Real-time Virtual Sensor for NO_x emisions and stoichiometric air-fuel ratio λ of a Marine Diesel Engine Using Neural Networks

Pantelis Dimitrakopoulos

MSc Thesis

School of Naval Architecture and Marine Engineering
National Technical University of Athens

Supervisor: Assistant Prof. George Papalambrou

Committee Member : Prof. N. Kyrtatos
Committee Member : Associate Prof. C. Papadopoulos

November 2019

Acknowledgements

This work has been carried out at the Laboratory of Marine Engineering (LME) at the School of Naval Architecture and Marine Engineering of the National Technical University of Athens, under the supervision of Assistant Professor George Papalambrou.

I would like thank Professor Nikolaos Kyrtatos for providing the opportunity to work with the full-scale hybrid diesel-electric marine propulsion powertrain of LME. Access to LME experimental facilities was essential in order to verify theoretical concepts from a practical perspective, gather valuable data and evaluate experimentally the designed models. I also thank him for being a member of my supervisors committee.

My greatest appreciation is owed to my thesis supervisor Assistant Professor George Papalambrou, for giving me the chance and motivation to work on the artificial intelligence topic for marine engines. I would also like to thank him for his patience and persistent support. His spirit motivated me all this time to carry out this thesis. The door to his office was always open whenever I ran into a trouble spot or had a question about my research or writing.

I would also like to thank Associate Professor Christos Papadopoulos for evaluating my work and being a member of my supervisors committee.

It is imporant to express my sincere gratitude to Mr. Nikolaos Planakis, researcher of LME for his contribution to unlock several milestones. I am extremely thankful to him for sharing expertise, and sincere and valuable guidance and encouragement extended to me.

Finally, I must express my very profound gratitude to my girlfriend, my classmates, my colleauges at Gamma Technologies and my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Abstract

λ value and NO_x mass species have been proven important variables to emissions control and reduction for marine diesel engines. Models for these quantities can substitute real sensors that are often cost ineffecient and faulty. On top of that, model based control for emissions must operate on the basis of fast and accurate models. Artificial neural networks (ANN) are data based models that calculate predictions on relatively simple operations using non-linear functions. They appear in many flavors and configurations. In the present work, the time-delay neural network (TDNN) and recurrent neural networks (RNN) with inputs delay are investigated as models for λ value and NO_x variables.

Specifically, virtual sensors were developed for marine diesel engine's turbine-out NOx emisions and λ value based on raw measurements from laboratory data acquisition. A method was developed to search for the optimal neural network configuration between time-delay neural network models and recurrent neural network models with inputs delay. Model inputs are decided based on traditional thermodynamic models, such as the Zeldovich mechanism for NO_x formation, and the available quantities sensors in any in-market marine engine. The resulting models capture the non-linear phenomenon of NO_x formation and changes in λ value of the marine diesel engine. Calculations performance is fast and portability of the models is easy. The resulting neural network models are deployed on an ECU prototype machine and they are validated in real-time side by side with the real system's sensors. The validation tests include engine operating points within the training range but of different pattern. The real-time validation for the recurrent neural network models shows that their predictions stay consistent in most operating areas and the dynamic behavior of the emissions variables is captured and reproduced accurately. The recurrent neural network model for λ value was compared against a first principle physics based virtual sensor with more accurate results. Therefore, the validation proves that the RNN models generalize adequately within the training range with the minimum possible complexity. On the other hand, the time-delay neural network models are more complex and they do not exemplify the same accuracy in the validation tests, especially in unknown to them operating areas. As a result, the recurrent neural network models with input delay are suggested to be used as benchmark models for emissions control applications.

Contents

Chapter 1

Introduction

Framework

The growing environmental hazards and the never-ending strive for fuel efficient systems apply pressure to the marine diesel engine industry for optimized controls, emissions and fuel economy strategies. These strigent goals have forced manufacturers of marine engines to develop new technologies related to the propulsion power units of ships, aiming at lower exhaust emission and enhanced power management. Typical applications in new marine powertrains are exhaust gas recirculation (EGR), inteligent fuel injection with three phase common rail injection systems, cylinder de-activation strategies, selective catalytic reduction systems (SCR) with catalyst injection for emissions neutralization and others. For example, EGR systems are effective in reducing the NO_x emissions but they can be very complex as shown in the Fig. 1.1 . Another approach that manufacturers follow is the hybrid marine powertrains. Nowadays, they are widely developed and offered by most manufacturers despite the low market share as of today. The hybrid powertrain combines the combustion engine with battery powered propulsion and thus they optimize the engine operation and fuel efficiency while reducing emissions signficiantly[1].

Different types of vessels operate in various conditions. For example, marine powerplants in cargo vessels usually operate in steady state sea going condition. On the other hand, some ship types operate in coastal areas and within port range, such as passenger ships, yachts, tugs, special purpose vessels, etc. This diverse scenario of operation affect significantly the overall efficiency of the marine powertrain system. As a result often is the case where these systems operate far from the high efficiency range.

Consequently, marine diesel engines operate outside of steady-state conditions in many situations such as maneuvering, port operation or rough seas. These transient operation circumstances are the most unpredictable of a diesel engine, especially, in terms of exhaust emission and fuel consumption. During load transients, all the engine variables change continuously, deterring the engine from its high efficiency operating area. At the same time, the exhaust gas quality is drastically affected, mainly due to the delayed response of the turbo-charger to provide the sufficient amount of air on time. As a result of the lean combustion, λ value, which is an indication for the air-fuel mixture quality, drops in the area of fuel rich mixtures with a consequent rise of the pollutant emissions of the engine.

These phenomena are typically tackled by advanced and complex controls systems that control the emissions reduction systems, such as the EGR and SCR. The controls systems rely on high sampling frequencies of the controlled subsystems; high enough to produce effective re-actions that will acomplish the emissions reduction and fuel consump-

[1] https://www.man-es.com/docs/default-source/default-document-library/man-es$_l$3$_f$actsheets$_4$p$_m$an$_h$ybrid − propulsion$_p$review.pdf?sfvrsn = 267c03fb$_6$

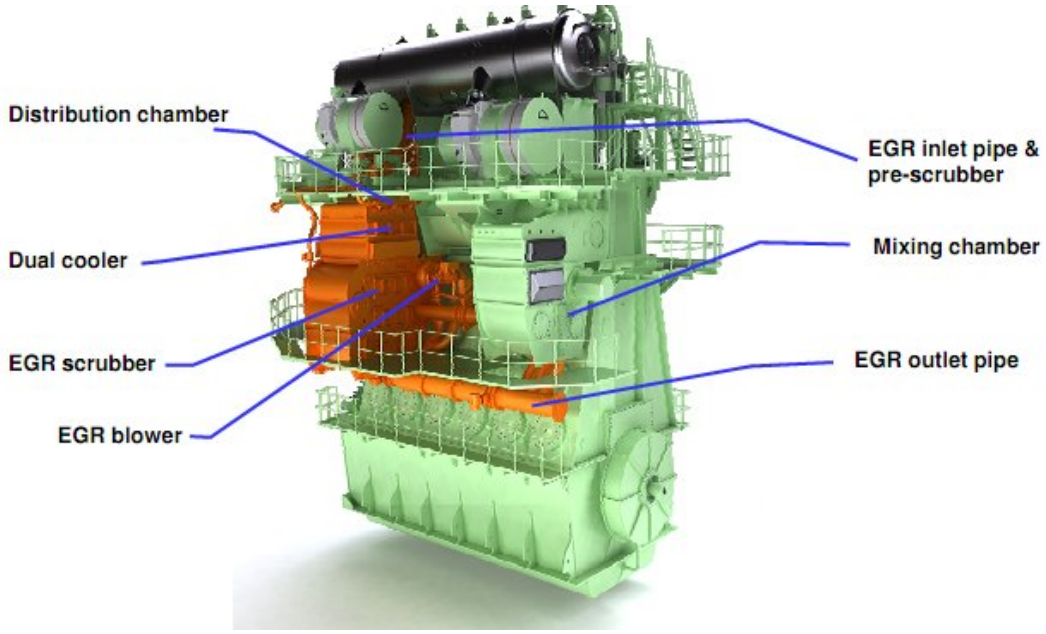

Figure 1.1: MAN NOx reduction system by exhaust gas recirculation EGR.

tion standards. In some cases the marine engine powertrain's sensor can provide the necessary information to the controls system. Although, there are cases where this does not happen in a timely manner, the sensors installation and maintance is expensive or the quantities are not measured at all in important engine positions, e.g. in-cylinder quantities. For these cases, virtual sensors can provide information with regards to the necessary engine variables.

Literature Review

Mathematical models can be used to substitute expensive and fault-prone sensors or introduce new quantities that are not traditionally measured by sensors to the control system. These models are often called virtual sensors because they measure quantities values just as any traditional sensor but the measurements are not captured by sensors. Instead the "measured" quantities are products of calculations based on already installed sensors in the system. The additional quantities facilitate the optimal control, cost-efficient system maintenance and early fault prediction or detection.

Global market researchers expected the global virtual sensors market size to grow from 235$ million in 2018 to 910$ million by 2023. The major growth drivers include predictive maintenance and potential reduction in the time and cost compared to physical sensors as well as rising adoption of the internet of things (IoT) cloud platforms. Virtual sensor solutions help organizations to estimate product properties or process conditions based on mathematical models. In a sentence, virtual sensors are mathematical models which estimate the desired values based on the inputs from other physical sensors already installed in a system. Organizations use virtual sensors mainly to achieve efficiency and in locations where physical sensors cannot be implemented. Systems such as the marine diesel powertrains are not being exempt from this trend.

As any system, marine diesel engine systems can benefit from virtual sensors for optimal emissions control or fuel consumption. Sensors upstream and downstream the cylinder provide control and diagnostic system valuable measurements in a timely manner. However, sensors installation , maintenance and replacement at their end-of-life bare significant

cost, especially in the after cylinders exhaust air path. In other cases a direct measurement by sensors might not be even possible or just too expensive to justify the sensor installation in the first place. On the other hand, a virtual sensor can be developed and used for the estimation of the unmeasured engine parameters. Such sensors are based either on first principles or measured data. As shown before, neural network models are data based and require measurement data for their training in the direction of minimizing their predictions errors. Additionally they also require the existence of pre-installed sensors in order to use the measured data as inputs for their calculations during real conditions operation. Commercial marine engines are equipped with sensors and diagnostics that can cover these requirements. Therefore, model-based virtual sensors can be harnessed for replacing costly traditional sensors or even predict faster and more accurate quantities that were not measured directly, e.g. the λ value. Such models can reproduce the benchmark system behavior for a controls strategy or the on-board diagnostics systems. In the present work, models of NO_x emissions and λ value are developed using the two neural network model permutations presented in the following chapters, the time-delay neural network (TDNN) and the recurrent neural network (RNN). Each one of the neural network types is evaluated as a virtual sensor for real time operation.

A virtual sensor for marine engine powertrain variables should enable reliable system operation by enhancing the diagnostics and fault-detection control system with more and accurate information. Map-based approaches have been used widely in the literature. Their benefit is that they can execute seamlessly fast but they produce poor results during transient phenomena [4]. Marine diesel engines often operate outside of steady-state conditions. Therefore a model should be able to capture the non-linear phenomena during the transient operation which occurs typically during maneuvering and operation within the port range, where strict emissions regulations apply. Traditional flow and thermodynamic differential equations and semi-empirical models are capable of reproducing adequately both steady state and transient phenomena. Some NO_x formation models, such as the zero-dimensional Zeldovich mechanism, can execute relatively fast but they take into account combustion quantities that are rarely available or computationally expensive to calculate via combustion models [1]. These models might require data to calibrate their results to a specific marine engine. In the same principle, measurement data often show correlation between most marine engine quantities of interest. When there are available sufficient measurements data, data-driven models such as artificial neural networks (ANN) can be used to describe the relationship between the various quantities. A substantial number of neural networks permutations have been validated as virtual sensors in the automotive, railway and aerospace industry [8] [9] [10] [12]. Specifically, in [6], neural network models are developed successfully for the diesel engine-out emissions, including soot, in static and transient conditions for assisting advanced model-based engine control strategies, such as the economic model predictive control (eMPC). [13] have put to the same test multiple network architectures and validated that the neural networks with inputs delay and derivative inputs provide good predictions for NOx emissions of CI engines and generalize adequately within the operating range. Additionally recurrent neural network (RNN) models have been identified and validated for virtual sensing of NO emissions in spark ignition (SI) internal combustion engines (ICE) with an cumulative estimation error lower than 2% throughout transient operation [7]. Methodologies have been propose to satisfy reliable on-board installation of advanced virtual sensors validating results with a production automotive diesel engine [9]. Installed virtual sensors, such as RNN, models have been proven sustainable by adjusting their predictions with least square technique to account for an engine's aging [14]. More complex neural network architectures have been validated for multiple target recognition in the field of intelligent

driving by learning historical visual semantics and target position information [11]. As a result, neural networks models, especially those which take into consideration previous states of the simulated system such as the TDNN and RNN architectures are capable of producing accurate predictions. For powertrain system, such models have been validated for reproducing diesel engines quantities successfully.

