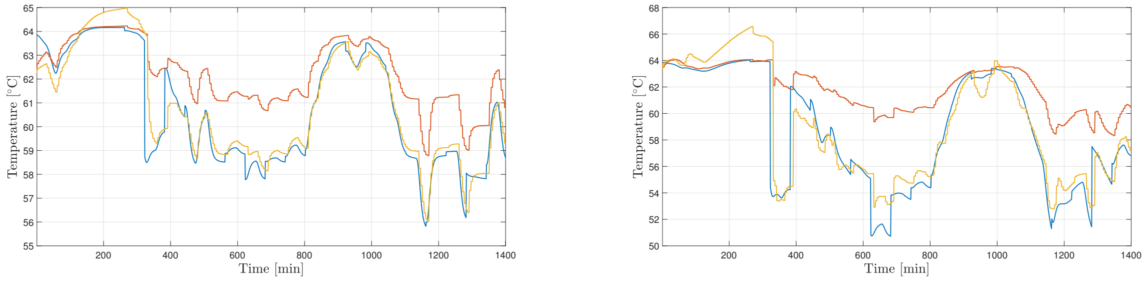
After the description of the Moving Horizon Estimation problem, we can here address a practical example. It is worth reminding that the objective of this section is to find a new set of network parameters to be employed in a novel model capable of describing the dynamics of the plant whose first supply and return pipe has been modified (+20% in the length and -20% in the diameter).

As in Chapter 4, the aforementioned MHE problem is implemented in MATLAB R2022a and solved through CasaADi with the solver Ipopt. In particular, after a closed-loop daily operation of the AROMA network, data are collected. Since several root mean square errors between measured outputs and the ones predicted by the original PB-GRU exceed their thresholds, the statistical distance is computed. $T^2(u)$ does not overcome its threshold and hence we can conclude that the operating conditions are regular whereas there has been a modification in the plant (see Case 2 described in Section 5.3.1). In fact, the inputs used for this simulation are the usual ones: the boiler reference temperature is the one computed by the MPC algorithm and the power consumption is the nominal one reported in Figure 1.10b.
At this stage, the MHE is performed by using the daily collected data ($N = 288$). The optimization procedure took one hour and thirty-two minutes to return the optimized network parameters (weights and biases of the six output layers).
In order to assess the algorithm performance, it is necessary to validate the new model using a different input set with respect to the one employed in the optimization procedure. However, the input values must be included in the training set boundaries so that their statistical distance is within the T^2_α limits.
In Figure 5.4 the behaviours of the old and new model variables compared to the measured ones are shown. In detail, for the sake of conciseness, it is reported just the trend of the first and fifth user variables (respectively, the closest and the furthest with respect to the heating station) and the overall return temperature and mass flow rate. Moreover, in Table 5.3 the performance index evaluation is reported. In particular, since it has not

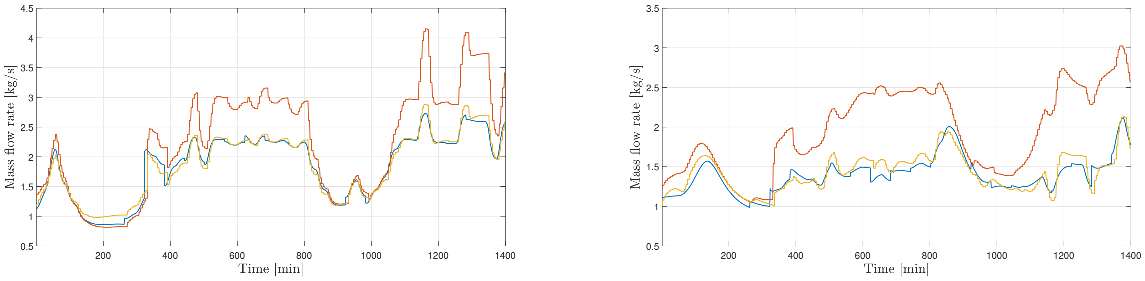
been performed a re-training procedure but an optimization one, we may compare the minimum, maximum and, instead of FIT, average R^2 values (2.2).



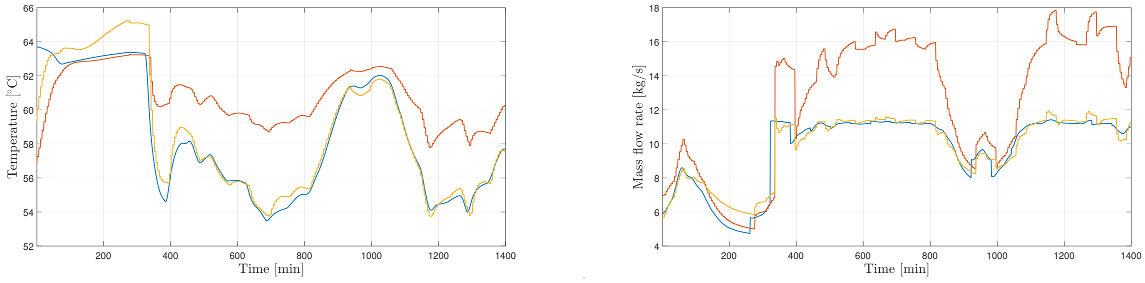
(a) First user supply temperature (b) Fifth user supply temperature



(c) First user return temperature (d) Fifth user return temperature



(e) First user mass flow rate (f) Fifth user mass flow rate



(g) Return temperature (h) Return mass flow rate

Figure 5.4: Comparison among the variables identified by the old PB-GRU (red), by the new PB-GRU (yellow) and the measured ones (blue), in case of a novel plant.

Model	$\boldsymbol{R^2_{min}}$[%]	$\boldsymbol{R^2_{max}}$[%]	$\boldsymbol{R^2_{avg}}$[%]
Original PB-GRU	$-1.4128 \cdot 10^3$	99.3454	-117.4485
New PB-GRU	75.1967	99.4572	91.6002

Table 5.3: Comparison of the R^2 values between the variables identified by the original and by the new model.

As clearly visible from the previous plots and table, the model identified through the MHE algorithm outperforms the old one. Indeed, the two neural networks have the usual physics-based structure, with the same number of neurons and hidden layers. However, because of the structural modification in the first pipe, the old PB-GRU is no longer able to correctly capture the non-linear dynamics of the novel plant (R^2 values small). By contrast, the PB-GRU characterized by the usual interconnections parameters and by new output weights and biases is able to fit well the novel system status (R^2 values high). Actually, by computing again the different $RMSE$ values between measured and new predicted outputs, we find out that all the seventeen errors are below their corresponding triggering thresholds.
In addition, it is convenient to test this new model in closed-loop and to compare it with the case in which the old model is employed. Clearly, the plant that is controlled by the model predictive regulator is the modified one. As a result, we can notice that, even if the two cost functions (represented by the boiler electrical cost) take approximately the same value, the overall simulation time is fairly different. In fact, the MPC algorithm that makes use of the new model (both in the finite horizon control optimization problem and in the observer) requires a total simulation time of one hour and twenty-six minutes. On the other hand, the MPC algorithm that exploits the old model takes two hours and twelve minutes to complete the daily optimization.

To sum up, this lifelong learning solution is definitely suitable in the event of a structural system change: the algorithm is able to improve the identification performance of the old model and, overall, of the model predictive control.

However, as anticipated, the computational effort required by the MHE algorithm is not negligible. Even though the example presented required an optimization time of roughly one hour and half, the latter explodes with the number of optimization variables. For this reason, the computations have been made on a set of data collected at the end of a single day. Larger datasets, instead, would require a much longer resolution time and sometimes they would be even intractable. This computational issue leads us to adopt a naive but rather effective variation of the original MHE algorithm proposed in [12]. In

practice, we remove from the cost function the minimization of the mismatch between the original network parameters and the new optimized ones. In place of this term, we add a constraint on the difference between the output optimization variables and the outputs predicted by the old PB-GRU. This additional constraint aims at avoiding catastrophic forgetting by limiting the mismatch between new and old output variables.
We can synthesize the above-mentioned optimization problem as follows:

$$
\begin{aligned}
\hat{\Theta}^* = \arg\min_{\hat{\Theta},\hat{x}_0} \{J = \sum_{k=0}^{N} ||y_k - \hat{y}_k||^2\} & \\
\text{s.t.} \qquad \hat{x}_0 = \bar{x}_0 & \\
\hat{x}_{k+1} = f(\hat{x}_k, u_k; \hat{\Theta}) & \\
\hat{y}_k = g(\hat{x}_k, u_k; \hat{\Theta}) & \\
||\hat{y}_k - \bar{y}_k||^2 \leq \varepsilon & \\
\forall k \in \{0, .., N\} &
\end{aligned}
\tag{5.4}
$$

In other words, the squared difference between new outputs ($\hat{y}_k$) and old ones ($\bar{y}_k = f(\bar{x}_k, u_k; \bar{\Theta})$) must comply with the maximum difference of ε. This last value has been empirically found with a trial and error approach: the maximum excursion yielding the best prediction result turned out to be 0.8.

By repeating the optimization procedure, new output weights and biases are determined. However, the validation phase suggests that the accuracy capabilities of the model obtained through the "traditional" MHE and of the model obtained through the "updated" MHE are almost overlapping (see Table 5.4). Conversely, the updated algorithm takes an overall optimization time smaller by roughly 25% than the one required by the traditional MHE procedure. In conclusion, the main reason why one may prefer the last proposed algorithm to the standard MHE lies in a faster optimization process.

Model	R^2_{min}[%]	R^2_{max}[%]	R^2_{avg}[%]	**Optimization time**
PB-GRU with traditional MHE	75.1967	99.4572	91.6002	1h32’
PB-GRU with updated MHE	77.0543	99.4177	91.9067	1h9’

Table 5.4: Comparison of the R^2 values and computational times between the traditional MHE problem and the updated one.

5.3.3. Operating Conditions Change Scenario

This section aims at giving an overview on how to proceed in case of operating conditions change. As anticipated, throughout the lifespan of a DHS it is most likely that the working status of the system changes. In detail, for "operating conditions change" we mean significant variations with respect to the usual possible range of power consumption. In Chapter 2, it has been explained how each load profile could typically range from 30kW to 420kW, because of the standard AROMA network functioning. Indeed, this range of powers has also been selected to generate pseudorandom binary signals (PRBS) used as training data so as to identify a system model. Therefore, we may call the power consumptions belonging to that wide interval as "standard" operating conditions. By contrast, even though the input training set includes a large variety of scenarios, it could happen that, because of exogenous and unpredictable factors, the actual load profiles exceed their ordinary range of power. For instance, if a winter were abnormally cold, the power consumptions would be particularly high and out of the original working conditions area.

Clearly, an alteration in the input values does not require an overall re-identification procedure, since the original model is still able to predict the system dynamics in the vast majority of the cases. Therefore, it is simply necessary to enlarge the initial neural network model so that it is able to identify the plant behaviour even when the operating conditions are unusual. Actually, the underlying issue consists in the lack of abnormal power consumptions in the original training set. As a consequence, the "old" physics-based gated recurrent unit network is unable to correctly simulate the system dynamics only when anomalous disturbances occur.