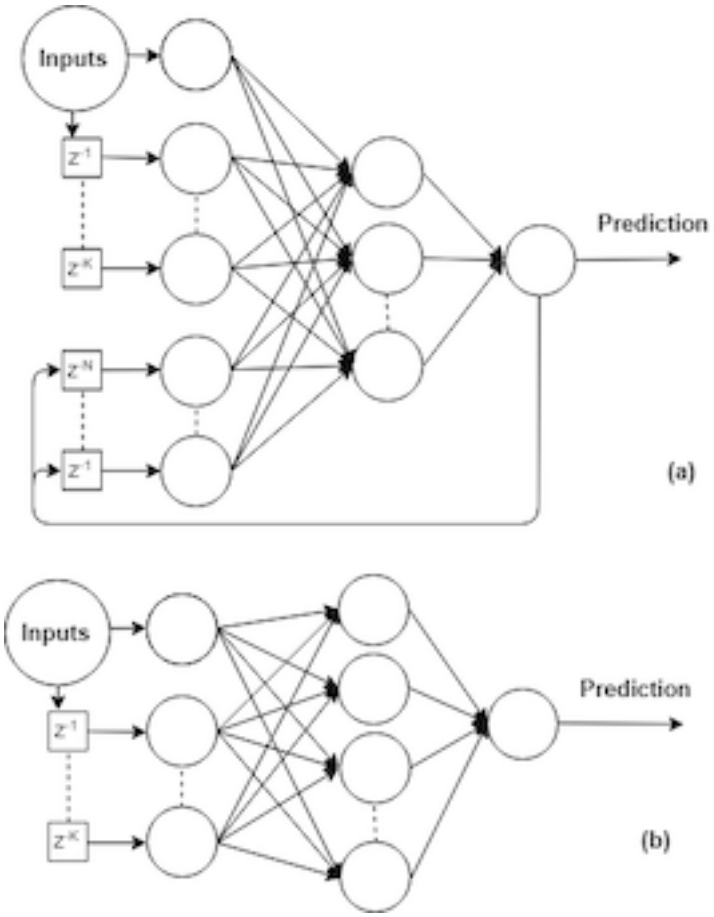

Figure 1.2: Recurrent NN (a) and Time delay NN (b).

Thesis Structure

In this thesis, two neural network model structures are investigated for modeling emissions of the Hybrid-electric powertrain of Laboratory of Marine Engineering (LME). For this investigation it was developed a method that relies on engineering principles for the models inputs selection and the powertrain identification and uses machine learning industry best practices for the neural network design [25]. The quantities modeled are the NO_x and λ value downstream the turbine. The two model structures investigated are the time delay neural networks (TDNN) and the recurrent neural networks (RNN) because they take into account not only their input signals but also the powertrain's previous states to calculate their predictions, which is necessary to produce accurate predictions for the non-linear NO_x formation.

The structure of the thesis is as follows: in chapter 2 the experimental facility is presented, along with the installed engine sensors and the data acquisition system. chapter 3 describes the modeling principles from the engineering and the data scientist angle or, in other words, it is described the development of the models with regards to the input

signals selection and the description and optimal tuning of their hyper-parameters respectively. chapter 4 refers to the method designed to develop the neural network models and the selection of the best models based on the training data. chapter 5 includes the model training results and the selected models to be used for the real-time validation. chapter 6 includes the methodology of deploying the models on a prototyping ECU board for real time validation and the results of the validation tests. Finally, the conclusions of this work are presented in chapter 7.

Chapter 2

Experimental Facility

The hybrid diesel-electric power plant is named as HIPPO-1 for the LME and consists of a internal combustion engine (ICE) in parallel connection to an electric motor (EM). In this configuration the rotational speed of the ICE and the EM are identical and the supplied torques add together to maintain the total torque demand applied by a hydrodynamic water brake (WB). In Fig. 2.1 and Fig. 2.2 the experimental hybrid powertrain of LME is presented, along with a schematic representation in AUTOCAD.

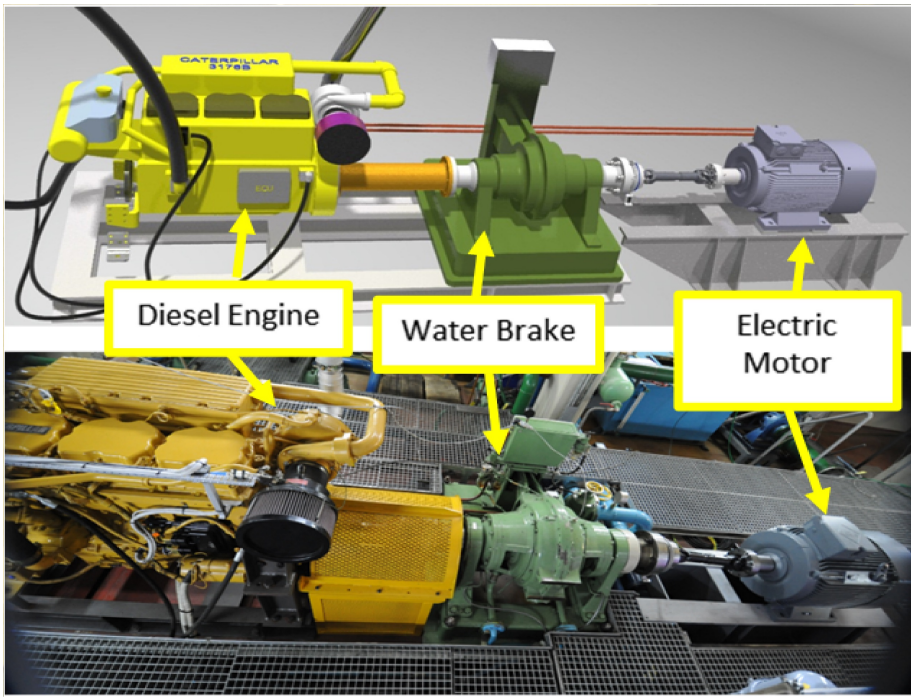

Figure 2.1: The HIPPO-1 hybrid diesel-electric testbed of LME.

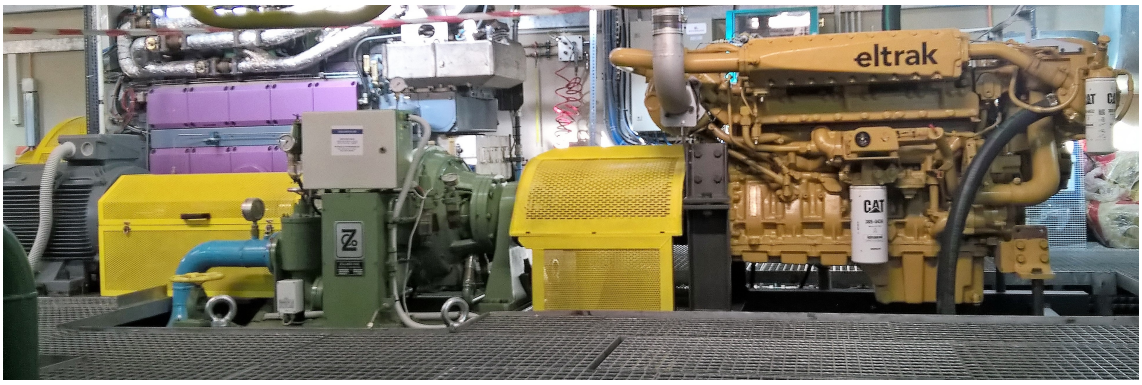

Figure 2.2: The HIPPO-1 hybrid diesel-electric testbed of LME. Between the internal combustion engine (right) and the electric motor (left) stands the water brake next to its controller board

2.1 Mechanical Componets

The ICE is a turbocharged CATERPILLAR 6-cylinder 10.3-liter 4-stroke marine diesel engine, model 3671B, producing 425 kW at 2300 rpm. According to the speed reference and the deviation of the speed measurement, the electronic control unit (ECU) of the ICE controls the fuel injection in the cylinders in closed loop control, using controller in the form of look-up tables. The installed sensors in the diesel engine are presented in Fig. 2.3

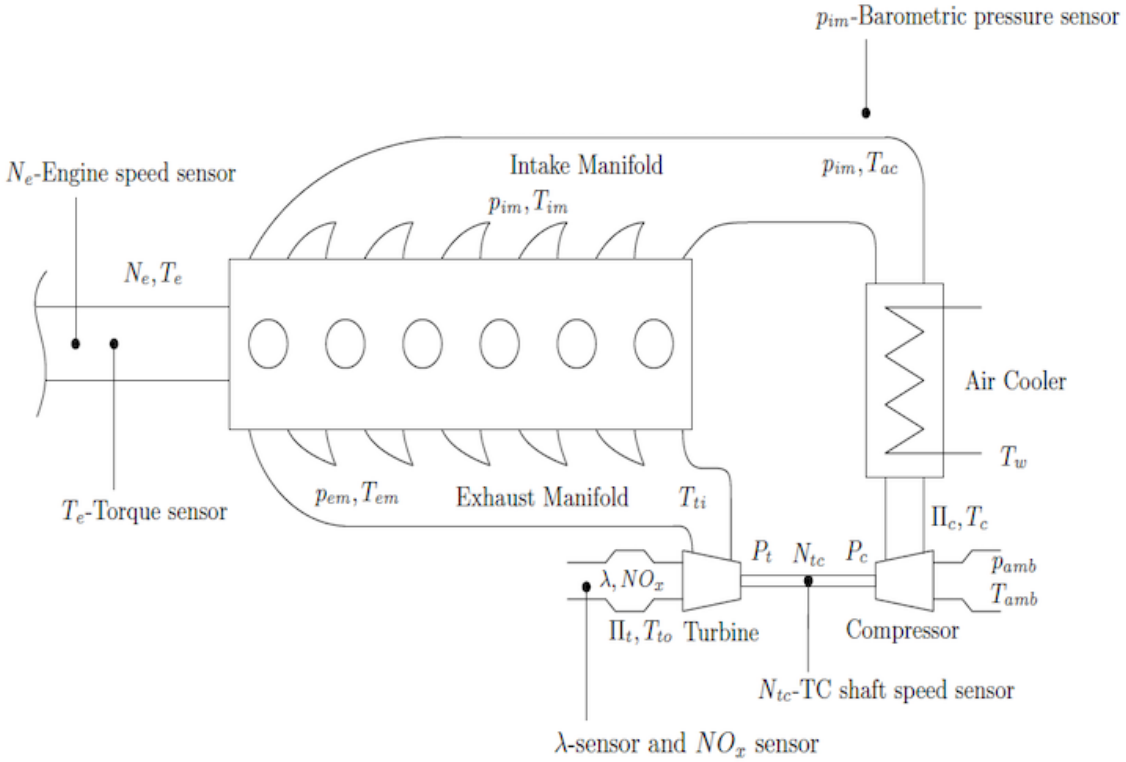

Figure 2.3: The HIPPO-1 diesel engined installed sensors.

The water brake of HIPPO-1 installation is manufactured by AVL Zoellner Gmbh, type 9n 38F, with 1200 kW load capacity, operating up to 4000 rpm. The water brake consists of two parts, the stator and the rotor, which is driven by the engine shaft. Between

the two WB parts, the water level is regulated in order to produce the requested torque demand. The WB is controlled by a H_∞ controller designed at LME[1].

2.2 Sensors and Data Acquisition System

The installed sensors in the diesel engine are presented in Fig. 2.3. The NOx and λ values are provided by a *SmartNOx* sensor in the manifold downstream of the turbocharger (TC), manufactured by NGK. Exhaust gas opacity is measured by a AVL 439 opacimeter in the exhaust duct of the CAT engine. Fuel mass flow measurements are provided by two ABB Coriolis flow-meters, one at supply and one at return fuel lines. TC speed and intake manifold pressured are also measured.

The platform for the Data Acquisition and control of the powertrain is based on the dSpace DS1103 (Fig. 2.5) controller board, with rapid control prototyping capability, programmed under the MATLAB/Simulink environment.

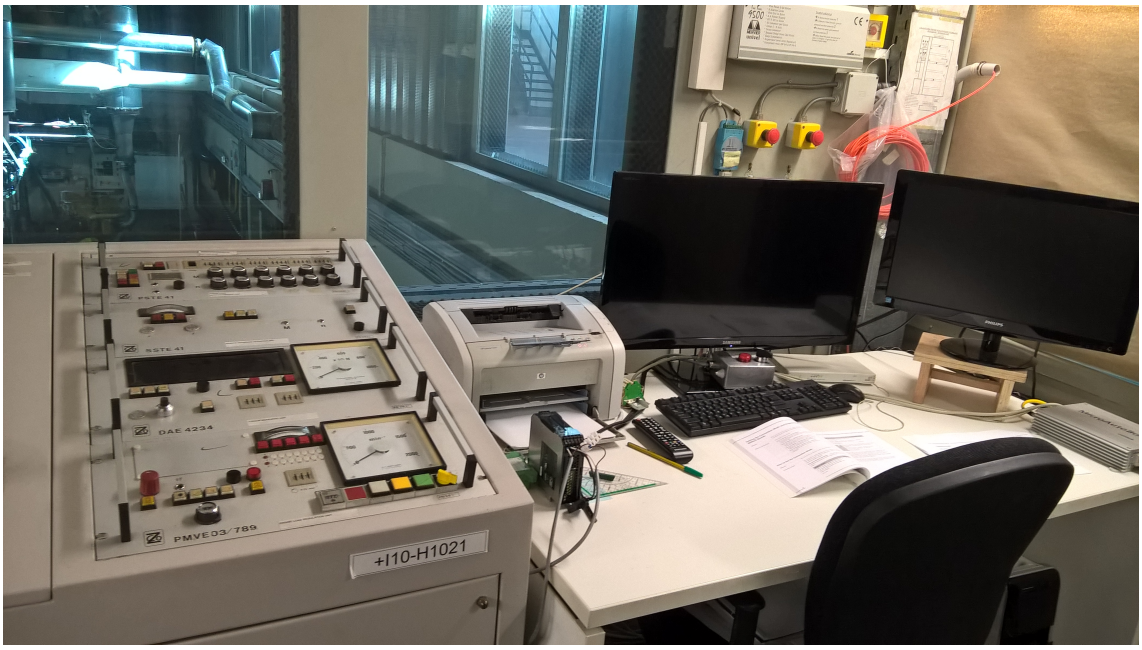

Figure 2.4: The engine control room. WB control board in front and the monitoring system of the hybrid plant on the right. Behind the safety glass, HIPPO-1 powertrain can be distinguished

A picture of the powertrain monitoring screen in the control room (Fig. 2.4) at LME is shown in Fig. 2.6, where all the utilities of the monitoring and the control board are presented.

[1]C. Gkerekos. 2015. Experimental Modeling and Robust Controller Design for the Transient Loading of a Marine Diesel Engine. *Diploma Thesis*

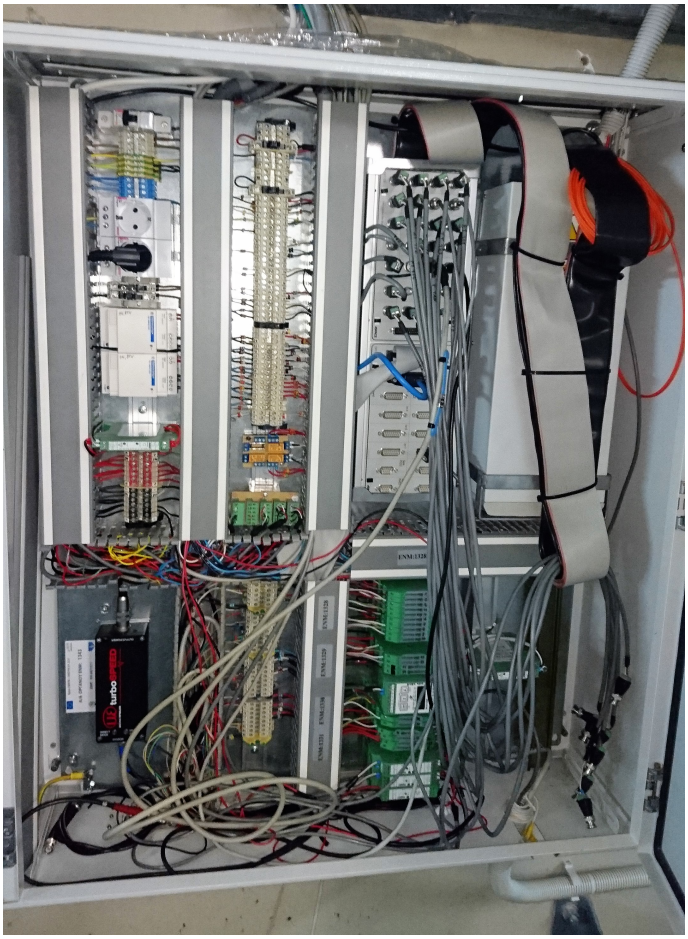

Figure 2.5: The HIPPO-1 dSpace monitoring and control board.

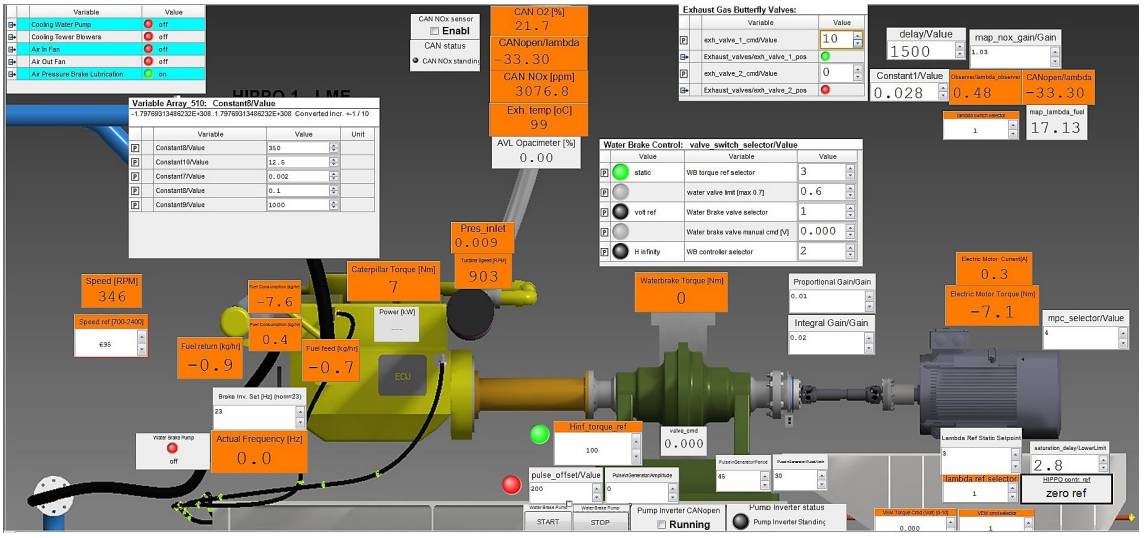

Figure 2.6: The HIPPO-1 monitoring and control screen.

Chapter 3

Modeling

3.1 Introduction

The main aim of the modeling procedure is to construct a mathematical description of the hybrid power plant behavior in such a way that the extracted neural network models can produce accurate results and be trained relatively fast for the smallest possible configurations.

For this purpose, the models must be designed based on the engineering principles of other semi-empirical models. This chapter includes the modeling principles behind diesel engine's physical phenomena, such as the NO_x formation during combustion and the relationship of the mixture quality for the emissions via the λ value. These principles are essentials for the neural network model inputs selection and the models accuracy.

3.2 NO_x Emissions

Nitro-oxides emissions are produced from the reaction of nitrogen and oxygen gases during the combustion at each cycle and it is common knowledge that they are extremely harmful to global population's health. In ICE applications, the nitrogen-oxides emissions appear typically in the exhaust gases of diesel engines but they can also be traced in SI engines run on petrol and other fuels [19]. Almost all vessels use a diesel engine configuration for their propulsion and that is why the marine industry have been heavily targeted with regulations for the emissions reduction from the International Marine Organization (IMO). In the last two decades the emissions reduction targets have became more and more strigent. Unless the manufacturers and ship owners would like to be restricted from doing business in the Emissions Control Areas (ECA) such as US and EU ports, they are racing to comply with the latest regulations as shown in Fig. 3.1 until 2021.

The term NO_x includes both mono-nitrogen oxides, NO, and NO_2. The dominant species are the NO, approx 95%, and thus NO_2 is either neglected or considered as part of the NO because the majority of NO_2 species owes its existence to NO. As a result, they can be considered as a single entity in nitrogen-oxide emissions studies. Cumulative NO_x in exhaxt, sum of NO and NO_2, is considered to occur as:

- Themal NO_x: it occurs from the oxidation of the atmospheric nitrogen. The rate of formation is mainly a function of the in-cylinder temperature and local concentration of the reacting species. At the higher temperatures the N_2 and O_2 dissociate into monatomic N and O which then form NO emissions after a series of chemical reactions.

Tier	Ship construction date on or after	Total weighted cycle emission limit (g/kWh) n = engine's rated speed (rpm)		
		n < 130	n = 130 - 1999	n ≥ 2000
I	1 January 2000	17.0	$45 \cdot n^{(-0.2)}$ e.g., 720 rpm – 12.1	9.8
II	1 January 2011	14.4	$44 \cdot n^{(-0.23)}$ e.g., 720 rpm – 9.7	7.7
III	1 January 2016	3.4	$9 \cdot n^{(-0.2)}$ e.g., 720 rpm – 2.4	2.0

Figure 3.1: IMO's NO_x control requirements.

- Prompt NO_x: it occurs when the N_2 of the air is released and reacts with the diesel fuel radicals, i.e. C or CH_x, and forms NH, HCN and others. These species are oxidized and form NO_x through several reactions.
- Fuel NO_x: it exists because of the nitrogen fraction in the diesel fuel. When nitrogen content in the diesel fuel is released as a radical, then it forms N_2 or NO. However the nitrogen in diesel fuel is typically low and therefore the formed NO can be neglected given the final NO_x emissions.

Overall, the contribution of the thermal NO to the cumulative NO_x is substantial when compared to prompt or fuel NO_x emissions. That is why thermal NO is a focal emission species that marine diesel manufactures try to reduce by developing advanced technologies and operating techniques, such as advanced injection systems for three-phase fuel injection to reduce the in-cylinder temperature during an engine's cycle. For this purpose it is important that there are predictive and accurate emissions models to assist the control strategy development and on board diagnostics.

3.2.1 Emissions Modeling

Modeling NO_x formation and combustion is possible by several approaches. There are empirical, physical and semi-empirical models. Empirical models are based on measurements and use coefficients and parameters that can be tuned to use measurement data and produce NO_x results. Tuning implies a calibration of the empirical model to the measurements so that the model results error from the measurements is minimized, ideally, to zero. Physical models are based on physics and do not need measurement as inputs to produce results. Instead they use variables such as geometry, fuel and material properties, flow and other equations to predict results. Semi empirical models predict results under the same principle but measurements are used to calibrate coefficients and serve as initial conditions.

Physical models can be multi-dimensional or even zero-dimensional. For example, computational fluid dynamics models are multi-dimensional models which take into account all 3 space dimension and time. Zero dimensional models do not produce spatial dependent results and all phenomena are described by control volumes independent of the fluid local position.

Regardless of the fidelity of the model, there have been several studies for the combustion and NO_x formation and most of them are based on the Zeldovich mechanism,

which is similar to Heywood's approach. Many of the models depend on in-cylinder quantities which are used as measurements or calculation results from heat release models. Specifically, NO_x formation is a highly non-linear phenomenon that can be described by local in-cylinder temperature and air-fuel species concentration. Given the fact that in-cylinder local quantities cannot be reliably measured, complex physics models such as the extended and super extended Zeldovich mechanisms [1] can reproduce the NO_x formation depending on the combustion parameters, as described in 3.1:

$$NO_x = f(m_{air}, m_{fuel}, \lambda, RPM, P_{inj}, T_{in}, SOI, EGR_{fraction}) \tag{3.1}$$

Most of these quantities are not available by sensors in commercial marine diesel engines due to the lack of physical sensors and maps from the engine manufacturer, e.g. start of injection of maps. Although some of them are available by physical sensors and in different locations, e.g. at the intake manifold or downstream the turbine. For example, physical sensors measure the intake manifold absolute pressure which is directly correlated to the intake mass air flow. As a result in order to model the NO_x formation during operation for the engine control systems, one must rely on both what engine parameters are available and make sense to use based on first principle's theory.

3.2.2 Relative Air-Fuel Ratio: λ value

Marine diesel engines are equipped with multiple closed-loop control systems that take care of the emissions reduction, e.g. EGR or SCR systems. In these cases, the controls objective is to open or close some valve downstream or upstream the engine cylinders or inject water or other fluids to reduce the in-cylinder temperature or the emissions species directly in the exhaust gases. In order for these systems to perform effective actions, most of them rely on the quality of the exhaust gas, among other variables, with regards to the air to fuel ratio 3.2.

$$\mathbb{AF}_{ratio} = \frac{m_{air}}{m_{fuel}} \tag{3.2}$$

The air mass flow rate m_{air} and the mass fuel rate m_{fuel} are easily measured from sensors as by-products of pressure and species measurements. The normal operating range for the CI engines like the marine diesel engines is the between 12 and 70 or 0.014 and 0.056 for the FA_{ratio}, i.e. the inverse of AFR 3.2.

Fuel and air mixtures with more or less than the stoichiometric air requirements is possible to burn and produce the required heat. However, lean or rich mixtures affect the combustion substantially and that is why it is important to monitor the relative stoichiometric ratio of the mixture. In fuel rich combustion, i.e. less than the stoichiometric needed air, there is not sufficient oxygen to oxidize fully the diesel fuel's C and H. With excess air, i.e. fuel lean combustion, the extra air exists in the exhaust gases unchanged. Because the composition of the combustion products is substantially different for lean or rich combustion, the ratio of the actual FA_{ratio} to the stoichiometric ratio is more indicative for mixture composition. The same is true for the inverse stoichiometric ratio or the relative AF_{ratio}. In other words the relative AF_{ratio} is commonly referred as λ value 3.3.

$$\lambda = \frac{A/F_{actual}}{AF_{stoichiometric}} \tag{3.3}$$

λ value equal to 1 indicates a stoichiometric mixtures, less than 1 is leaner mixtures and greater than 1 is for rich mixtures. The relative ratio λ value has been proven very important mainly because of two reasons:

- The behavior of *lambda* dynamics can be relatively easy to be modeled and provides accurate information about the diesel engine performance and/or operating point [28] and
- emission content can be expressed as a function of λ value, as shown in Fig. 3.2 [29]. λ value was proven suitable for indicating nitrogen oxides (NO_x) and Particulate Matter (PM) formation.