Before discussing the resolution procedure of the just mentioned problem, it is useful to introduce an example of unusual inputs for the AROMA network. A reasonable and effective change could be to triplicate or even quadruplicate the overall power consumption, as shown in Figure 5.5. It goes without saying that, being the load profiles out of the original training set, the old PB-GRU described in Chapter 3 fails to properly predict the system dynamics, in particular in correspondence of power peaks (see Figure 5.8).

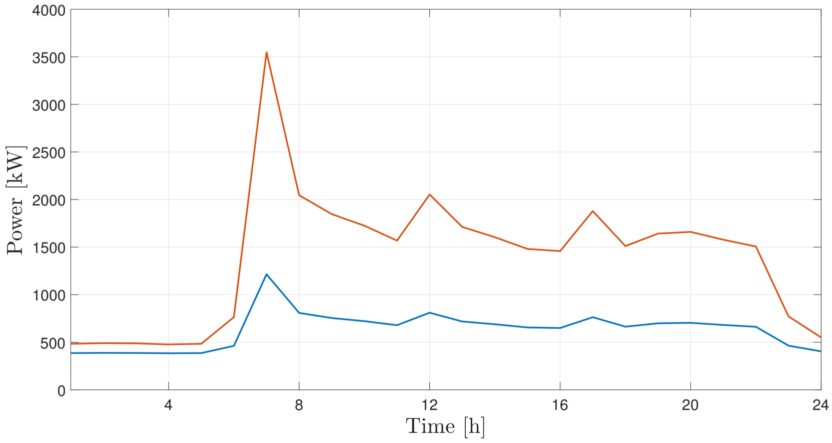


Figure 5.5: Comparison between a daily overall standard power consumption (blue) and an abnormal one (red).

5.3.3.1. Model Uncertainty Identification

In order to tackle the problem of new operating conditions, we exploit a model uncertainty identification strategy. An interesting insight on learning-based model predictive control using different types of uncertainty is provided in [32].

The first thing to notice is that, in case of working status change, there is no need to re-identify from scratch the system model. Indeed, we would like to preserve the existing model since it is able to capture most working conditions. In addition, by maintaining the original neural network equations it is possible to prevent catastrophic forgetting. However, when the plant is fed with very unusual inputs, the old PB-GRU alone is not enough. For this reason, we propose a simple yet efficient strategy to enlarge the original system model. In particular, the goal is to estimate the model uncertainty caused by new and unpredictable disturbances. To this end, after data collection, it is necessary to identify offline a further network able to capture the model error dynamics. This novel neural network is referred to as "additive", "incremental" or "ΔRNN" (see Figure 5.6).

At this stage, a new training procedure can be run exploiting Python and the usual library implemented in [8]. However, three main differences with respect to the procedure analysed in the previous chapters can be pointed out.

First, in the model identification procedure carried out in Chapter 2 and 3, input data were highly informative being made up of a large amount of pseudorandom binary sequences. In this chapter however, since the exciting signals information is already embedded in the original PB-GRU plus it is simply necessary to find out the error dynamics caused by abnormal power consumptions, we may exploit realistic data gathered during closed-loop operations. In this way, the additive neural network would be fed with further information, that is, actual measurements from the controlled plant, and it should be able to refine the predictive capability of the existing PB-GRU. Furthermore, these data must be collected after a certain period of time. Differently from the structural change case (Section 5.3.2) where the plant could be modified overnight, an operating conditions modification typically depends on the season of the year. For this reason, it makes little sense to collect only a single-day data. Instead, it is far more reasonable to gather at least a few days of plant operation measurements. This choice is likewise motivated by the fact that the additive neural network must be trained from scratch and hence an adequate amount of data is required. In addition, besides data collected during abnormal operating conditions, it is intuitive to think that the additive neural network needs some data gathered during ordinary operating conditions too, so that the new total PB-GRU is able to predict both plant situations. In fact, if ΔPB-GRU is trained with working conditions out of the standard power range only, when normal load profiles occur the additive network negatively affects the prediction of the original PB-GRU, resulting in an overall deteriorated performance. To sum up, it is essential to feed the additive neural network with a decent amount of data, gathered both in normal and in abnormal operating conditions.

Second, as far as the output data are concerned, it is necessary to collect at the end of a certain period of time the difference between actual measured outputs y and the corresponding values $\hat{y}$ predicted by the original PB-GRU, i.e. $\varepsilon = \Delta\hat{y} = y - \hat{y}$. In this way, the additive neural network identifies the model uncertainty dynamics, namely the prediction error made by the primary network.

Third, even though the physics-based structure of the incremental neural network is not altered, a modification in the number of total states, and hence neurons, is required. Indeed, adding a further neural network results in additional optimization variables (as many as the states and outputs of the ΔRNN) in an MPC problem. For this reason, we seek to keep the amount of additive states as low as possible. In addition, the error dynamics that must be identified by the incremental PB-GRU is pretty limited and thus a small number of neurons is entirely legitimate.

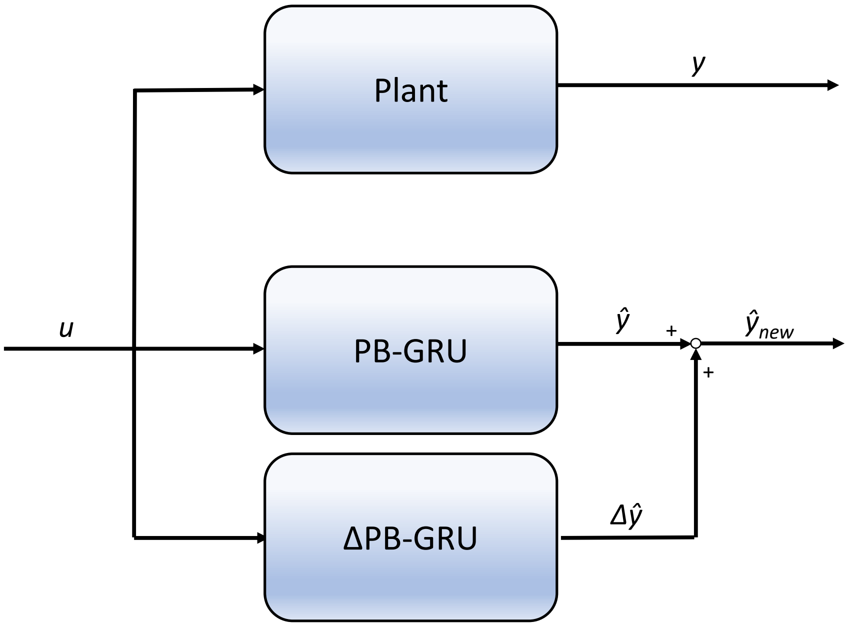


Figure 5.6: Schematic representation of the additive network structure. The incremental part is denoted by ΔPB-GRU and its objective is to identify the model-plant mismatch, i.e. $\Delta\hat{y}$. Altogether, this term is added to the original output estimation $\hat{y}$ so as to find the actual predicted output $\hat{y}_{new}$.

5.3.3.2. Results

In this section some numerical results regarding the operating conditions change problem are reported.

First, let us assume that after a broad period of the AROMA network functioning, we collect data from the last week, where anomalous power consumptions have been fed to the system. As usual, the first thing to do is to monitor the system status by computing the $RMSE$. We find out that various errors overcome their corresponding thresholds, hence the calculation of statistical distances is required. Since the Mahalanobis distances related to input and output overcome their thresholds with, respectively, the values of 14.6838 and 62.3070, we may conclude that a change in the operating conditions occurred. At this stage, a training procedure used to identify the model error dynamics is carried out. In addition to the seven days where unusual working conditions have been detected, we insert in the training set an equal number of realistic data collected during standard operating conditions, so as to have a balance between them. After all, we can assume that, since we spotted some data in anomalous conditions, the AROMA network must have been operated in a much larger number of normal conditions, whose data are therefore at our disposal.

Second, having the plant-model mismatch a pretty limited dynamics, we can lower the number of neurons used in the additive network with respect to the one employed for the original PB-GRU. For instance, we could train an incremental neural network characterized by one third (18) of the original states (54). This modification is actually undertaken for control purposes too. Indeed, the lower is the number of network neurons, the lighter is the MPC computational burden. However, for the sake of completeness, we tested both a ΔPB-GRU having, altogether, 54 states and one having 18 states ([2,2,2,3,4,5] neurons). Clearly, the former showed a slightly higher accuracy than the latter, but the MPC algorithm exploiting the 54-state incremental network was basically unmanageable, contrarily to the 18-state one. In conclusion, an effective trade-off between model accuracy and computational effort could be to use an incremental neural network with one third of the original neurons.

Finally, it is convenient to visualize some identification results. Once the additive RNN is trained, it is able to predict the error dynamics in many cases (see Figure 5.7).

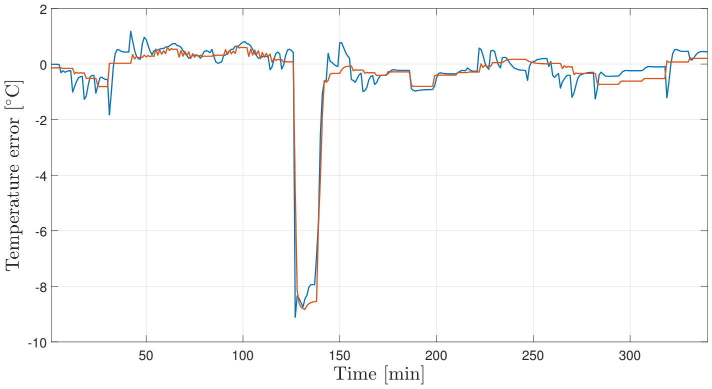
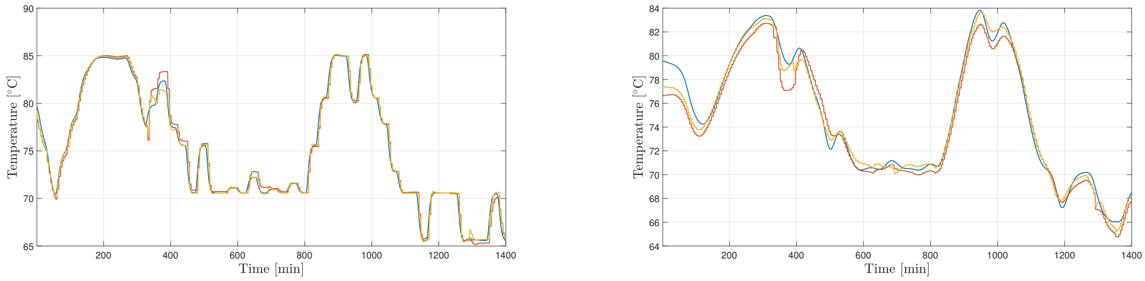


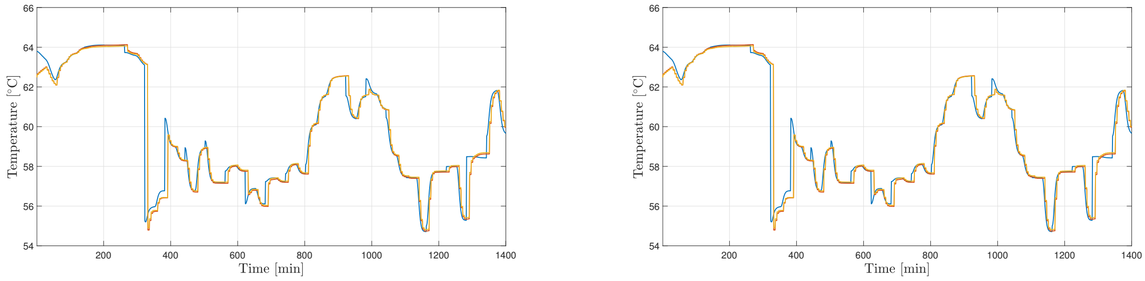
Figure 5.7: Additive network identification results. The error between the measured supply temperature of the second user and the one predicted by the old PB-GRU (blue) is compared to the error identified by the additive PB-GRU (red).