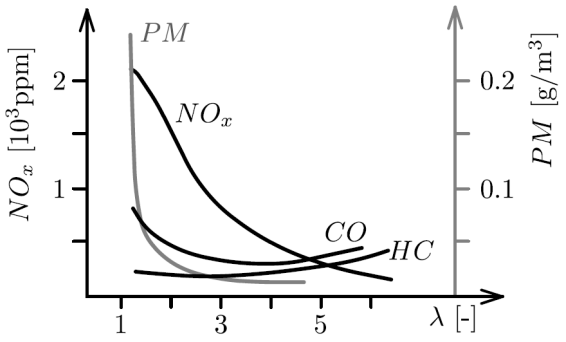

Figure 3.2: Correlation between NO_x and PM emissions to λ value.

As indicated in Fig. 3.2, λ value can be used as an indicator to define several limits for a diesel engine such as smoke, power or temperature downstream the exhaust valve limits, while it can be used to assess the thermal load of a marine engine [15]. That is why λ value is carefully monitored during operation.

3.2.3 Model Inputs selection

Any model is comprised of three parts: the output quantity that it attempts to predict, its calculation mechanism with its coefficients and the input quantities that uses to predict the output. In the beginning of the model design only one of these three is decided almost immediately, i.e. the outputs, as it derives from the models existence purpose. In this thesis, the designed models should predict for the NO_x species and λ value in the exhaust gases downstream the turbine. So the outputs of the models are decided as NO_x species and λ value. The calculation mechanism will be described in section 3.3. In order for the models to be used as assistance of ICE controls system, it is important that the models inputs are decided on rigorous selection based on the theory layed in section 3.2.

For the nitro-oxides model, NO_x formation is a tough problem to solve and the models developed so far take into consideration in-cylinder quantities as shown in . On the other hand, λ value is easier to calculate using semi-empirical or linear models [29]. Neural network model development must take into consideration the existing theory and most common installed sensors in commercial marine engine systems in order to draw variables and build accurate and useful models. In other words, accurate data based model are in need in the emissions controls but they must use input data that are either available from the installed sensor systems or are easily extracted from the Measured quantities via simple calculations or maps. By doing so, the on-board usage of neural network models is promoted and they will not be limited on research level studies for emission control. Taking a few steps back, the installed sensors and calculations provide the essential engine parameters data that are used for the neural network model training. Consequently, the models inputs should be based on the available quantities and should agree with theory simultaneously.

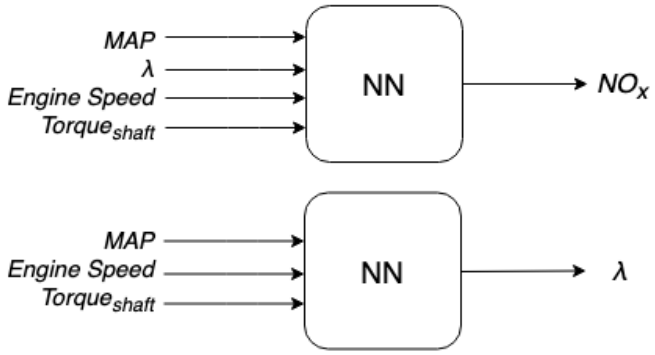

Figure 3.3: Selected Inputs for the NO_x and λ value models.

As stated previously, cumulative NO_x formation is indicated mostly by the thermal NO_x fraction, which is substantially larger than the prompt and fuel NO_x. Its formation depends on chemical kinetics in the in-cylinder local burned gas temperature, the N_2 and the O_2 concentration. Therefore it is a highly non-linear phenomenon that can be affected by local in-cylinder conditions and air-fuel species concentration. Physics models such as the extended and super extended Zeldovich mechanisms can capture the NO_x formation depending on the combustion parameters such as the in-cylinder pressure and temperature. In-cylinder quantities are neither available by sensors in the Laboratory testbed DAQ system nor on a commercial marine diesel engine. Therefore, a minimalistic approach has to be followed with regards to the model input signals selection. The most important ones were chosen as representatives of the operating point and the dynamic behavior of the marine engine as dictated by first principle's theory. As a result, NO_x emissions are modeled with the intake manifold pressure and engine shaft speed as representative of the engine's intake air mass flow which is not measured directly. Torque shaft output and measured λ value are used as representative quantities of the injection parameters such as the SOI, injected fuel mass and pressure. The same principles apply to the λ value models, as shown in fig. Fig. 3.3:

This approach is preferred not only because it adheres to the lack of multiple measured quantities from the LME testbed, but also satisfies the portability and re-usability purposes of the models. Therefore, NO_x and λ value model are developed based on four and three respectively fundamental and readily available quantities for commercial marine engines. It has to be noted that two of the input variables, the engine speed and torque output, are fixed value inputs as shown in Fig. 3.4. This means that they change as instant steps and can be thought as operating point selectors. For the values in-between the steps, the resulting models are expected to interpolate their predictions. It has to be noted that the models are expected to perform worse if the engine speed or torque input signals change in different pattern, e.g. as ramp.

Instant changes are also expected to be smoothed out from the models predictions by the other two input signals for NO_x model -the Measured λ value and manifold absolute pressure- and the one other signal for the λ value model -the manifold absolute pressure. These signals have been logged for their full trajectory in previous experiments and thus they retain a dynamic profile as shown in Fig. 3.5, which the models should utilize mostly for their predictions.

As a result, the resulting models should have enough information from their input

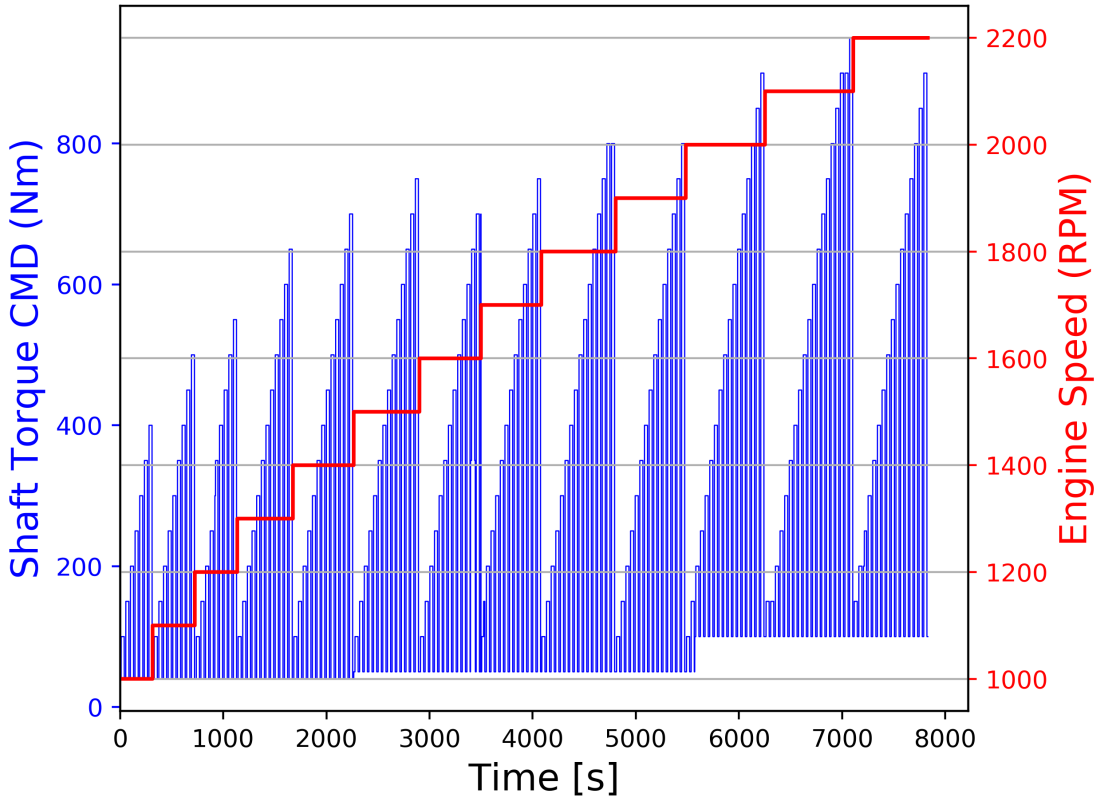

Figure 3.4: Engine speed and Shaft torque static input signals.

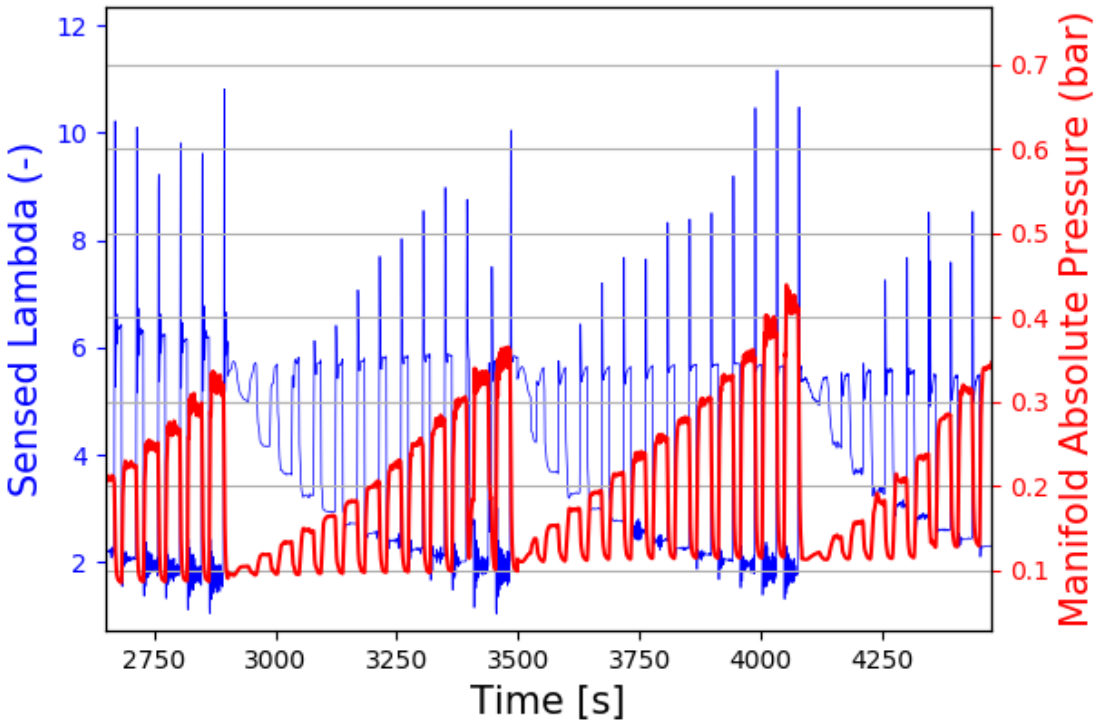

Figure 3.5: Measured λ value and manifold absolute pressure dynamic input signals.

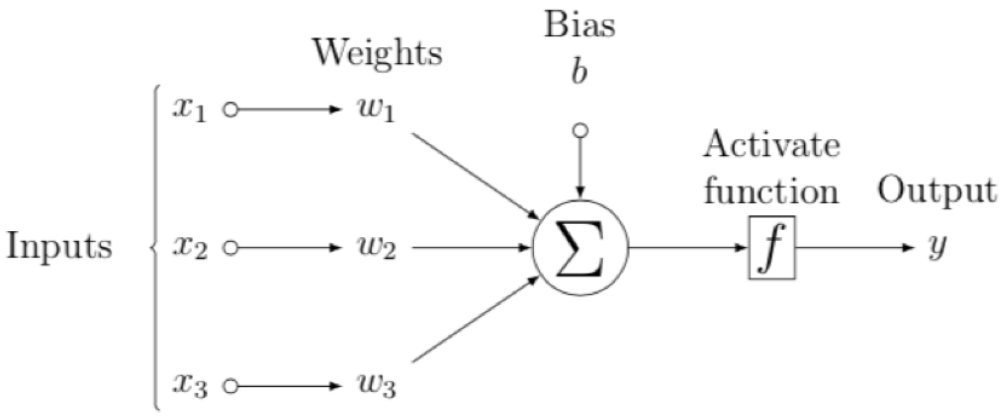

Figure 3.6: Rosenblatt's [20] perceptron: the simplest neural network.

ports to calculate accurately the target quantities. This should be true considering that most of the engine operating parameters, measured or unmeasured, are correlated with these basic four variables. Therefore the neural network error minimization process, i.e. the training, should be able to construct all the input-output relationships.

3.3 Overview of Neural Networks

In this section, all ground level principles are layed out with regards to neural networks models. As mentioned, a model is constituted by three parts: the target output prediction, its calculation mechanism with its parameters and the input quantities that uses to predict the output. The paragraphs below defines the calculation mechanism and the parameters of neural network models. Additionally, it is described the tuning of the model predictions for error minimization and best practices to reduce the model size and training effort.

3.3.1 Artificial Neural Network (ANN)

The perceptron is the simplest neural network form invented in 1957 by Frank Rosenblatt [20] based on the idea of the biological neuron, which, given input stimulus, fires its own signal, the output. That is why perceptrons are typically called neurons and a format of stacked perceptrons compromise a network which is called an artificial neural network (ANN). The graph of single perceptron is shown in Fig. 3.6.

The calculation scheme in Fig. 3.6 expects 3 inputs x_1, x_2 and x_3, each one is multiplied by a weight, then all are summed in a single value which is then summed with a bias value and the result is fed in an function that produces the final perceptron output. The function in the end is called the activation function of the perceptron because its main role is to activate -or not- the specific neuron. In other words, it controls if the neuron spits its result or not. At the same time the activation function gives the neuron its calculation super-power to predict non-linear quantities such as a marine engine's parameters. Frank Rosenblatt's perceptron had a step activation function which fired the output based on the final value the calculation before it as shown in 3.4.

$$y_n = f(\sum x \cdot w + b) \tag{3.4}$$

It is obvious then that the activation function plays the most important role for a perceptron and it is responsible for the final output.

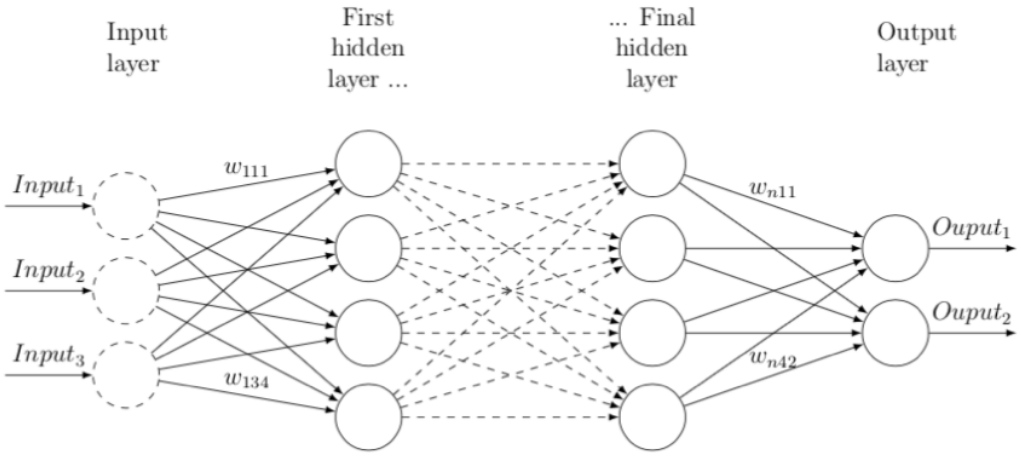

Figure 3.7: Typical feed-forward artificial neural network (ANN).

A single perceptron can be used as a simple linear binary classifier which computes a linear combination of inputs and if the result exceeds a threshold, it outputs the positive or negative class. While this might seem a simple task for a single perceptron, stacking more of them on organized formats, complicated regression problems and classification tasks can be solved, such as fault detection or time-series prediction. A stacked format of perceptron is what is called an Artificial Neural Network or a Multi-Layer Perceptron (MLP) Fig. 3.7.

This structure is more complicated than the single perceptron, compromised by multiple layers, each one of them containing multiple perceptrons or neurons. This architecture is called dense because each neuron is connected to the following layer through vector of weights as shown above. Each circle is a representation of a single perceptron with an activation function which takes the weighted sum of the previous layer's output as input with a bias. The output of the m-th neuron can be defined as shown in 3.5.

$$Out_{m_i} = f(\sum_k out(n_k) \cdot w_{ij} + bias_{m_i}) \tag{3.5}$$

- Out_{m_i} the calculated ouput of m_i
- n_k the neurons of layer i
- m_i the neurons of layer i+1
- f the activation function of the neuron
- w_{ij} the weight corresponding to link between n_i and m_j
- $bias_{m_i}$ the bias value of m_i

To sum up, one has to feed the single perceptron or the stacked neurons with one or more inputs and propagate them through the network equations in order to get the final one or more outputs of the network.

3.3.2 Neural Network Calculation Mechanism

In the previous chapter, a neural network was described as more than one neurons stacked in one or more layers. However, developing "clever" neural network architectures that can solve complicated problems takes more than stacking neurons together. It is about tuning the number of neurons and layers to each application. For example, a neural network model for regression might have one or two hidden layers with five to ten neurons each. On the other side, models for image or speech recognition might occupy more deep and more complicated architectures than the dense network to serve their purpose. Therefore, it is essential that one can decide on the model's hyper parameters:

- the number of neurons
- the number of layers

These two describe the calculation structure of the ANN. While there are general rules on choosing them, there is no rule of thumb or golden rule that always works. These decisions are problem dependent and they are typically assisted by error minimization algorithms. Other important decisions for the design of ANN is the:

- neurons activation function which gives the ANN its calculation power and
- training algorithm and tuning which is responsible for minimizing the ANN's results error given the target value

All these decisions are problem dependent and the principles behind deciding them are layed out in the following paragraphs.

Neurons and Hidden Layers

For many problems, one can begin with a single hidden layer and will get reasonable results [23]. It has actually been shown that an artificial neural network with just one hidden layer can model even the most complex functions provided it has enough neurons. For a long time, these facts convinced researchers that there was no need to investigate any deeper neural networks architectures. Although, most of them overlooked that deep architectures have higher parameter efficiency than shallow ones. This means they can model complex functions using exponentially fewer neurons than shallow nets, making them much faster to train.

Real world data is usually structured in an hierarchical way and deep neural networks can take advantage of this fact; early neurons can adjust to features that are universal and occur repeatedly in the data and later neurons adjust to more niche features that discretize the final outputs [2]. For example, if a trained model with a shallow architecture can adequately recognize faces in pictures, then its layers can be used in more deep architectures which aim to recognize hairstyles as early layers. This way the deep network will not have to learn from scratch all the low-level connections that are characteristic for faces in pictures and it will only have to be trained on the higher-level structures, e.g. face reactions, hairstyles etc. For many problems one can start with just one or two hidden layers and the model should perform as well as 97%-98% accuracy for most function approximation or modeling the MNSIT dataset. For more complex applications, such as non-linear systems approximation or speech recognition, they typically require networks with more layers or more complex architectures than the dense network.

The number of neurons in the input and output layer is arbitrarily defined by the type of input and the number of necessary outputs. This means that the model designer

or the data itself impose the number for the input and output layers. For example, a model for the classification of the alphabet from 64x64 pixels images will have 24 outputs and 64*64=4096 input neurons. Predicting in-cylinder thermodynamic quantities can be achieved by multi-input-single-output neural network models that use inputs from the fixed set of available sensors. Regarding the neurons of hidden layers, a common practice is to size them to form a funnel, with fewer and fewer neurons at each layer, for the reason described in the previous paragraph.

Unfortunately, finding the perfect number of neurons for each hidden layer is described in most references as both arbitrary and result of efficient design [24]. The truth is that there are several ways to choose the architecture of an ANN, e.g. full factorial search for the best architecture by sweep testing all possible architectures or occupying a more clever algorithm to control the ANN design. Nowadays there are frameworks that can be used to easily develop and test quickly numerous prototype neural network architectures and therefore test a lot of permutations, before deciding what architecture is ideal for a specific problem.

Activation functions

The activation function lays in the heart of each node and it is responsible for giving neural networks their power to approximate tough problems such as non-linear dynamic systems. These functions are called activation functions because they can activate the node output based on the input of the node. There are simple activation functions such the rectifier function or the identity and there are more sophisticated non-linear activation functions such as the sigmoid or the Gaussian. Some are performing well for regression problems, e.g the hyperbolic tangent, and some others are performing well for classification problems, e.g. the softmax. This paragraph contains the activation functions that were considered, tested or chosen in the present thesis. The current thesis tackles a complex regression problem with only positive outputs and that is why linear, tanh and ReLU functions were initially considered used.

In automotive applications of ANNs, it has been verified that the linear and ReLU functions can assit an ANN produce accurate results [16]. Although, they can be used in all layers with very good results. They are considerably simple and can be used to pass the biased input to the node output or rectify negative values accordingly as shown in Fig. 3.8.

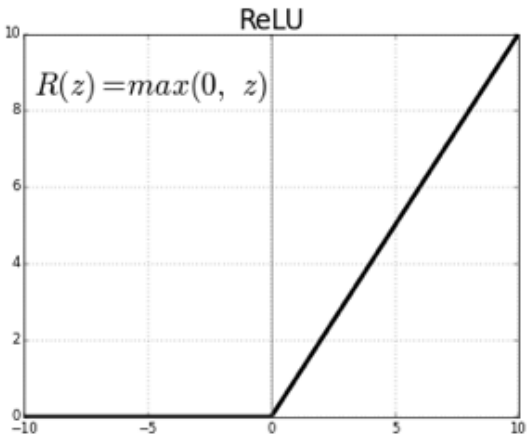

Figure 3.8: ReLU activation function.

ReLU is differentiable at all points except 0, i.e. the left derivative at z=0 is 0 and

the right derivative is 1. This may seem like ReLU is not eligible for use in gradient based error minimization algorithms, such as the back propagation algorithm that is used to train neural networks. In practice this is not true, because training algorithms do not usually arrive at a local minimum of the loss function and ReLU's non-differentiability is acceptable. Additionally, there are a lot of variations of ReLU that can help in various problems, such as linear, leaky ReLU, parametric ReLU.

Nonlinear activation functions are what give neural networks their non-linear approximation capabilities. For example, the sigmoid functions are the most used as they increase monotonically, are symmetrical and are typically saturated to . The hyperbolic tangent outputs span from -1 and 1 and function is defined in 3.6.

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{\exp^x - \exp^{-x}}{\exp^x + \exp^{-x}} = \frac{\exp^{2x} - 1}{\exp^{2x} + 1} \tag{3.6}$$

where x in the case of a neural network node is its input.

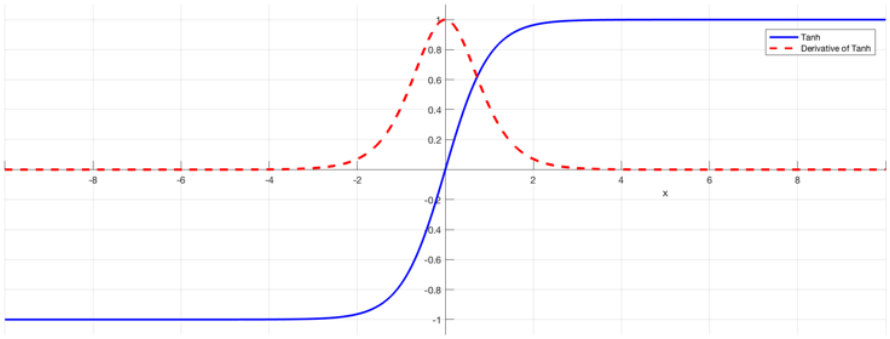

Figure 3.9: tanh activation function.

Symmetric sigmoid functions, as the tanh, are preferred because they are more likely to generate outputs, that are on average close to zero. As it is mentioned later, this helps substantially with the neural network's error minimization. This is in contrast to other activation functions, e.g. the logistic, which produce only positive outputs and can make the neural network training more complex [17].

In general, there are a lot of sources in bibliography that suggest activation functions over others. However, there is no global activation function for all problems as it really depends on what is the neural network asked to approximate, how data are framed and, most importantly, what training process is followed. Training process and efficiency will be discussed in the next paragraphs.

Training ANN: Back-Propagation

There are several approaches to error optimization and machine learning, but most of the successful approaches can be categorized as gradient-based learning methods. The learning machine is simply illustrated in Fig. 3.10.

In the simplest setup the learning problem consists of finding all values of a neural network parameters, the weights (W), that minimize the cost function. In practice, the performance of the system on a training data set is of small significance, because the more relevant measure is the error of the model in the field where it would be used in practice. As a result, it is really important to maximize the model's ability to generalize.

$$E^p = \frac{1}{2}(D^p - M(Z^p, W))^2, \qquad E_{train} = \frac{1}{P}\sum_{p=1} E^p$$

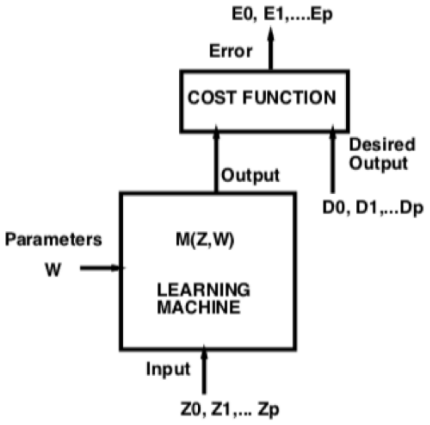

Figure 3.10: Gradient based learning flow.

A widely used method is the back-propagation training. In the simplest format, such as training a neuron, a typical back-propagation algorithm works in three steps:

1. A feed forward pass generates an approximation $\hat{y}$ of the target value, the prediction, using the initial weight and bias values. This pass is illustrated by the the green arrows in Fig. 3.11.

2. Then a reverse pass, i.e. starting from the output backwards, generates local derivatives of each node's output, taking a advantage of the rule chain, and generates the differentials of weights for the error. The back-wards pass is illustrated by the the red arrows in Fig. 3.11.