Even though the accuracy is not extremely high due to the presence of small oscillations around 0°C (FIT = 20.3%), the predictive capability of the overall new PB-GRU, i.e. the summation of the old PB-GRU and the additive one, is impressive.

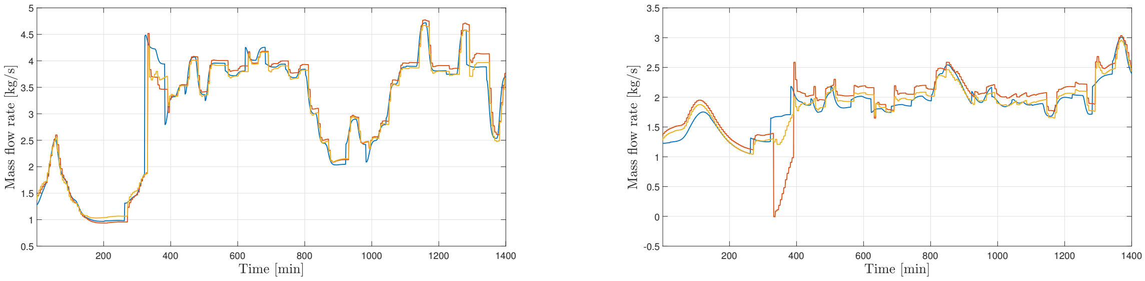
In fact, when the validation procedure is performed using novel inputs out of the standard training range (the nominal overall power consumption is multiplied by a 3.7 coefficient), the results are very satisfactory. As visible in Figure 5.8 and in Table 5.5, the total neural network (original PB-GRU+ΔPB-GRU) is perfectly able to capture the system dynamics even in presence of abnormal power profiles.



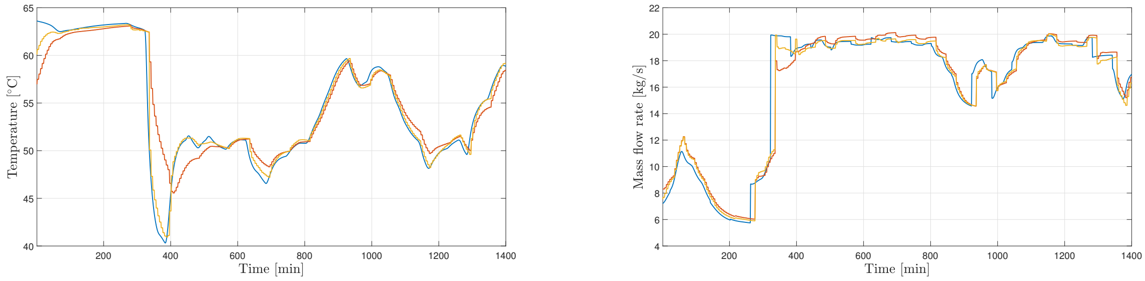
(a) First user supply temperature (b) Fifth user supply temperature



(c) First user return temperature (d) Fifth user return temperature



(e) First user mass flow rate (f) Fifth user mass flow rate



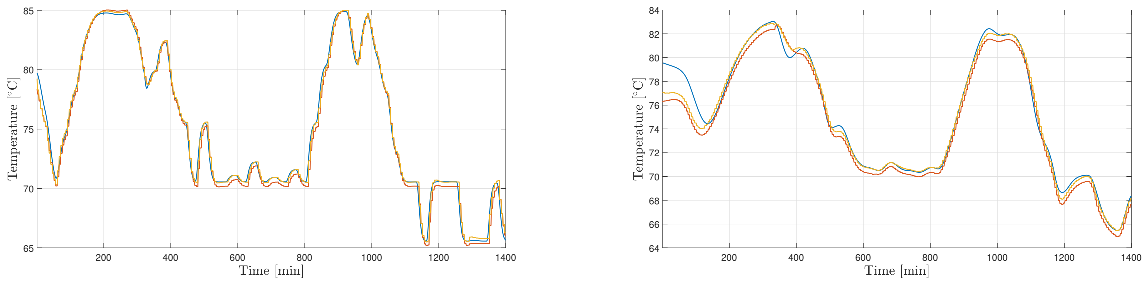
(g) Return temperature (h) Return mass flow rate

Figure 5.8: Comparison among the variables identified by the old PB-GRU (red), by the new overall PB-GRU (yellow) and the measured ones (blue), in case of abnormal operating conditions.

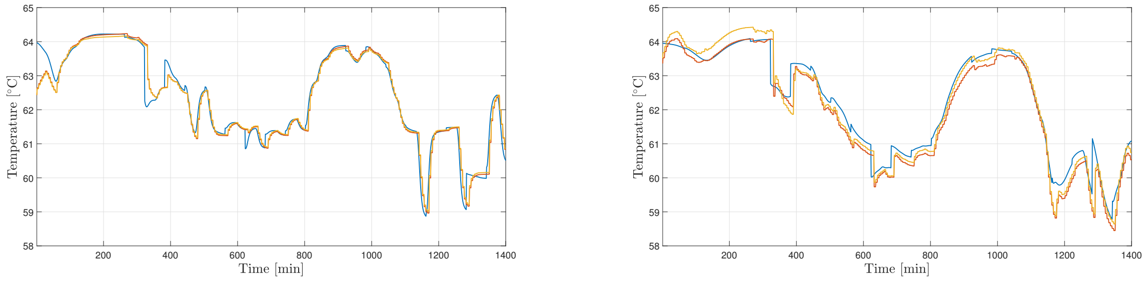
Model	$\boldsymbol{R^2_{min}}$[%]	$\boldsymbol{R^2_{max}}$[%]	$\boldsymbol{R^2_{avg}}$[%]
Original PB-GRU	20.8904	98.5662	78.2661
PB-GRU+ΔPB-GRU	78.9819	99.0748	91.9028

Table 5.5: Comparison of the R^2 values between the variables identified by the original and by the new overall network, in case of abnormal operating conditions.

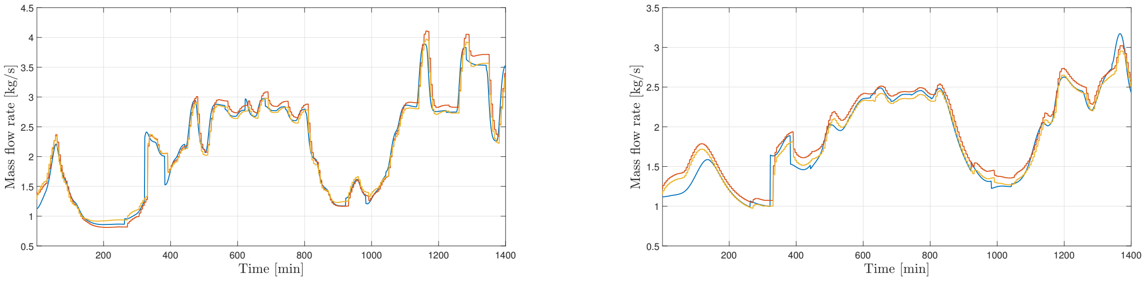
Moreover, the new total PB-GRU is also able to correctly capture the plant dynamics when standard operating conditions are restored. This means that the additive neural network does not deteriorate the original PB-GRU performance, but rather the latter is slightly improved thanks to the incremental training data collected during closed-loop operations (see Figure 5.9 and Table 5.6).



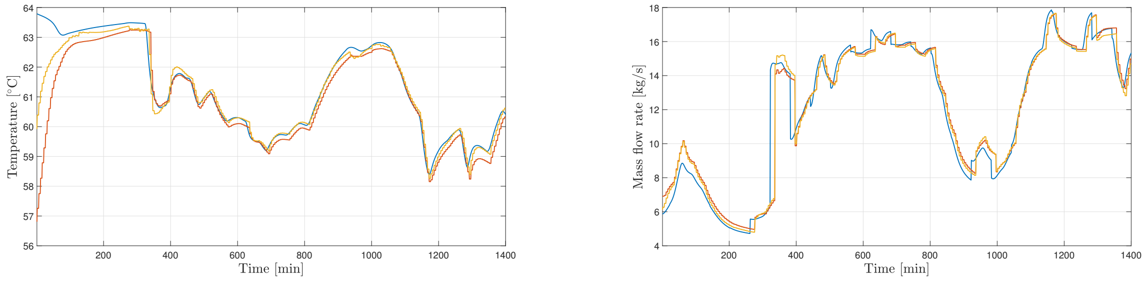
(a) First user supply temperature (b) Fifth user supply temperature



(c) First user return temperature (d) Fifth user return temperature



(e) First user mass flow rate (f) Fifth user mass flow rate



(g) Return temperature (h) Return mass flow rate

Figure 5.9: Comparison among the variables identified by the old PB-GRU (red), by the new overall PB-GRU (yellow) and the measured ones (blue), in case of normal operating conditions.

Model	$\boldsymbol{R^2_{min}}[\%]$	$\boldsymbol{R^2_{max}}[\%]$	$\boldsymbol{R^2_{avg}}[\%]$
Original PB-GRU	64.2446	99.0405	92.8900
PB-GRU+ΔPB-GRU	87.9900	99.4243	95.1885

Table 5.6: Comparison of the R^2 values between the variables identified by the original and by the new overall network, in case of normal operating conditions.

Finally, it is convenient to test this new total model in closed-loop. The results are indeed pretty regular, i.e. no constraints violation occurs and the control variable varies according to the principles explained in Chapter 4. However, the overall simulation time is definitely greater than the usual one, but this does not surprise. As anticipated, adding a further neural network to the MPC scheme has the effect of increasing the number of optimization variables. Thereby, the so-updated MPC regulator requires almost three hours to complete the optimization procedure over a daily operating simulation.

To sum up, the lifelong learning solution proposed in this section is perfectly suitable in the case of a working conditions change: the algorithm is actually able to improve the identification performance of the original model alone.