3. The differentials confess the direction of which the error moves, i.e. increases or decreases, based on what direction the weight values move. Therefore weights are updated accordingly to minimize the error. When the weights are updated then a training iteration or epoch is finished.

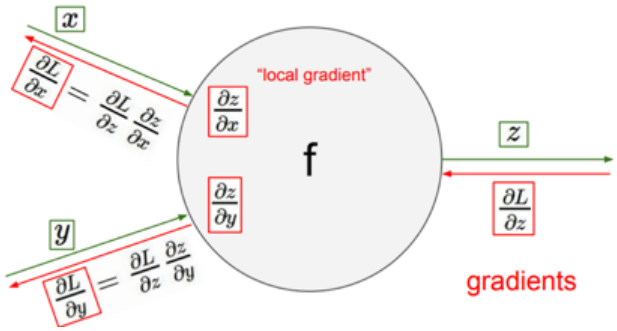

Figure 3.11: Back-propagation zoomed at a single node.

The loop goes on until the final approximation $\hat{y}$ satisfies the expected error threshold between y and $\hat{y}$. After the cost function generates an acceptable error then training loop is finished. The last update for the neural network's weight and bias vector is saved and used for predictions. The last iteration values and the ANN architecture is what is called a trained neural network and what is used during simulation.

Consequently, training a neural network is a standard optimization process and, like every optimization problem, there is no standard settings with regards how to tune it to be successful always. Nevertheless, there are settings that can make the optimization process faster and more effective. These settings are described in the next paragraph.

Efficient Back-Propagation

Back propagation is a very popular neural network training algorithm because it is simple, computationally efficient and often works. However, getting it to work well, and sometimes to work at all, is considered more of an art than science. Designing and training a network using back-propagation requires making many seemingly arbitrary choices such as the number of node and layers, type of node activation functions, algorithm learning rates, deciding training and test set and a lot of other choices. These choices can be fairly critical, yet there is no golden rule that should be always followed, as they are problem and data dependent. This means that no formula could guarantee that a network architecture will converge to a good solution, convergence will be quick and if it will even occur. In this section, a set of best practices will be presented as they were followed later on to make the training process easier for the algorithms used.

Normalizing the inputs is one of the most known practices. Convergence is usually faster if the average of each input variable over the training set is close to zero. Any shift of the average input away from zero will bias the updates of the weights in a particular direction, i.e. increase or decrease, and thus slow down learning. Therefore, it is good to shift the inputs so that the average over the training set is close to zero. Additionally, convergence is faster not only if the inputs are shifted but also if they are scaled so that all have about the same covariance or covariance coefficient.

For marine engine applications, normalizing data is necessary in order to accelerate training because most of the quantities are positive for most of the time and their values differ in orders of magnitude, especially if it is considered that each variable is described by units and units can differ for the same variables, e.g. pressure, temperature, mass flows, pressure ratio and various other quantities. In the current thesis, same quantities use the same unit and all of them are normalized with a minmax scale between 0 or the variable's minimum range value, if it is less than 1, and maximum value equal to 1, in order for the input variables to adjust better to the activation functions used, i.e. tanh.

Activation function affect substantially how training will be accomplished. The most common functions are the sigmoid. As described before tanh is ideal for accelerating training. Especially, Yann LeCun *et al* [25] recommended the sigmoid in 3.7, because it can speed-up training when used with normalized inputs of same covariance.

$$f(x) = 1.7159 \tanh(\frac{2x}{3}) \tag{3.7}$$

The simple tanh is used as the activation function of all neural networks developed under the current thesis. NO_x and λ value are two quantities that follow non-linear trajectories through the operating area of a marine engine, that is why tanh is the most appropriate activation function. The covariance study for the inputs and outputs is neglected given that the inputs have already been decided by the engine sensors configuration and traditional first principle's theory.

Choosing target values is also very important, especially if sigmoid functions are used as activation functions. It is usually best to avoid using target values that are close to the sigmoid's asymptotes and rather be spread to within the "leaner" part of the sigmoid. Care must be taken though to ensure that the target values don't restrict nodes only to the linear part of the sigmoid, or else the non-linear capabilities of sigmoid are not

taken advantage [25]. For example, if a normalized dataset between -1 and 1 is dense at maximum or minimum values, probably a normalization between -0.98 and 0.98 could accelerate training. Another solution might be to use biased versions of the simple tanh.

Some advanced training options can also affect training speed and efficiency. Optimal initialization of neural network weights can have a significant effect on the training process. Weights should be chosen randomly but in such a way that the sigmoid activation functions are primarily activated in their linear region. If weights are all very large then the sigmoid will saturate resulting in small gradients that make learning slow. If weights are very small, then gradients will also be very small. Intermediate weights that range over the sigmoid linear region have the advantage that the gradients are large enough for learning to proceed and the network will learn the linear part of the mapping before the more difficult nonlinear part. This can be achieved by choosing initial weights randomly from a distribution with zero mean and standard deviation of $= m^{(} - 1/2)$, with m the number of connections feeding into the node, assuming first that normalized uncorrelated training variables and tanh function are used. Finally, advanced learning techniques and algorithms might also be occupied to improve training, e.g. using adaptive learning rates algorithms [25].

When the aforementioned techniques are used, they can accelerate and optimize training of a neural network. Training speed-up is essential for the time series approximation problems and even more for marine engine parameters approximation. This is the case because the engine parameters must be sampled with relatively low intervals, e.g. 1-100ms, when compared to other machine learning problems. The low sampling of diesel engine quantities are directly correlated to the cycle duration and the systems time constant and they must not be neglected. However, more target data lead to more calculations during training and therefore training slowdown. In general, the accuracy of traditional ANN models is not guaranteed to converge fast or be optimal during training and that is why more complex model architectures than the multi-layer perceptron have been developed.

3.3.3 Time-Delay Neural Network

A multi-layer perceptron can be a universal function approximator as long as it has enough nodes and layers [23]. This means that even the more complex non-linear systems can be approximated accurately enough with a multi-later perceptron without the need to use very complex cell architectures. In practice, the training effort to train huge stack of perceptron layers is substantial. As a result alterations of the typical multi-layer perceptrons are used to accelerate training and achieve good accuracy with smaller neural network models. One alteration is the multi-layer perceptron that use as inputs their inputs and their previous values and it is known as time delay neural network (TDNN).

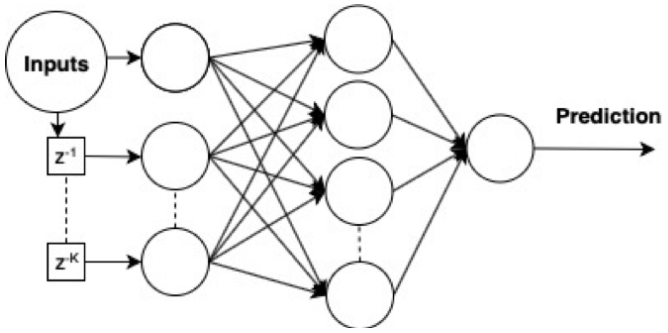

Figure 3.12: Time-delay Neural Network (TDNN).

In principle, these models have been chosen because a dynamic system, such as the

marine engine of the laboratory facility, depends on its initial conditions and its previous states. As a result, a neural network model is more accurate if it takes into account the system's previous states to calculate its next ones. Independent of the other model inputs, using previous input variable states as additional inputs improves the model's accuracy with smaller architectures compared to typical multi-layer perceptron models.

3.3.4 Recurrent Neural Networks

Multi-layer perceptron models have significant mapping capabilities and can perform well prediction unknown data within the training range, even with a reduced set of identification data. However, advanced neural network architectures have been proven to be effective for modelling non-linear dynamic systems by introducing feedback connections a recursive computation structure, i.e. Recurrent Neural Networks. Among others Arsie *et al.* [16] proposed such a model based on predicted output feedback for simulating the nonlinear dynamics of the two phase fuel flow in the intake manifold of an SI engine. Such an application proves that this type of Neural Networks can be used to simulate non-linear dynamic systems. the Recurrent Neural Networks (RNN) meet this requirement because of their dynamic properties. They are derived from a stack of neurons that include feedback connections among themselves. Thus a dynamic effect is introduced into the computational mechanism of the model by a local state. Moreover, by retaining the non-linear capability of the typical MLPs, the RNNs are suitable for non-linear dynamic modelling because of the state recursion functionality. Depending on the feedback topology, state can be retained after each prediction for each neuron or the complete neural network structure. Under this principle, the RNNs are classified as local and external Recurrent Neural Network when the feedback happens locally at each neuron or if only the output prediction is fed back to the input layer respectively [18].

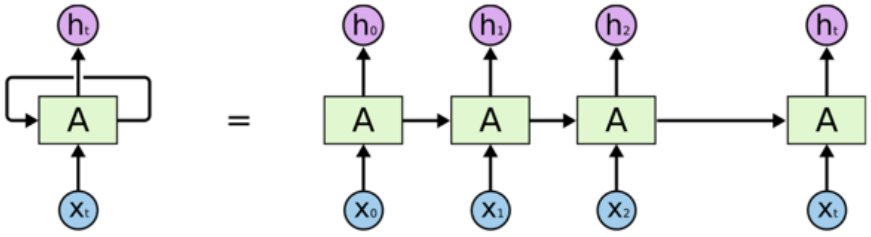

Figure 3.13: Neuron with internal state.

What happens at the first prediction in Fig. 3.13 is that the neuron is fed by an input x_0, produces h_0 that is the neuron output and at the same time is kept as the neuron state for the next prediction. Next, the neuron is fed by an input x_1 and produces h_1 with the contribution of both h_0 and x_1 and so on and so forth. This capability is helpful when the model prediction depends on the internal state of the model, i.e. neural network. In most of the marine engine problems, the state of the system depends substantially on the system's previous state and that is why marine engine variables are traditionally described by differential equations. Taking this into account, any candidate model of a marine engine should be more accurate, if it takes into account the previous state of the system and that is why RNNs with external feedback of the prediction and the previous input values are studied in the current thesis.

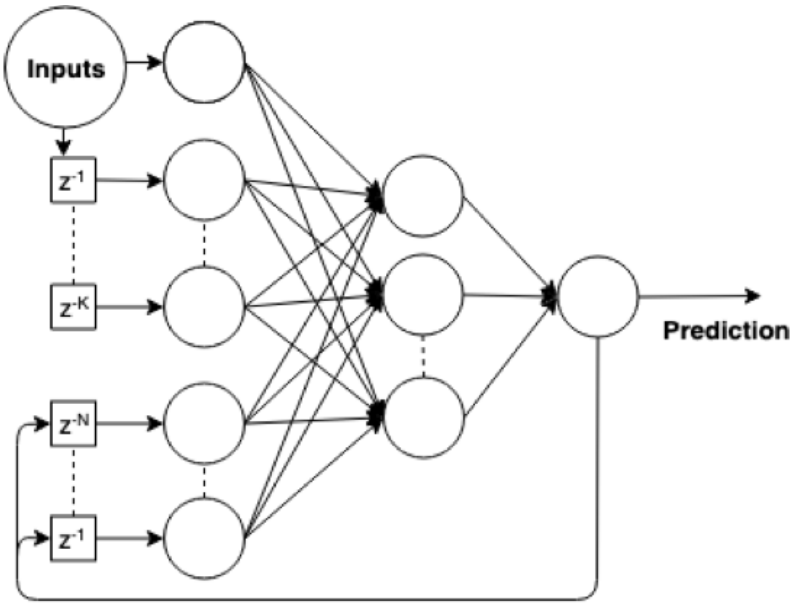

Figure 3.14: External Recurrent Neural Network with Input delays.

In this approach previous predictions are appended as inputs of the complete model and they take part in the calculations for the next prediction. This way a dynamic effect is introduced to the model. Overall, RNNs are capable of behaving similarly to dynamic systems and their dynamics can be described through a set of equivalent nonlinear ordinary differential equations. Hence the time evolution of an RNN can exhibit a convergence to a fixed point, periodic oscillations with constant or variable frequencies or even a chaotic behavior. The stability problem is significantly important, especially when control strategies depend on the dynamic model, thus an appropriate mathematical processing is required. However, in this project the dynamic models developed are only used for simulations. That is why the generated models are not tested in depth in terms of stability, but are extensively validated based on their generalization capability.

Finally, recurrent neural networks are trained under the same principles as the typical Multi-Layer Perceptron, typically with the Back Propagation algorithms. The standard recurrent node adds one more parameter, i.e. one more weight, to train. There are more sophisticated and complex recurrent node architectures, such as the Long-Short Term Memory cells, but they are not investigated in this project.

Chapter 4

Models Design and Training

In this chapter, the ground is layed for the application side of the thesis. Most of the principles presented at the previous chapters are effective when used with careful planning. The absolute goal for this thesis is to develop models for the LME marine engine hybrid test-bed with regards to the NO_x emissions and λ value. The resulting models will be used either as virtual sensors or reference models for controls applications in future projects of the LME. The models developed are data-based and their quality depends on the quantity and quality of data. For that reason, measurements data and system identification studies have been used from previous projects and thesis of the LME. In this chapter, past measurements are presented, prepared and used for the neural network models development. In order to create the model an automated script is designed and used to decide the models' architecture. The script is designed in a modular way so that it can be re-used for other applications or variations of the same problem with different datasets. Finally, the training results are presented.

4.1 Data preparation

4.1.1 Training Data

The data acquisition and sensors systems presented in 2 were used to acquire the measurement data. Developing a neural networks model depends on the data quality and density because:

1. the models draw their trusted range on the training data range and
2. the predictions depend on the training input signals dynamic

Therefore, the data should be enough to represent the complete operation range of a dynamic system, such as the hybrid marine engine test-bed.The measurement data used contain the intake manifold absolute pressure, λ value, NO_x emissions, engine shaft speed and torque for each operating point combination. Specifically, the dataset used in this project is derived from measurements with the ICE warm and the operating point is determined by the shaft speed request which is increased from 1000 RPM to 2200RPM at 100RPM increments with alternating torque step requests as presented in Fig. 3.4. The full combination of training operating points is shown in Fig. 4.1.

This values are used as static values because the raw measurement of engine speed and shaft torque include a lot of noise mainly from the speed governor of the testbed - which is affected by the water brake- as shown in Fig. 4.2 and Fig. 4.3.

The training datasets come as raw data and snapshots from 135 minutes of continuous measurements, which are then stiched together in the full 8000s figures, e.g. Fig. 4.6.

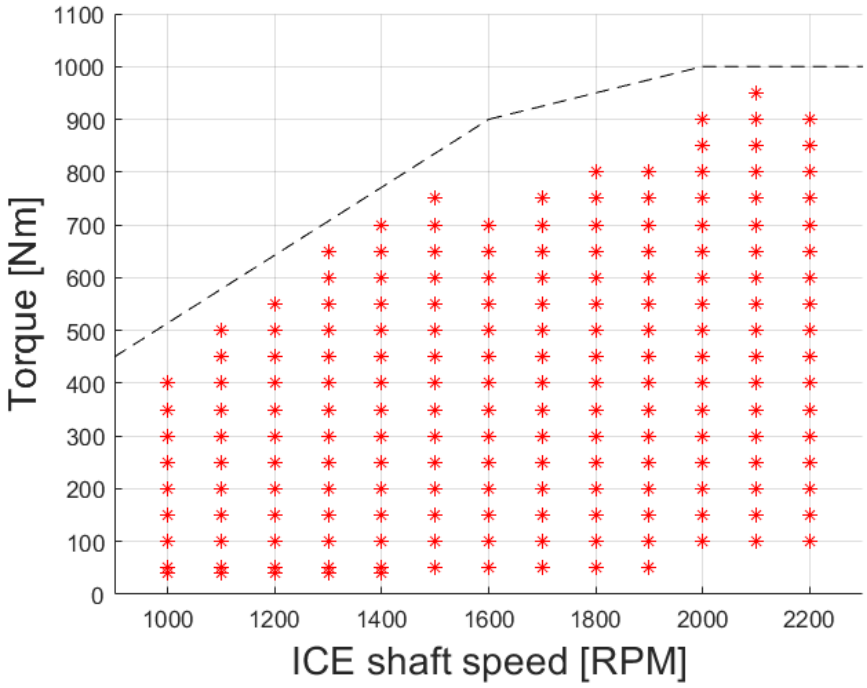

Figure 4.1: ICE shaft speed and Torque load training data coverage within engine envelope.

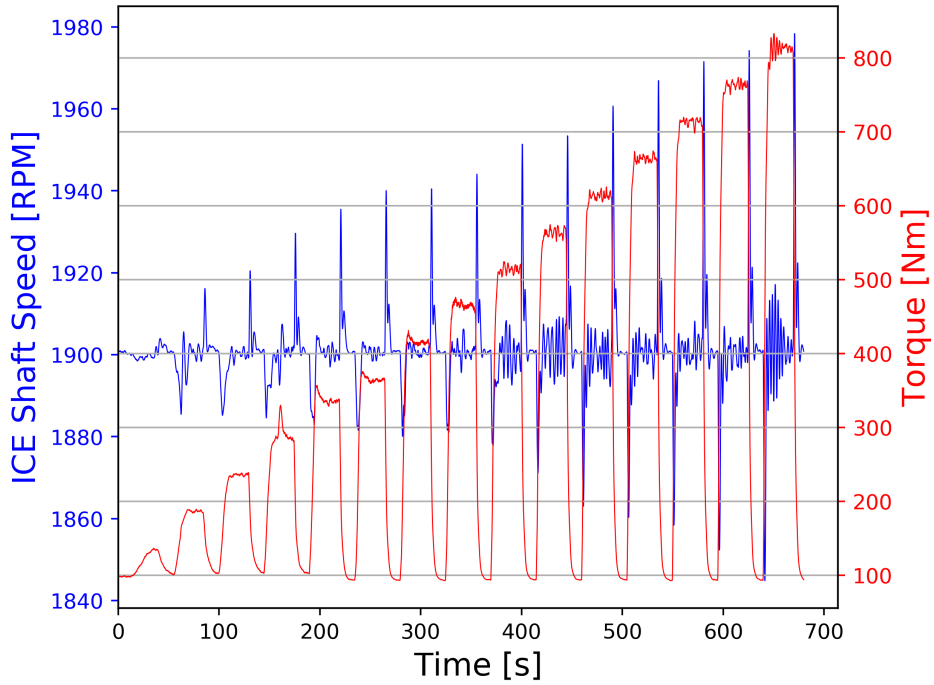

Figure 4.2: Raw ICE shaft speed and alternating step torque load request for 1900 RPM.

Each engine speed target is tested for a time period of 30s to 70s depending on the torque request alternations for the key quantities of the test-bed. For example, figures Fig. 4.2 and Fig. 4.3 are snapshots from static command engine speed tests. After stitching the tests together, the result is the full training trajectories for inputs and outputs of the

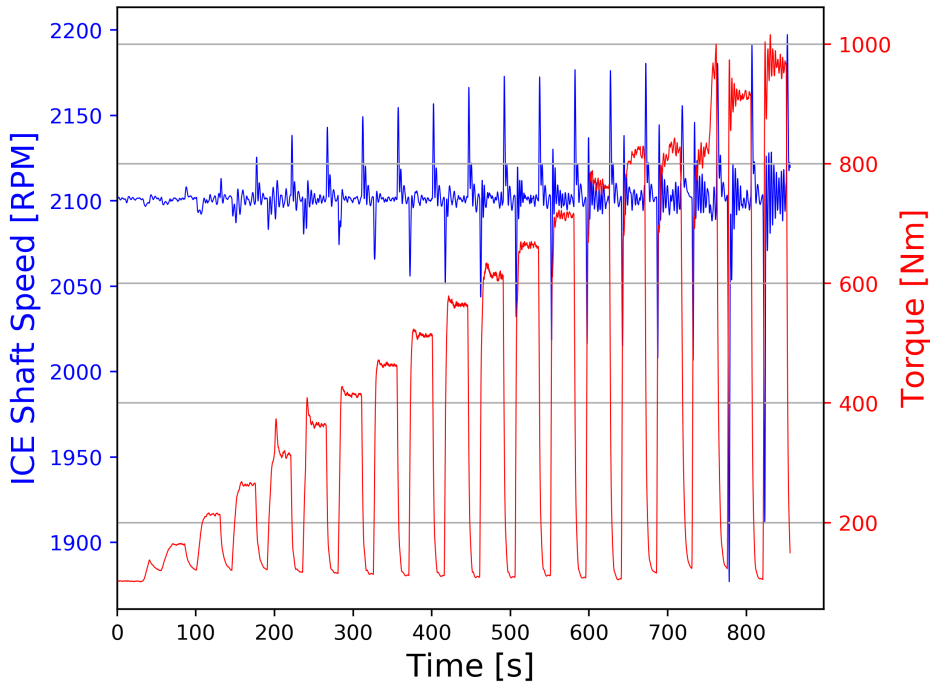

Figure 4.3: Raw ICE shaft speed and alternating step torque load request for 2100 RPM.

models. In Fig. 3.4 only two of the training inputs are illustrated. There are also used the manifold absolute pressure and, only for the NO_x models, the measured λ value as inputs.

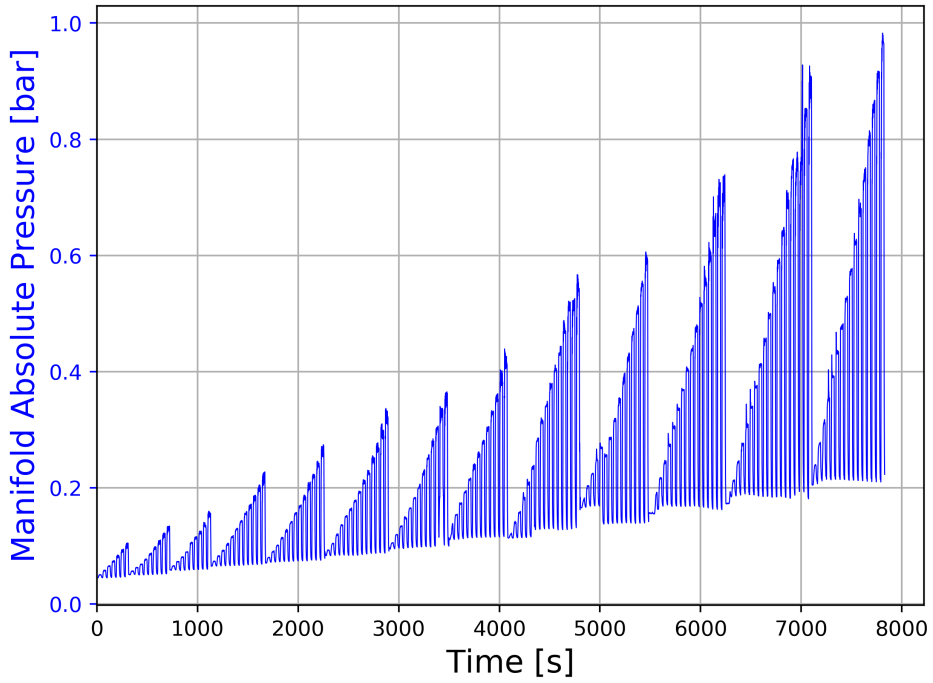

Figure 4.4: Manifold absolute pressure (bar) for the complete training timeseries.

The full training trajectory for each one of these quantities is more than 8000s sampled

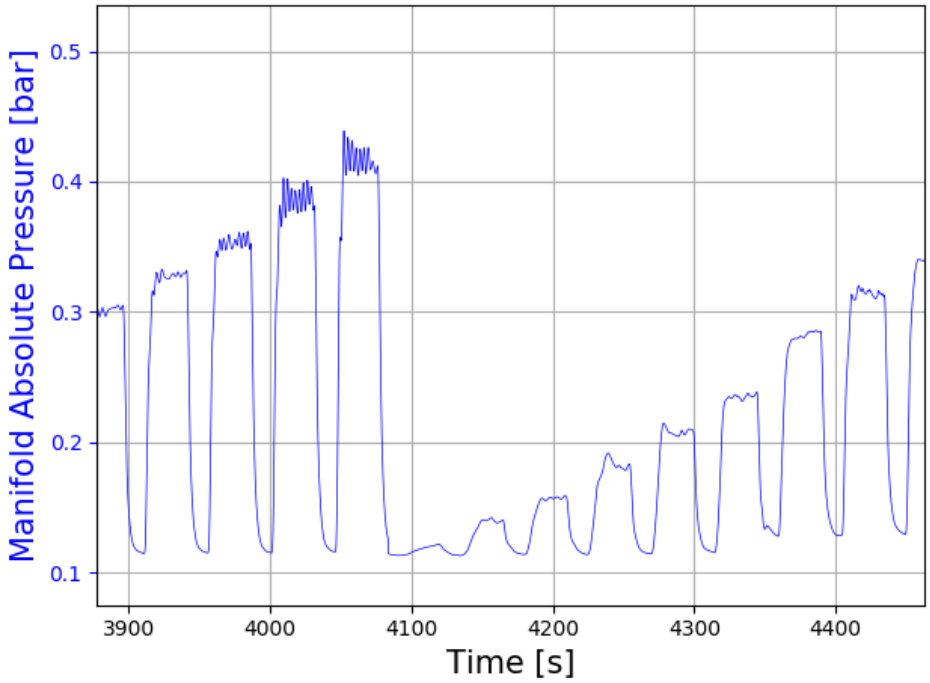

Figure 4.5: Zoomed manifold absolute pressure (bar).