However, one might wonder why, since a further training procedure is performed, it is not carried out a re-training of the original neural network with additive input data. For sure, this is a plausible alternative. In fact, the original PB-GRU could be trained from scratch appending to the initial PRBS input signals some data collected during anomalous operating conditions. If we compare the validation results of the overall PB-GRU (obtained by summing up the standard GRU and the additive one) and of the PB-GRU which has been re-trained with further information, we can notice that the performances are pretty similar. In fact, they have an average R^2 value of 91.9028 and 91.9536, respectively. This should come at no surprise since the additional data appended to the original ones to re-train the primary PB-GRU are exactly the same data through which the incremental network has been trained.
However, the re-training procedure of the original PB-GRU with further data took almost five hours and half (1500 epochs). By contrast, the additive network training lasted around three hours, for the same amount of epochs. In conclusion, we may say that re-training the original PB-GRU is not so efficient in terms of computational time, also because several hours were originally spent to perform the initial training where most of the new dataset was already included. In fact, the amount of data collected in anomalous operating conditions is undoubtedly lower than the original PRBS length. For this reason, we may conclude that it is more efficient to train a smaller additive PB-GRU with a restricted dataset made of only normal or abnormal realistic operating conditions that were not included in the original training set.

5.4. Long-term Monitoring

In this section some final remarks are highlighted. Indeed, lifelong learning is still an open issue under several aspects and, in particular, this chapter treatment leaves some questions unanswered.

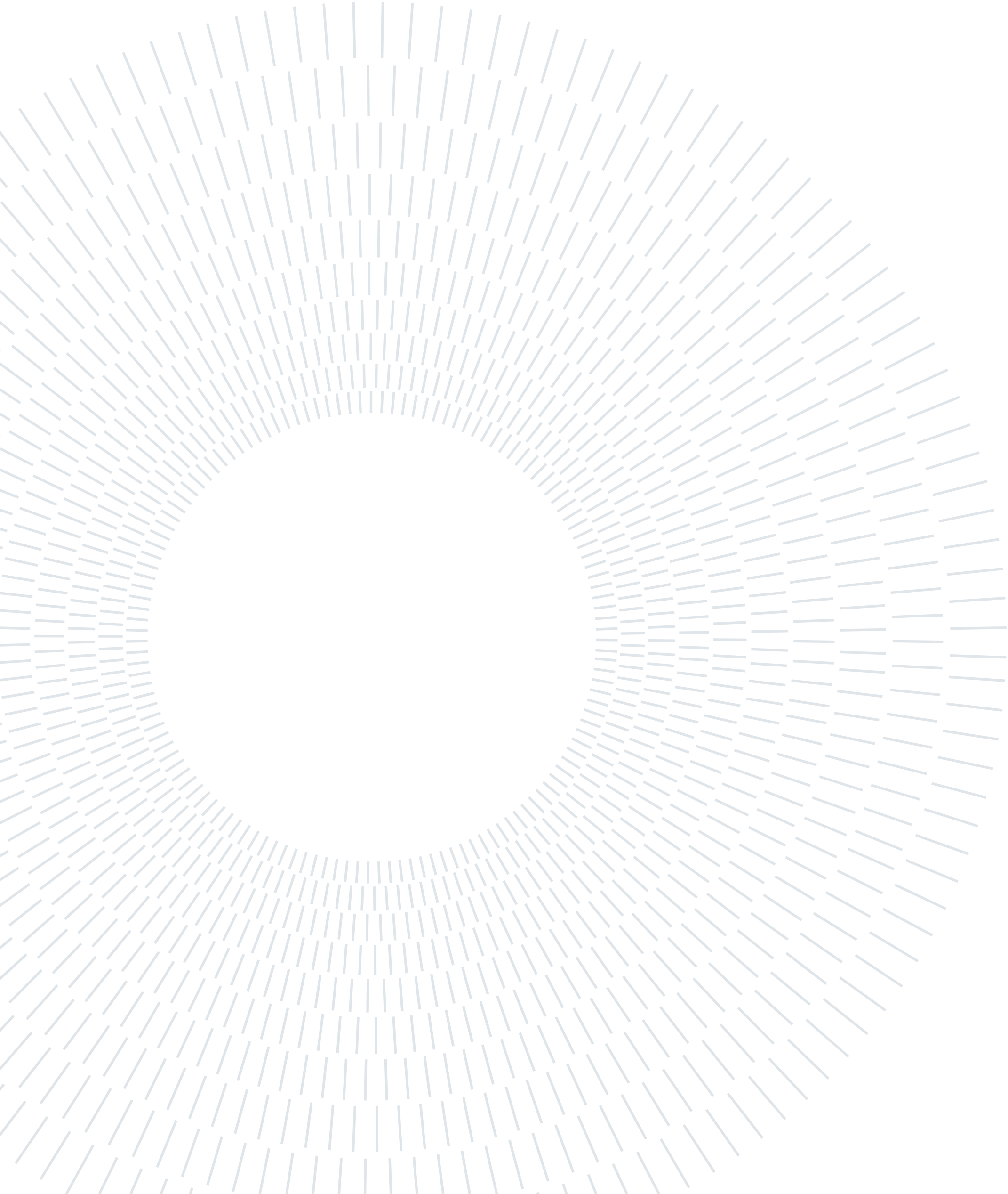
First, one might wonder what happens when multiple changes of plant occur over the years. The method proposed in this thesis, namely the MHE technique, requires a new optimization procedure each time a structural change is detected (through $RMSE$ and T^2). Hence, we could conclude that, since a structural modification in the plant is independent from others, one has to re-optimize the network output weights and biases every time new data are available.

Second, a similar rationale can be applied to the operating conditions change scenario. In fact, when a new working status is detected, the additive RNN must be re-trained by appending to existing unusual data the new ones. By doing so, we end up having a single incremental RNN which is re-trained with an updated training set containing the standard working status, "old" abnormal conditions and "new" ones. However, this circumstance would not happen really often, being the original training set very wide and the anomalous cases not so frequent, by definition. An alternative could be to train a novel RNN each time new conditions are sensed. In this way, in the global MPC scheme we would have the summation of a certain number of additive networks. Unfortunately, this second option could bring to an intractability of the control problem. Actually, if several incremental networks were added, the finite horizon control optimization problem would have a very large number of optimization variables and, eventually, it would become unmanageable. In conclusion, when anomalous conditions are detected, we propose to simply re-train the already existing additive RNN with larger and larger datasets. Clearly, once the incremental network is trained and tested, the benchmark training set with respect to which the Mahalanobis distance is computed must be enlarged including the new anomalous operating conditions that, by this time, are no longer anomalous.

Finally, the two proposed algorithms for the lifelong learning and monitoring of a district heating system can be applied one right after the other, offline, when simultaneous modifications occur in the plant.

5.5. Conclusions

In this chapter the challenge of long-term monitoring of the AROMA network has been tackled. In detail, a methodological algorithm to manage different kinds of scenario has been proposed. Our lifelong learning strategy mainly focused on anomaly detection through the simple but rather effective computation of the statistical distance. Moreover, for each detected scenario, a different resolution algorithm has been provided. In particular, in case of plant change, a simplification of the standard moving horizon estimation problem has been adopted so as to significantly lower the computational time. Furthermore, in case of operating conditions change, a model uncertainty identification, once again performed through an incremental physics-based neural network, has been presented. In conclusion, we managed to cope with the lifelong learning and control problem via multiple techniques fine-tuned according to the type of change detected.



Conclusions and Future Developments

In the Introduction, a general overview on the thesis challenges and objectives has been presented. In this final chapter, it is therefore useful to summarize the main achievements and possible developments of the work.

The first outcome consists in the modelling of a district heating network through data-driven methods, and in particular via recurrent neural networks. Even though this technique turns out to have greater identification performance than classical models such as state-space and polynomials, a further improvement can be made. In fact, the crucial contribution of the thesis consists in the development of a physics-based machine learning method. Thanks to a structure of the neural network that resembles the physical system topology and interactions, many enhancements with respect to standard RNNs are obtained: higher predictive accuracy, faster training procedure, greater interpretability and easier problem detection.

A further achievement of the work is related to the non-linear model predictive control strategy applied to district heating systems. In particular, an MPC scheme has been implemented by making use of the physics-based gated recurrent unit equations both in the finite horizon control optimization problem formulation and in the state observer design.

The last contribution regards the lifelong learning issue. Since, in the long run, the district heating system under analysis must be continually monitored and supervised, we proposed a novel algorithm to tackle the problem of model changes over time. In particular, an effective solution depending on the category of scenario detected was presented.

Finally, although the results achieved in thesis are rewarding, the work is nonetheless a starting point for many possible reflections and developments.

Above all, given that the physics-based neural network developed in Chapter 3 has been only tested on the AROMA network case study, it would be desirable to try out the

aforementioned machine learning method on other and different types of complex systems, such as industrial and chemical plants, but also energy and biological systems. In fact, since such systems are often interconnected through physical networks characterized by a well-defined modular and sequential topology, it would reasonable to identify their model through the physics-informed machine learning approach. In the end, a formal and methodological generalization of the just mentioned technique could be eventually synthesized so that various applications could benefit from it.
In addition, as anticipated in Chapter 3, for plants that present unclear interactions among their nodes (*meshed* networks), the PB-RNN implementation is not trivial and requires further in-depth studies.

Another challenge left open in Chapter 4 regards the implementation of a closed-loop state observer by means of physics-based neural network equations. Actually, for the sake of simplicity, we implemented a plain open-loop observer. Indeed, the latter makes use of the PB-GRU model, but, from the control point of view, it would be more effective implementing an observer which exploits a feedback innovation term too. Additionally, for an even more efficient predictive control strategy, it would be necessary to develop a powerful disturbance forecasting algorithm, which was not provided in this thesis because out of its scope.

Lastly, the lifelong learning topic, being an outstanding issue by itself, left some questions unanswered. Indeed, the challenge with task-incremental learning is mainly to optimize the trade-off between performance and computational complexity and to use information learned in one task to improve performance on other tasks [85].
In this thesis case, for instance, the adopted moving horizon estimation algorithm still lacks of scalability [12]. In other words, when a huge number of optimization variables is inserted in the problem, the computational time grows exponentially. It would be therefore advisable to modify the MHE algorithm in order to get rid of the intractability issue caused by the high dimensionality of the optimization problem.
Finally, the identification procedure of the incremental neural network proposed in Chapter 5 could be definitely deepened and enhanced. In fact, the method is run offline and the additive PB-RNN results in a more complex control problem. However, the various approaches suggesting to estimate model uncertainty online [32, 90, 94] yield poor predictive accuracy because of the data scarcity. In the end, further investigation regarding the trade-off between online and offline plant-model mismatch estimation is surely advisable.

Bibliography

[1] *MATLAB, "System Identification Toolbox", Mathworks. [ONLINE].*

[2] A. Alanqar, H. Durand, and P. D. Christofides. Error-triggered on-line model identification for model-based feedback control. *AIChE Journal*, 63(3):949–966, 2017.

[3] B. Anderson, T. S. Hy, and R. Kondor. Cormorant: Covariant molecular neural networks. *Advances in neural information processing systems*, 32, 2019.

[4] E. M. Azoff. *Neural network time series forecasting of financial markets*. John Wiley & Sons, Inc., 1994.

[5] A. Bemporad. Recurrent neural network training with convex loss and regularization functions by extended kalman filtering. *IEEE Transactions on Automatic Control*, 2022.

[6] F. Bianchi, A. Castellini, P. Tarocco, and A. Farinelli. Load forecasting in district heating networks: Model comparison on a real-world case study. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 553–565. Springer, 2019.

[7] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen. Recurrent neural networks for short-term load forecasting: an overview and comparative analysis. Springer. 2017.

[8] F. Bonassi. *ssnet: a Python module for training State Space neural NETworks, Available:* `https://github.com/bonassifabio/ssnet`.

[9] F. Bonassi, E. Terzi, M. Farina, and R. Scattolini. Lstm neural networks: Input to state stability and probabilistic safety verification. In *Learning for Dynamics and Control*, pages 85–94. PMLR, 2020.

[10] F. Bonassi, O. da Silva, and R. Scattolini. Nonlinear mpc for offset-free tracking of systems learned by gru neural networks. *IFAC-PapersOnLine*, 54(14):54–59, 2021.

[11] F. Bonassi, M. Farina, J. Xie, and R. Scattolini. On recurrent neural networks for learning-based control: recent results and ideas for future developments. *Journal of Process Control*, 114:92–104, 2022.

[12] F. Bonassi, J. Xie, M. Farina, and R. Scattolini. Towards lifelong learning of recurrent

neural networks for control design. In *2022 European Control Conference (ECC)*, pages 2018–2023. IEEE, 2022.

[13] J. A. Bullinaria. Recurrent neural networks. *Neural Computation: Lecture*, 12, 2013.

[14] Z. Chang, Y. Zhang, and W. Chen. Electricity price prediction based on hybrid model of adam optimized lstm neural network and wavelet transform. *Energy*, 187: 115804, 2019.

[15] B. Chen, C. Shen, D. Wang, L. Kong, L. Chen, and Z. Zhu. A lifelong learning method for gearbox diagnosis with incremental fault types. *IEEE Transactions on Instrumentation and Measurement*, 71:1–10, 2022.

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[17] A. Daw, A. Karpatne, W. Watkins, J. Read, and V. Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[18] A. Daw, R. Q. Thomas, C. C. Carey, J. S. Read, A. P. Appling, and A. Karpatne. Physics-guided architecture (pga) of neural networks for quantifying uncertainty in lake temperature modeling. In *Proceedings of the 2020 siam international conference on data mining*, pages 532–540. SIAM, 2020.

[19] G. De Nicolao and R. Scattolini. *Identificazione parametrica.* CUSL ed, 1997.

[20] F. Dörfler, J. W. Simpson-Porco, and F. Bullo. Electrical networks and algebraic graph theory: Models, properties, and applications. *Proceedings of the IEEE*, 106 (5):977–1005, 2018.

[21] J. Drgoňa, A. R. Tuor, V. Chandan, and D. L. Vrabie. Physics-constrained deep learning of multi-zone building thermal dynamics. *Energy and Buildings*, 243:110992, 2021.

[22] S. G. Dukelow. The control of boilers. Instrument Society of America, Research Triangle Park, NC. 1986.

[23] H. Fang, J. Xia, K. Zhu, Y. Su, and Y. Jiang. Industrial waste heat utilization for low temperature district heating. *Energy policy*, 62:236–246, 2013.

[24] S. S. Farahani, Z. Lukszo, T. Keviczky, B. De Schutter, and R. M. Murray. Robust model predictive control for an uncertain smart thermal grid. In *2016 European Control Conference (ECC)*, pages 1195–1200. IEEE, 2016.

[25] F. Filippetti, G. Franceschini, and C. Tassoni. Neural networks aided on-line diagnostics of induction motor rotor faults. In *Conference Record of the 1993 IEEE Industry Applications Conference Twenty-Eighth IAS Annual Meeting*, pages 316–323. IEEE, 1993.

[26] B. Foss and T. A. N. Heirung. Merging optimization and control. *Lecture Notes*, 2013.

[27] N. Fumo. A review on the basics of building energy estimation. *Renewable and sustainable energy reviews*, 31:53–60, 2014.

[28] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[29] E. Guelpa, G. Mutani, V. Todeschi, and V. Verda. Reduction of co2 emissions in urban areas through optimal expansion of existing district heating networks. *Journal of Cleaner Production*, 204:117–129, 2018.

[30] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[31] R. Hedjar. Adaptive neural network model predictive control. *International Journal of Innovative Computing, Information and Control*, 9(3):1245–1257, 2013.

[32] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[33] J. Hinrichs, D. Felsmann, S. Schweitzer-De Bortoli, H.-J. Tomczak, and H. Pitsch. Numerical and experimental investigation of pollutant formation and emissions in a full-scale cylindrical heating unit of a condensing gas boiler. *Applied Energy*, 229: 977–989, 2018.

[34] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

[35] S. Idowu, S. Saguna, C. Åhlund, and O. Schelén. Applied machine learning: Forecasting heat load in district heating system. *Energy and Buildings*, 133:478–488, 2016.

[36] H. Jabbar and R. Z. Khan. Methods to avoid over-fitting and under-fitting in su-

pervised machine learning (comparative study). *Computer Science, Communication and Instrumentation Devices*, 70, 2015.

[37] X. Jia, A. Karpatne, J. Willard, M. Steinbach, J. Read, P. C. Hanson, H. A. Dugan, and V. Kumar. Physics guided recurrent neural networks for modeling dynamical systems: Application to monitoring water temperature and quality in lakes. *arXiv preprint arXiv:1810.02880*, 2018.

[38] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 558–566. SIAM, 2019.

[39] A. Karpatne, R. Kannan, and V. Kumar. *Knowledge Guided Machine Learning: Accelerating Discovery using Scientific Knowledge and Data*. CRC Press, 2022.

[40] S. Karsoliya. Approximating number of hidden layer neurons in multiple hidden layer bpnn architecture. *International Journal of Engineering Trends and Technology*, 3 (6):714–717, 2012.

[41] A. Khandelwal, S. Xu, X. Li, X. Jia, M. Stienbach, C. Duffy, J. Nieber, and V. Kumar. Physics guided machine learning methods for hydrology. *arXiv preprint arXiv:2012.02854*, 2020.

[42] K. Kontu, S. Rinne, and S. Junnila. Introducing modern heat pumps to existing district heating systems–global lessons from viable decarbonizing of district heating in finland. *Energy*, 166:862–870, 2019.

[43] R. Krug, V. Mehrmann, and M. Schmidt. Nonlinear optimization of district heating networks. *Optimization and Engineering*, 22(2):783–819, 2021.

[44] A. La Bella and A. Del Corno. Optimal management and data-based predictive control of district heating systems: The Novate Milanese experimental case-study. *Control Engineering Practice*, 132:105429, 2023.

[45] A. La Bella, A. Del Corno, and A. Scaburri. Data-driven modelling and optimal management of district heating networks. In *2021 AEIT International Annual Conference (AEIT)*, pages 1–6. IEEE, 2021.

[46] M. Lalo and R. Scattolini. *Advanced and multivariable control.* Pitagora editrice Bologna, 2014.

[47] R. Lippmann. Book review:" neural networks, a comprehensive foundation", by simon haykin. *International Journal of Neural Systems*, 5(04):363–364, 1994.

[48] F. Liu, L. Zheng, and R. Zhang. Emissions and thermal efficiency for premixed burners in a condensing gas boiler. *Energy*, 202:117449, 2020.

[49] L. Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

[50] L. Ljung. Black-box models from input-output measurements. In *IMTC 2001. Proceedings of the 18th IEEE instrumentation and measurement technology conference. Rediscovering measurement in the age of informatics (Cat. No. 01CH 37188)*, volume 1, pages 138–146. IEEE, 2001.

[51] P. D. Lund, J. Mikkola, and J. Ypyä. Smart energy system design for large clean power schemes in urban areas. *Journal of Cleaner Production*, 103:437–445, 2015.

[52] L. Lyu, Z. Chen, and B. Yao. Development of pump and valves combined hydraulic system for both high tracking precision and high energy efficiency. *IEEE Transactions on Industrial Electronics*, 66(9):7189–7198, 2018.

[53] D. Machalek, J. Tuttle, K. Andersson, and K. M. Powell. Dynamic energy system modeling using hybrid physics-based and machine learning encoder-decoder models. *Energy and AI*, page 100172, 2022.

[54] E. Mäki, L. Kannari, I. Hannula, and J. Shemeikka. Decarbonization of a district heating system with a combination of solar heat and bioenergy: A techno-economic case study in the northern european context. *Renewable Energy*, 175:1174–1199, 2021.

[55] D. Mandic and J. Chambers. *Recurrent neural networks for prediction: learning algorithms, architectures and stability.* Wiley, 2001.

[56] B. C. Mateus, M. Mendes, J. T. Farinha, R. Assis, and A. M. Cardoso. Comparing lstm and gru models to predict the condition of a pulp paper press. *Energies*, 14(21): 6958, 2021.

[57] T. Miconi, K. Stanley, and J. Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International Conference on Machine Learning*, pages 3559–3568. PMLR, 2018.

[58] D. C. Montgomery. *Introduction to statistical quality control.* John Wiley & Sons, 2020.

[59] K. R. Muske and T. A. Badgwell. Disturbance modeling for offset-free linear model predictive control. *Journal of Process Control*, 12(5):617–632, 2002.

[60] S. A. Niaki, E. Haghighat, T. Campbell, A. Poursartip, and R. Vaziri. Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture. *Computer Methods in Applied Mechanics and Engineering*, 384:113959, 2021.

[61] L. Nigro. Hierarchical predictive control of networked multi-energy systems. M.sc. thesis. 2022.

[62] E. O'Riordan. *Distance measures and whitening procedures for high dimensional data*. PhD thesis, Cardiff University, 2023.

[63] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

[64] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[65] Y. Putter. Distributed control design of district heating networks. M.sc. thesis. 2018.

[66] M. Qu, O. Abdelaziz, and H. Yin. New configurations of a heat recovery absorption heat pump integrated with a natural gas boiler for boiler efficiency improvement. *Energy Conversion and Management*, 87:175–184, 2014.

[67] D. Quaggiotto, J. Vivian, and A. Zarrella. Management of a district heating network using model predictive control with and without thermal storage. *Optimization and Engineering*, 22(3):1897–1919, 2021.

[68] S. Rasp, M. S. Pritchard, and P. Gentine. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39): 9684–9689, 2018.

[69] B. Rezaie and M. A. Rosen. District heating and cooling: Review of technology and potential enhancements. *Applied energy*, 93:2–10, 2012.

[70] G. Sandou, S. Font, S. Tebbani, A. Hiret, C. Mondon, S. Tebbani, A. Hiret, and C. Mondon. Predictive control of a complex district heating network. In *IEEE conference on decision and control*, volume 44, page 7372. Citeseer, 2005.

[71] A. S. Santra and J.-L. Lin. Integrating long short-term memory and genetic algorithm for short-term load forecasting. *Energies*, 12(11):2040, 2019.

[72] S. Schmidgall and J. Hays. Stable lifelong learning: Spiking neurons as a solution to instability in plastic neural networks. In *Neuro-Inspired Computational Elements Conference*, pages 1–7, 2022.

[73] O. Schön, R.-S. Götte, and J. Timmermann. Multi-objective physics-guided recurrent neural networks for identifying non-autonomous dynamical systems. *arXiv preprint arXiv:2204.12972*, 2022.

[74] J. Schoukens and L. Ljung. Nonlinear system identification: A user-oriented road map. *IEEE Control Systems Magazine*, 39(6):28–99, 2019.

[75] N. Shi and D. Li. Rmsprop converges with proper hyperparameter. In *international conference on learning representation*, 2021.

[76] A. Soltoggio, K. O. Stanley, and S. Risi. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*, 108:48–67, 2018.

[77] T. Sommer, S. Mennel, and M. Sulzer. Lowering the pressure in district heating and cooling networks by alternating the connection of the expansion vessel. *Energy*, 172: 991–996, 2019.

[78] S. Spinelli. Optimization and control of smart thermal-energy grids. PhD thesis. 2021.

[79] D. Srivastava, Y. Singh, and A. Sahoo. Auto tuning of rnn hyper-parameters using cuckoo search algorithm. In *2019 Twelfth International Conference on Contemporary Computing (IC3)*, pages 1–5. IEEE, 2019.

[80] F. Sun, L. Fu, S. Zhang, and J. Sun. New waste heat district heating system with combined heat and power based on absorption heat exchange cycle in china. *Applied Thermal Engineering*, 37:136–144, 2012.

[81] B. Talebi, P. A. Mirzaei, A. Bastani, and F. Haghighat. A review of district heating systems: modeling and optimization. *Frontiers in Built Environment*, 2:22, 2016.

[82] E. Terzi, F. Bonassi, M. Farina, and R. Scattolini. Learning model predictive control with long short-term memory networks. *International Journal of Robust and Nonlinear Control*, 31(18):8877–8896, 2021.

[83] V. Thangarasa, T. Miconi, and G. W. Taylor. Differentiable hebbian plasticity for continual learning. In *International conference on machine learning (ICML) adaptive and multitask learning: Algorithms & Systems (AMTL) workshop*, page 2019, 2019.

[84] M. Uzair and N. Jamil. Effects of hidden layers on the efficiency of neural networks. In *2020 IEEE 23rd international multitopic conference (INMIC)*, pages 1–6. IEEE, 2020.

[85] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, pages 1–13, 2022.

[86] F. Verrilli, S. Srinivasan, G. Gambino, M. Canelli, M. Himanka, C. Del Vecchio, M. Sasso, and L. Glielmo. Model predictive control-based optimal operations of district heating system with thermal energy storage and flexible loads. *IEEE Transactions on Automation Science and Engineering*, 14(2):547–557, 2016.

[87] A. Wachter. *An interior point algorithm for large-scale nonlinear optimization with applications in process engineering.* PhD thesis, Carnegie Mellon University, 2002.

[88] M. Wirtz, L. Neumaier, P. Remmen, and D. Müller. Temperature control in 5th generation district heating and cooling networks: An milp-based operation optimization. *Applied Energy*, 288:116608, 2021.

[89] W. C. Wong, E. Chee, J. Li, and X. Wang. Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing. *Mathematics*, 6(11): 242, 2018.

[90] Z. Wu, D. Rincon, and P. D. Christofides. Real-time adaptive machine-learning-based predictive control of nonlinear processes. *Industrial & Engineering Chemistry Research*, 59(6):2275–2290, 2019.

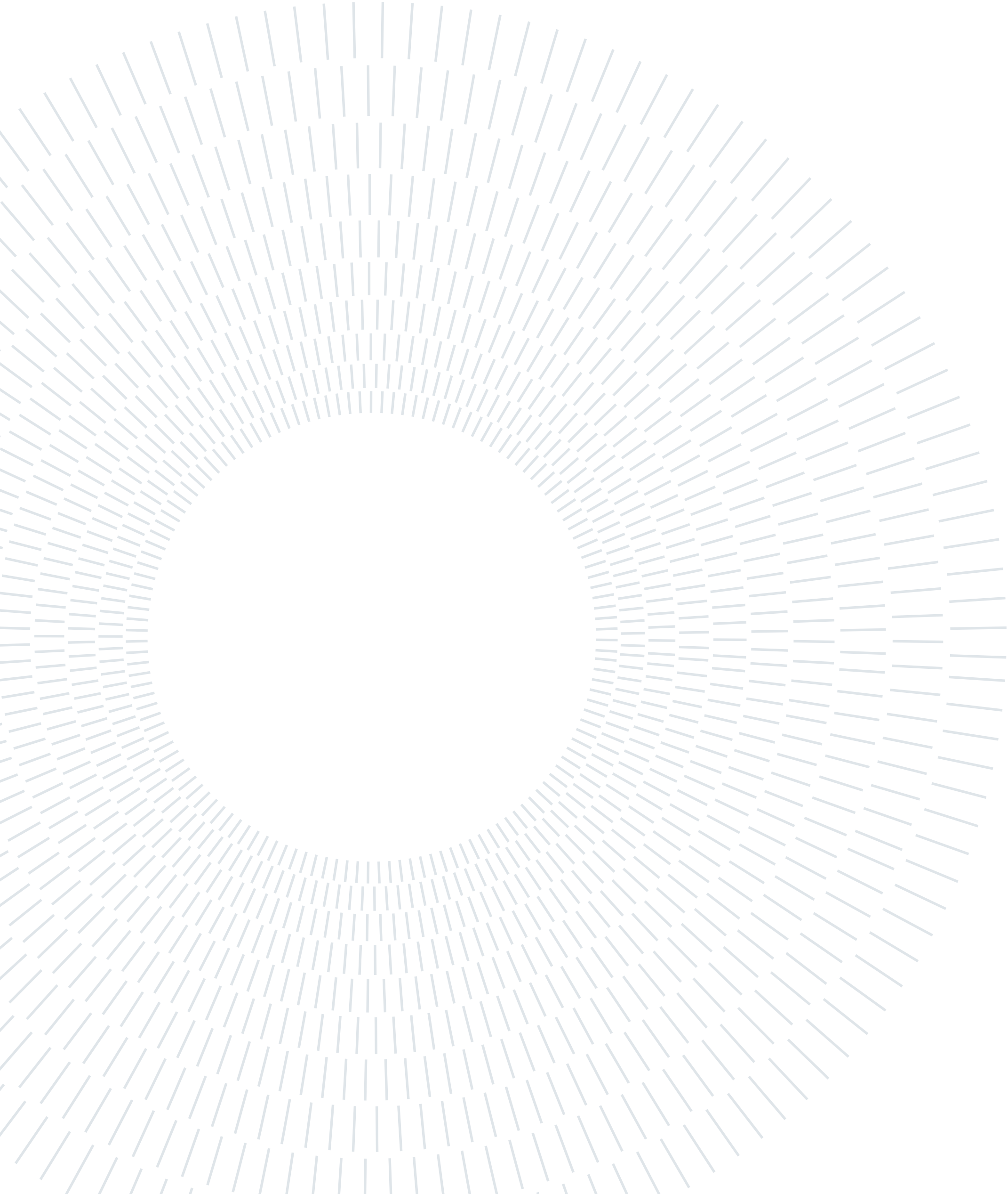
[91] Z. Wu, D. Rincon, and P. D. Christofides. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. *Journal of Process Control*, 89:74–84, 2020.

[92] Y. Xu and R. Goodacre. On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of analysis and testing*, 2(3): 249–262, 2018.

[93] B. Zhang, X. Sun, S. Liu, and X. Deng. Recurrent neural network-based model predictive control for multiple unmanned quadrotor formation flight. *International journal of aerospace engineering*, 2019:1–18, 2019.

[94] Y. Zheng and Z. Wu. Physics-informed online machine learning and predictive control of nonlinear processes with parameter uncertainty. *Industrial & Engineering Chemistry Research*, 2023.

[95] Z. Zheng, P. Luo, Y. Li, S. Luo, J. Jian, and Z. Huang. Towards lifelong thermal comfort prediction with kubeedge-sedna: online multi-task learning with metaknowledge base. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, pages 263–276, 2022.



List of Figures

1 CO_2 emissions by region. 1
2 A schematic example of a district heating system [44]. 3
3 A schematic description of physics-informed machine learning use of scientific knowledge and data with respect to standard methods [39]. 5

1.1 The gas boiler representation in the *DHN4Control* Modelica library. 10
1.2 The differential pressure pump in the *DHN4Control* Modelica library. . . . 12
1.3 The expansion vessel in the *DHN4Control* Modelica library. 13
1.4 Finite volume method: discretisation of a pipe with n sections. 13
1.5 A generic pipe in the *DHN4Control* Modelica library. 14
1.6 A generic consumer in the *DHN4Control* Modelica library. 14
1.7 Valve proportional controller implemented in Modelica. 15
1.8 AROMA network schematic representation: forward-flow arcs are plotted in solid red, backward-flow arcs in dashed blue, users in green and the heating station in dotted yellow. Nodes in the forward part are referred to as Fi, in the return part as Ri. 16
1.9 The complete AROMA network implemented in Modelica. 17
1.10 Daily simulation inputs. 19
1.11 Main AROMA network outputs in a daily simulation: in blue the first user's variables, in purple the second's, in red the third's, in green the fourth's, in yellow the fifth's and in light-blue the return variables. 20

2.1 Example of noisy inputs. 25
2.2 Input and output variables of a 10-day simulation, used for *training*: in blue the first user's variables, in purple the second's, in red the third's, in green the fourth's, in yellow the fifth's and in light-blue the return variables. 26
2.3 Input and output variables of a daily simulation, used for *testing*: in blue the first user's variables, in purple the second's, in red the third's, in green the fourth's, in yellow the fifth's and in light-blue the return variables. . . . 27
2.4 Polynomial models best identification results: in black the ground truth, in green the ARX model's result, in blue the SS' and in red the OE's. . . . 32
2.5 A schematic representation of an illustrative RNN. The input layer (6 inputs) is depicted in light-blue, the two hidden layers (7 neurons each) are displayed in green and the output layer (5 outputs) is depicted in orange. . 36

2.6 Variables identified by the best GRU (red) compared to the real ones (blue). 40

2.7 ARX and GRU results comparison. 42

2.8 Daily simulation: variables identified by a GRU (red) compared to the real ones (blue). 43

3.1 A schematic representation of the GRU implemented both with the traditional loss function and with the constrained one. Inputs are highlighted in light-blue, outputs in orange. 50

3.2 An intuitive representation of how the i^{th} load and variables of the physical system are turned into a GRU model. Its inputs are depicted in light-blue, the outputs in orange. The user is represented in green, the forward pipe in red and the return line in blue. 51

3.3 AROMA network scheme representing the forward-flow part: arcs are plotted in red, users in green and nodes are referred to as Fi. 52

3.4 Scheme representing the impact of each load on the others in the AROMA network, in terms of supply temperature. Users (GRUs) are depicted in green, supply in red. Each GRU connection has a different line style to better visualize the dependence. 53

3.5 Schematic physics-based RNN having as input, besides the supply temperature, the single P_i. Inputs are depicted in light-blue, outputs in orange. Being T_i^s an output for the i^{th} load and an input for the subsequent one, it is represented both in light-blue and in orange. 54

3.6 Schematic physics-based RNN having as input P_i and the summation of the other load profiles. Inputs are depicted in light-blue, outputs in orange. Being T_i^s an output for the i^{th} load and an input for the subsequent one, it is represented both in light-blue and in orange. 55

3.7 Schematic physics-based GRU used to hard-code the whole return network. Inputs are depicted in light-blue, outputs in orange. The index j stands as usual for $j = \{1, ..., n_{loads}\}$. 56

3.8 Scheme of the implemented physics-based RNN, highlighting the six GRUs and their input and output variables. Inputs are depicted in light-blue, outputs in orange. Being T_i^s an output for the i^{th} load and an input for the subsequent one, it is represented both in light-blue and in orange. . . . 57

3.9 Comparison between the FIT trend of a traditional RNN (red) and the FIT trend of a physics-based RNN (yellow). The target FIT is represented in blue. 63

3.10 Variables identified by a PB-GRU (red) compared to the real ones (blue). . 65

3.11 Comparison between the FIT trend of a traditional RNN (red) and the FIT trend of a physics-based one (yellow), both having 54 states. The target FIT is represented in blue. The model samples are collected every minute. 68
3.12 Comparison between the FIT trend of a traditional RNN (red) and the FIT trend of a physics-based one (yellow), both having 54 states. The target FIT is represented in blue. The model samples are collected every five minutes. 70
3.13 Comparison among the FIT trend of a traditional RNN trained with 15200 samples (red), of a PB-RNN trained with 15200 samples (yellow), of a traditional RNN trained with 3040 samples (blue) and of a PB-RNN trained with 3040 samples (purple). The model samples are collected every five minutes. 71
3.14 Comparison between the FIT trend of a traditional RNN (red) and the FIT trend of a PB-RNN (yellow), both trained with 3040 samples. The model samples are collected every five minutes. The training was stopped after 600 epochs because over-fitting occurred. 72

4.1 Scheme of the control algorithm, including the optimization part (MPC), the observer and the plant. 76
4.2 Comparison between the daily disturbance forecast (red) and the actual power demand (blue). 79
4.3 Daily electrical cost. Data extracted from an Italian average of January 2023. 83
4.4 Schematic representation of the blocking strategy. N is the prediction horizon, N_b is the number of steps in which the control variable u(k) is blocked. 87
4.5 PI-GRU-based MPC optimization results. When more variables are plotted in the same graph, the first user's variables are represented in blue, in purple the second's, in red the third's, in green the fourth's, in yellow the fifth's and in light-blue the overall return temperature (d). The constraints are depicted in black. The large boundaries of the boiler power (b) are not displayed to have a better close-up. 89
4.6 Standard GRU-based MPC optimization results. When more variables are plotted in the same graph, the first user's variables are represented in blue, in purple the second's, in red the third's, in green the fourth's, in yellow the fifth's and in light-blue the overall return temperature (d). The constraints are depicted in black. 91

5.1 Example of the normal distribution of a system variable error: T_5^s plant-model mismatch. The number of observations is 2000, i.e. around seven days sampled every five minutes. 98

5.2 Schematic representation of the ellipsoidal confidence region of T^2 in case of two variables. 99

5.3 Comparison between some AROMA network variables in the case of old plant (blue) and new plant (red). 105

5.4 Comparison among the variables identified by the old PB-GRU (red), by the new PB-GRU (yellow) and the measured ones (blue), in case of a novel plant. 109

5.5 Comparison between a daily overall standard power consumption (blue) and an abnormal one (red). 113

5.6 Schematic representation of the additive network structure. The incremental part is denoted by ΔPB-GRU and its objective is to identify the model-plant mismatch, i.e. $\Delta\hat{y}$. Altogether, this term is added to the original output estimation $\hat{y}$ so as to find the actual predicted output $\hat{y}_{new}$. 115

5.7 Additive network identification results. The error between the measured supply temperature of the second user and the one predicted by the old PB-GRU (blue) is compared to the error identified by the additive PB-GRU (red). 116

5.8 Comparison among the variables identified by the old PB-GRU (red), by the new overall PB-GRU (yellow) and the measured ones (blue), in case of abnormal operating conditions. 117

5.9 Comparison among the variables identified by the old PB-GRU (red), by the new overall PB-GRU (yellow) and the measured ones (blue), in case of normal operating conditions. 119

List of Tables

1.1 Weights of the AROMA network describing how the overall power request is partitioned among the five consumers. 18

2.1 Polynomial models FIT comparison. 33
2.2 Performance comparison among different models of RNN. 39
2.3 ARX and GRU FIT comparison. 41

3.1 Performance of a GRU with constrained loss function compared to a traditional one. 50
3.2 Comparison between the two physics-based approaches having different inputs. 55
3.3 Comparison of the AROMA network identified with standard RNN and PB-RNN. 63
3.4 Comparison of the AROMA network identified with standard RNN and PB-RNN, using a restricted amount of neurons per layer. The model samples are collected every minute. 68
3.5 Comparison of the AROMA network identified with standard RNN and PB-RNN, using a restricted amount of neurons per layer and 15200 samples. The model samples are collected every five minutes. 70
3.6 Comparison of the AROMA network identified with standard RNN and PB-RNN, using a smaller amount of data (3040 samples). The model samples are collected every five minutes. 71
3.7 Comparison of the AROMA network identified with standard RNN (best configuration) and PB-RNN, using a smaller amount of data. The model samples are collected every five minutes. 73

4.1 Main optimization variables. 78
4.2 Main optimization parameters. 86
4.3 Comparison between the performance of PI-RNN-based MPC and standard RNN-based MPC. The electrical cost, the average and maximum resolution time per iteration, together with the total resolution time, are reported for the two models. 90

5.1 Comparison of $RMSE$ in three different cases for each output variable. The thresholds are highlighted in light-blue, whereas the values exceeding the corresponding thresholds are highlighted in red. 102
5.2 Comparison of Mahalanobis distances in three different cases. The thresholds are highlighted in light-blue, whereas the values exceeding the corresponding thresholds are highlighted in red. 102
5.3 Comparison of the R^2 values between the variables identified by the original and by the new model. 110
5.4 Comparison of the R^2 values and computational times between the traditional MHE problem and the updated one. 111
5.5 Comparison of the R^2 values between the variables identified by the original and by the new overall network, in case of abnormal operating conditions. . 118
5.6 Comparison of the R^2 values between the variables identified by the original and by the new overall network, in case of normal operating conditions. . . 119
5.7 List of parameters used in the thesis 142
5.8 List of variables used in the thesis 143

List of Parameters

Symbol	Description	Value	SI unit
c_p	Water specific heat	4186	[J/kg K]
d_{ins}	Pipe insulation thickness	0.2	[m]
d_{wall}	Pipe metal wall thickness	0.003	[m]
D_j^{pipe}	Pipe internal diameter	*	[m]
k^{PID}	Boiler PID gain	$5 \cdot 10^4$	-
k^P	Load proportional gain	0.05	-
K_v	Flow factor	36	m^3/h
f	Fanning friction coefficient	0.004	-
λ_s	Steel thermal conductivity	45	[W/m K]
L_j^{pipe}	Pipe length	*	[m]
M_w	Mass of water inside the mixing volume	160	[kg]
n_{load}	Number of loads in AROMA network	5	-
P_{min}^{boiler}	Minimum boiler power	1	[kW]
P_{max}^{boiler}	Maximum boiler power	10	[MW]
P_{min}^{load}	Minimum load power	30	[kW]
P_{max}^{load}	Maximum load power	420	[kW]
p_{pump}	Differential pump pressure	5	[bar]
p_{vess}	Vessel fixed internal pressure	5	[bar]
Re	Reynolds number	1287	-
ρ	Water density	997	[kg/m^3]
ρ_s	Steel heat capacity per unit volume	$3.12 \cdot 10^6$	[J/m^3 K]
σ_{ins}	Pipe insulation thermal conductivity	0.02	[W/m K]
θ_{min}	Valve minimum opening area diameter	0.001	[m]
θ_n	Nominal valve opening	1	-
T_i^{PID}	Boiler PID integral time	16	[s]
T_{ext}	External temperature	25	[°C]
T_{ref}	Load reference temperature	65	[°C]
T_s	TBPTT input-output subsequences length	200	-

$T_{sampling}$	Sampling time	{60,300}	[s]
u_{nom}	Nominal fluid velocity	1.5	[m/s]
UA_j^{pipe}	Thermal conductance pipe-ambient	*	[W/K]
U_j^{pipe}	Heat transfer coefficient of the j^{th} pipe wall	*	[W/m^2 K]
V	Volume of water inside the boiler	0.16	[m^3]

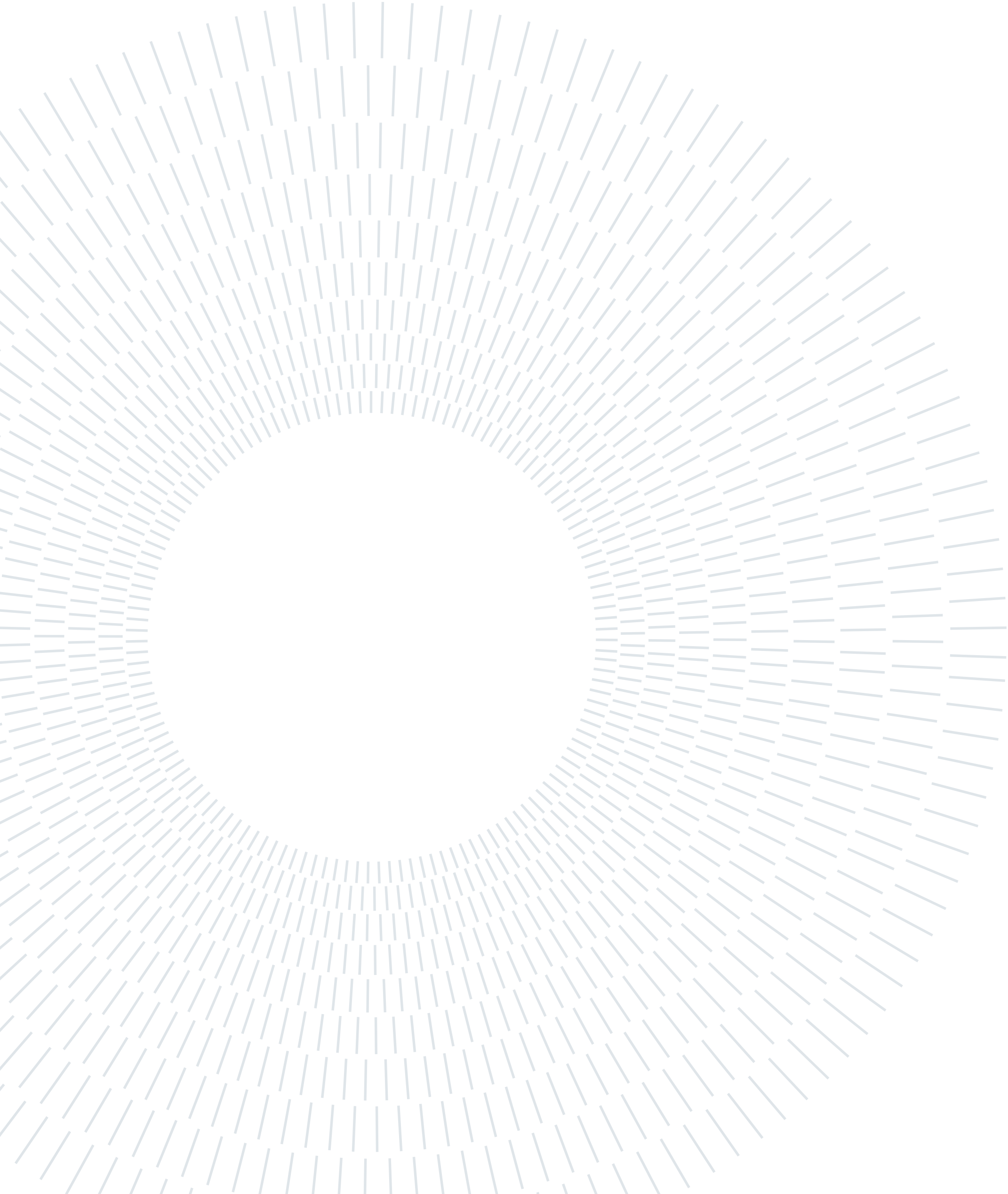
Table 5.7: List of parameters used in the thesis

* It depends on the considered pipe (j^{th}): the values are reported in [43].

List of Variables

Symbol	Description	SI unit
Δp^{pump}	Pressure drop between pump ports	[bar]
$\dot{m}_w$	Mass flow rate of water inside the mixing volume	[kg/s]
$\dot{m}_i$	Return mass flow rate of the i^{th} load	[kg/s]
$\dot{m}_{in}^{pump}$	Mass flow rate of water entering the pump	[kg/s]
$\dot{m}_{out}^{pump}$	Mass flow rate of water leaving the pump	[kg/s]
$\dot{m}_r$	Overall return mass flow rate	[kg/s]
P_i^{load}	Power of the i^{th} load	[W]
P^{boiler}	Boiler power	[W]
p_{in}^{pump}	Input pressure of the pump	[bar]
p_{out}^{pump}	Output pressure of the pump	[bar]
Q_{amb}	Heat losses to ambient	[W]
ρ_j^{pipe}	Water density in the j^{th} pipe	[kg/m^3]
T_i^s	Supply temperature of the i^{th} load	[K]
T_i^r	Return temperature of the i^{th} load	[K]
T^r	Overall return temperature	[K]
T_j^{pipe}	Water temperature in the j^{th} pipe	[K]
T_{in}	Temperature of water entering the mixing volume	[K]
T_{out}	Temperature of water leaving the mixing volume	[K]
T_{in}^{pump}	Temperature of water entering the pump	[K]
T_{out}^{pump}	Temperature of water leaving the pump	[K]
T_{out}^{boiler}	Boiler outlet temperature	[K]
T_{ref}^{boiler}	Boiler reference temperature	[K]
T_w	Temperature of water inside the mixing volume	[K]
τ	Mixing volume time constant	[s]
u_j^{pipe}	Flow velocity in the j^{th} pipe	[m/s]

Table 5.8: List of variables used in the thesis



List of Acronyms

Acronym	**Meaning**
ADAM	Adaptive Moment Estimation
ARMA	AutoRegressive Moving Average
ARMAX	AutoRegressive Moving Average with eXternal input
ARX	AutoRegressive with eXternal input
BJ	Box-Jenkins
CHP	Combined Heat and Power
DHN	District Heating Network
DHS	District Heating System
EKF	Extended Kalman Filter
EPANN	Evolved Plastic Artificial Neural Network
FHCOP	Finite-Horizon Control Optimization Problem
FIR	Finite Impulse Response
GRU	Gated Recurrent Unit
Ipopt	Interior Point Optimizer
KGML	Knowledge-Guided Machine Learning
LSTM	Long Short-Term Memory
MISO	Multiple Input Single Output
ML	Machine Learning
MPC	Model Predictive Control
MSE	Mean Square Error
N4SID	Numerical algorithm For Subspace State-Space System Identification
NLP	Non-Linear Programming
NMPC	Non-linear Model Predictive Control
NN	Neural Network
NRMSE	Normalized Root Mean Square Error
OE	Output-Error
PB-ML	Physics-based Machine Learning
PID	Proportional-Integral-Derivative

PI-RNN	Physics-Informed Recurrent Neural Network
RMSE	Root Mean Square Error
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
SS	State-Space
TBPTT	Truncated Back-Propagation Through Time
TES	Thermal Energy Storage
WGN	White Gaussian Noise

Acknowledgments

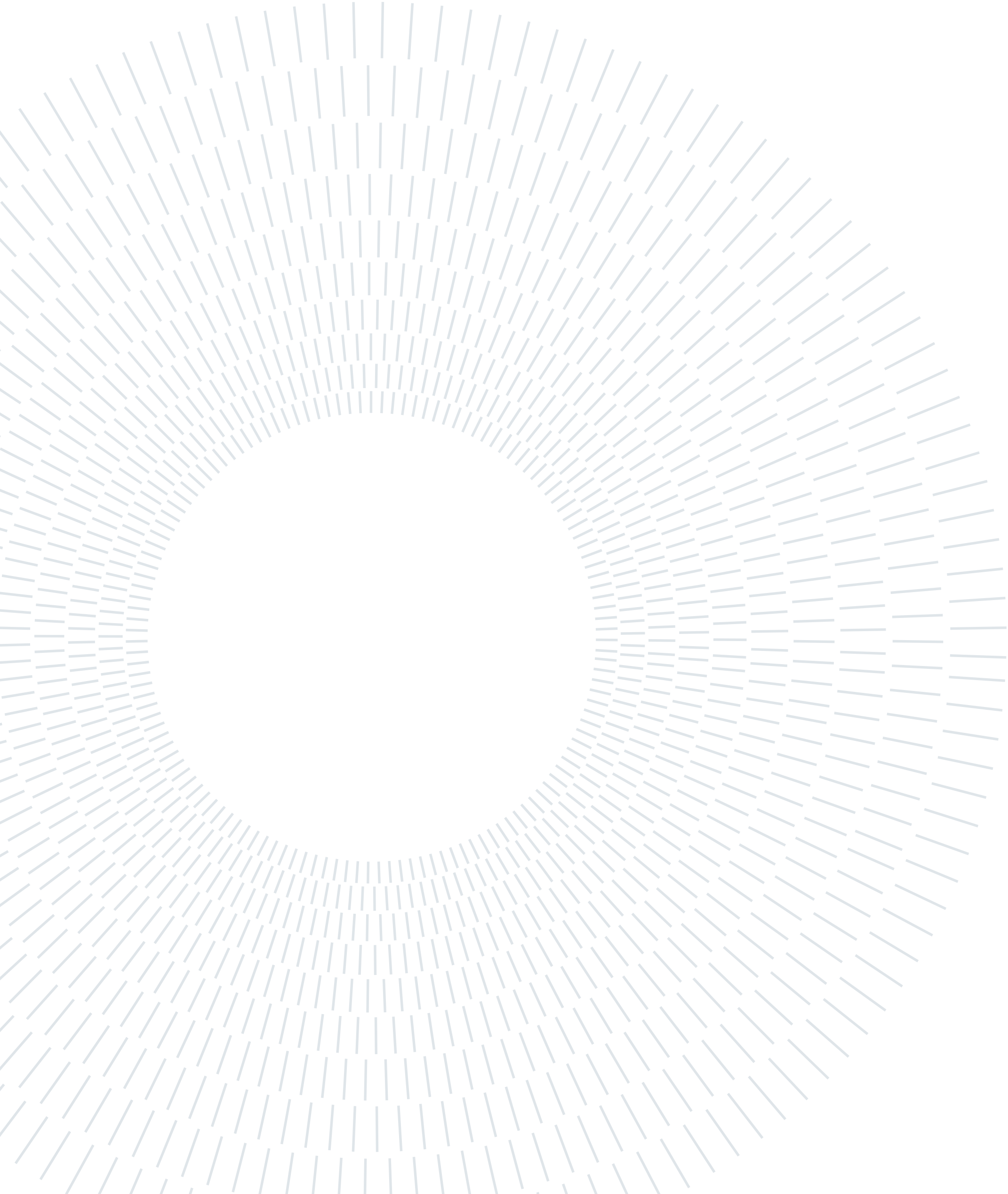
La gratitudine per questo traguardo di un percorso così ricco è incommensurabile. Sono tante le persone che mi hanno accompagnato e a cui devo un particolare riconoscimento.

In primis ringrazio il Professor Scattolini che mi ha dato l'opportunità di lavorare a questo progetto e che mi ha seguito e guidato durante lo svolgimento con professionalità ed esperienza. Sono molto grata per la fiducia che ha riposto in me, sia per l'esecuzione della tesi sia per le possibili esperienze future.
Un ringraziamento particolare va senza dubbio al Professor La Bella, il cui aiuto e supporto sono stati fondamentali in questi mesi e grazie al quale ho imparato inestimabili lezioni. Con la sua preparazione e pazienza è stato una significativa fonte di consiglio e sostegno. Un sincero grazie va anche a Fabio Bonassi, Ph.D. e al Dott. Lorenzo Nigro, che mi hanno aiutata con le loro preziose competenze a rendere questo lavoro più sostanzioso.

Ringrazio immensamente i miei genitori e mio fratello per avermi sempre sostenuta in tutte le mie scelte e fornito gli strumenti idonei per affrontarle. Il loro supporto è stato di fondamentale importanza: devo loro buona parte del merito dei miei traguardi. Un affettuoso ringraziamento anche alle mie nonne, zii e cugini che sono i miei fan numero uno e il cui incoraggiamento è sempre fonte di orgoglio.

Uno speciale ringraziamento va sicuramente ad Anna, Clarisse, Cassolo, Sonia e Capri, che durante gli anni universitari da fuorisede sono state una seconda famiglia per me (o meglio, delle *sorelle*). Sono immensamente grata per aver trovato in loro non solo delle preziose amiche, ma anche costanti fonti di ispirazione, consiglio e soprattutto di ingenti risate e goliardate. Grazie a loro vivere a Milano è stata una delle esperienze più arricchenti e significative della mia vita. Un generico grazie va anche a tutte le altre speciali persone con cui ho condiviso per cinque anni la stravagante vita collegiale.

Ringrazio Ali, Mati e Virgia perché, nonostante la lontananza, sono da tanti (troppi!) anni sempre al mio fianco nei momenti più importanti, divertenti ma anche complessi.

Un finale ringraziamento va all'ambiente galvanizzante del Politecnico e a tutte le persone che lo compongono, in particolar modo ai miei compagni di università, dalle gestionali agli automatici. Grazie a loro affrontare le sfide universitarie è stato non solo meno greve ma anche più piacevole e stimolante. Infine, ringrazio le brillanti persone incontrate a Losanna, grazie alla cui amicizia e ilarità l'Erasmus è stato un'esperienza decisamente formativa e *peculiare*.