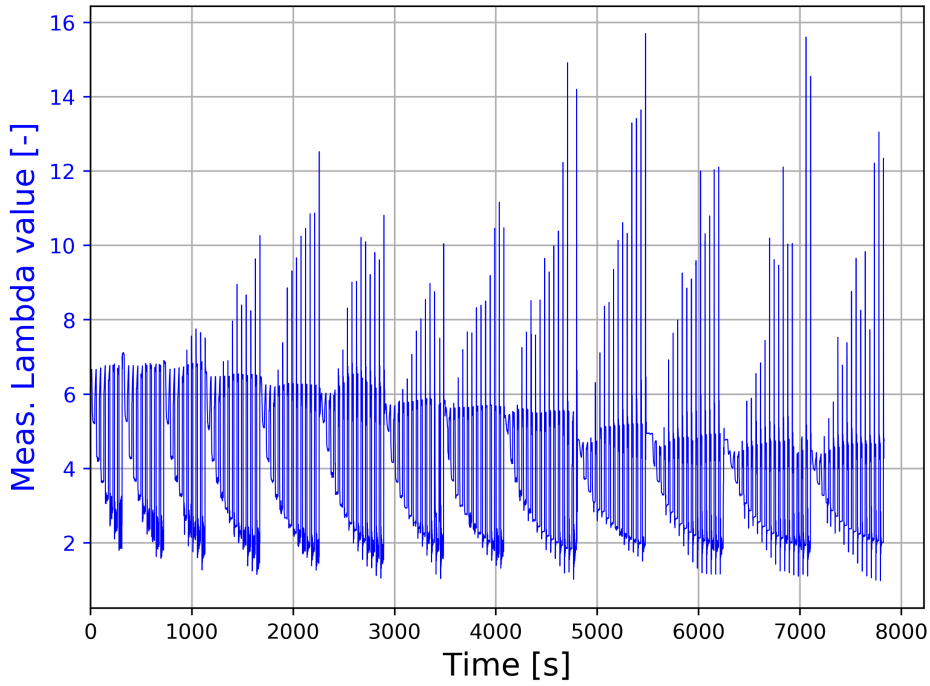

Figure 4.6: Measured λ value from sensors: used both as input to the NO_x models and as target trajectory for the λ value models.

by the default DAQ rate of 1ms, which leads to numerous data points. Even if all the best practices were followed to accelarate the models training, it would still be too long because of the huge matrix inversions the training algorithm has to perform. As a result

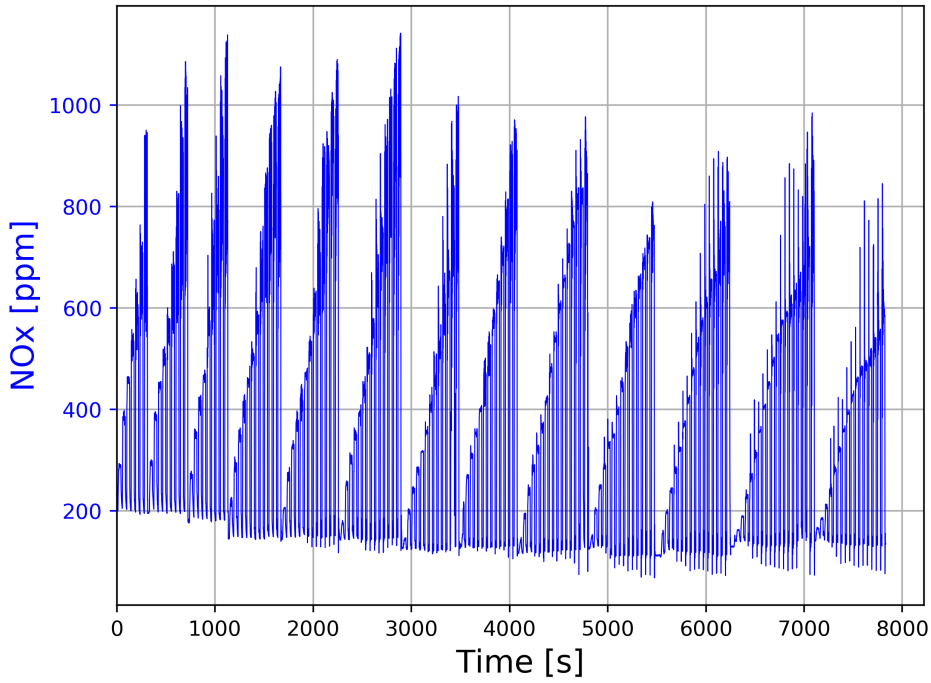

Figure 4.7: Measured NO_x [ppm] from sensors: used both as target trajectory for the NO_x.

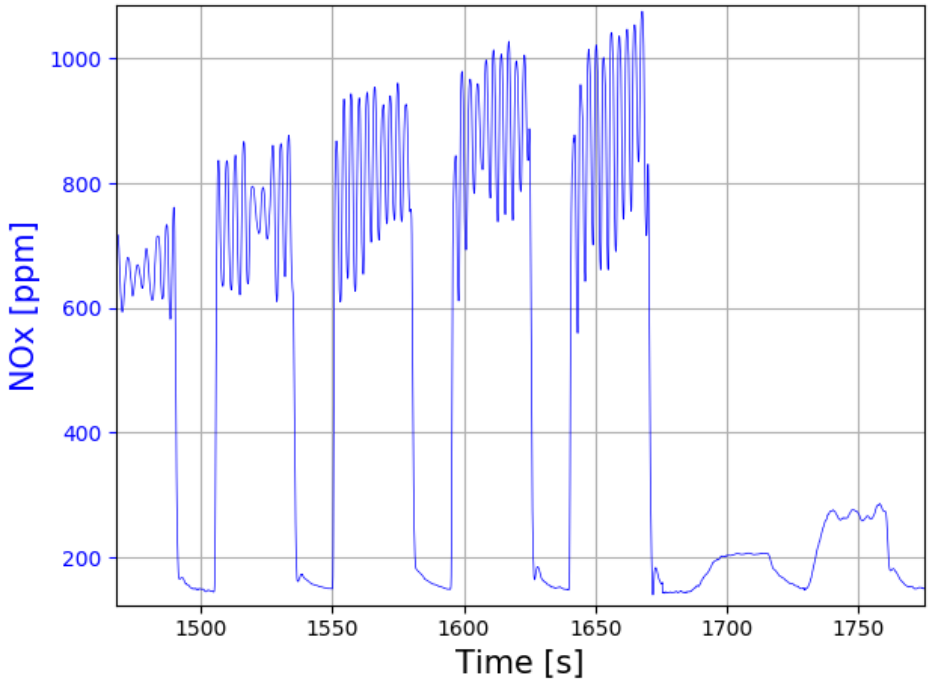

Figure 4.8: Zoomed Measured NO_x [ppm] from sensors: used both as target trajectory for the NO_x.

data must be reduced and cleaned before training.

4.1.2 Data Pre-processing

As mentioned in 3.3.2 , it is very important for the training of a neural network model to pre-process the data before training in order to accelerate it and lead it to a successful error minimization. Both data normalization and cleaning from redundant values that are going to slow-down the training algorithm, e.g. if the very dense datasets that were presented earlier are used directly. Finally the sensors locations introduce a delay between the inputs and the outputs that could affect significantly the correlation of the data and therefore affect negatively the model training. That is why the delay is identified and removed by cross-correlating input torque signal and target NOx and λ value.

First step is to clean the raw measurement data points and reduce them in the order of thousands in order to speed-up training. Although the data reduction should not occur randomly or important details may be lost from the training dataset. It is important that the reduced datasets cover the dynamics of the test bed dynamic behavior. If the datasets are reduced to intervals lower than the time constant of the system, then an aliasing effect is introduced to the measured data sets and the neural network models training will be in wrong data. On the other hand, the training will not converge promptly and the resulting neural network model will overfit, if the model is trained with more than necessary data points. That is why it is important that the system is identified and its time constant is taken into consideration when reducing the training datasets. The identification of the HIPPO-1 testbed is not carried out in this thesis but it is known from previous work in the LME. Papalambrou G. *et al* **??** carried out the system identification of the HIPPO-1 testbed and step response analysis showed that the system's time constant is approximately 0.4s, as a typical diesel engine.

The measurements are carried out with a sampling interval of 1ms which is approximately 2000 times more frequent than the HIPPO-1 system's time constant. This means that the training datasets are over-sampled and it is ok to reduce them. Despite the time constant being in the magnitude of seconds, the data must not be reduced in that scale because a lot of details will be skipped. For example, the models are data based models, but the NO_x formation and λ value quantities vary through the engine cycles which are less than the system's time constant, approx. 50-240ms depending the ICE shaft speed. That is why the data are re-sampled per 100 ms and therefor all information is preserved according to the Nyquist theorem. As shown in Fig. 4.9 , the resulting data do not lose the important details in the transient response area despite the fact that the datasets are thinned by a factor of 100. Additionally to reducing the data, the curves are smoothed from the staircase profile of the Measured trajectories that the data acquisition system introduces due to continuous high rate measurements, which is inconsistent with the value update. In other words, there were too many data points in the initial datasets, even more than the DAQ logged and that is why each value was held for several 1ms instances before it changed. This effect would have been present in the resulting models.

Moreover, it is essential to normalize data depending on the activation function that is going to be used for each layer. The scaled data capture the dynamic behavior of HIPPO-1 testbed and no aliasing effect is introduced. All training input and output data are normalized between 0 and 1 and the resulting data sets mean is close to zero. The scale for each quantity is global for all training datasets and it is equal to the operating range of the test-bed in order for the models to capture the systems' complete behavior. As a result, each quantity is scaled with a single global range, regardless if there are more than one dataset used per quantity and if each training data set spans in a different range. The data normalization happens automatically inside the Python code used for the model development and the resulting models memorize the data min and max range in an internal vector as it will be presented later. During simulation the model uses this

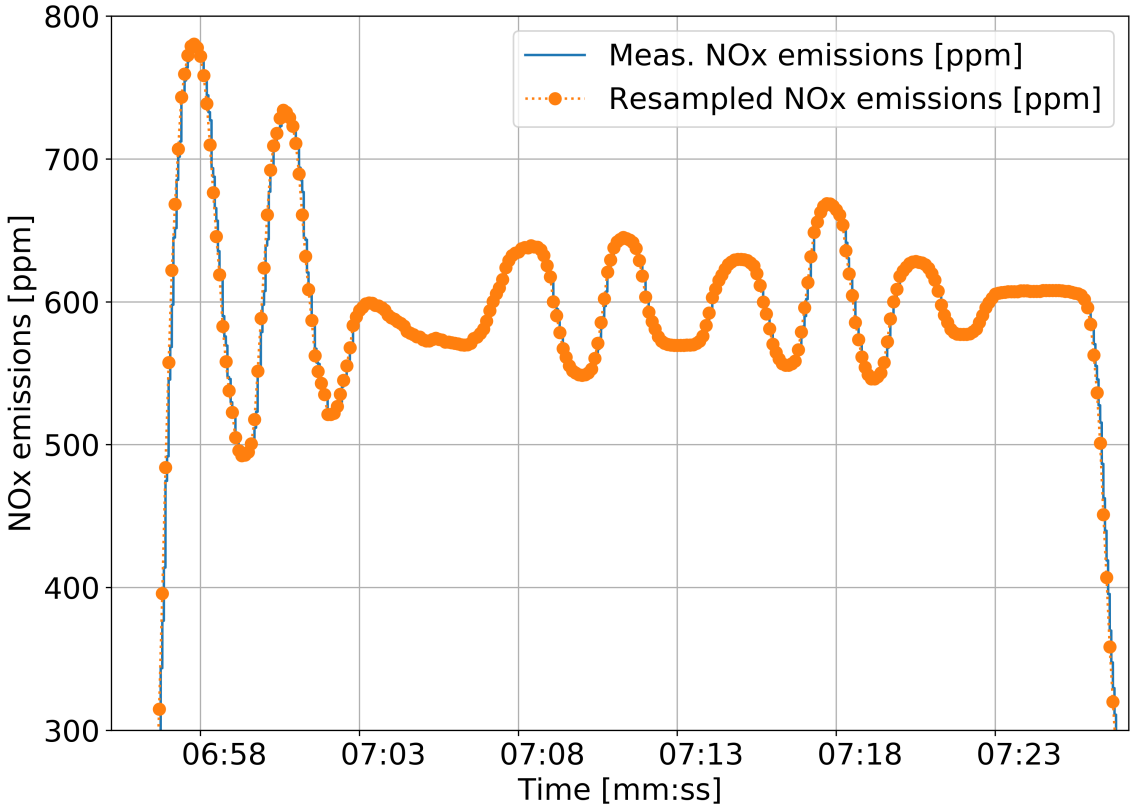

Figure 4.9: Measured NO_x [ppm] from sensors: Reducing oversampled training set.

range to convert the input quantities and produce its output.

Lastly, sensors location at the exhaust duct introduce delay in the measurement data. Laboratory NO_x/O_2 sensor is installed approx. 1m downstream from ICE turbine and the measurements are found to be delayed with regards to other more synchronized quantities. For example the torque and ICE shaft speed commands are signals that are control by the HIPPO-1 operator and occur instantly. After a few seconds of delay and system's time constant, the resulting shaft torque output is reached simultaneously with the NO_x emissions. As a result the NO_x measurements should be synchronized with the resulting torque. This is not the case for the HIPPO-1 testbed and that why the normalized correlation coefficient method is used to identify the delay duration for both of the target signals, NO_x and λ value which are measured in the same location. The correlation coefficient is calculated between the requested torque step command, which agrees timely with the simultaneous change in ICE shaft torque output, and the Measured NO_x. As shown in Fig. Fig. 4.10, the correlation coefficient was calculated to estimate the delay between the input and the output signals.

The shifted data sets reach a correlation of 0.97 on average between the torque pulse and the NO_x measurement. Other inputs data such as the engine speed or λ value cannot be shifted or transformed in a more friendly to correlation value format because they will lose their dynamic. As a result they are left as they are for the NO_x models.

Overall, data preparation was essential because the training data contained measurement noise, delays, redundant data points and they had different value ranges and these could have slowed down the model training.

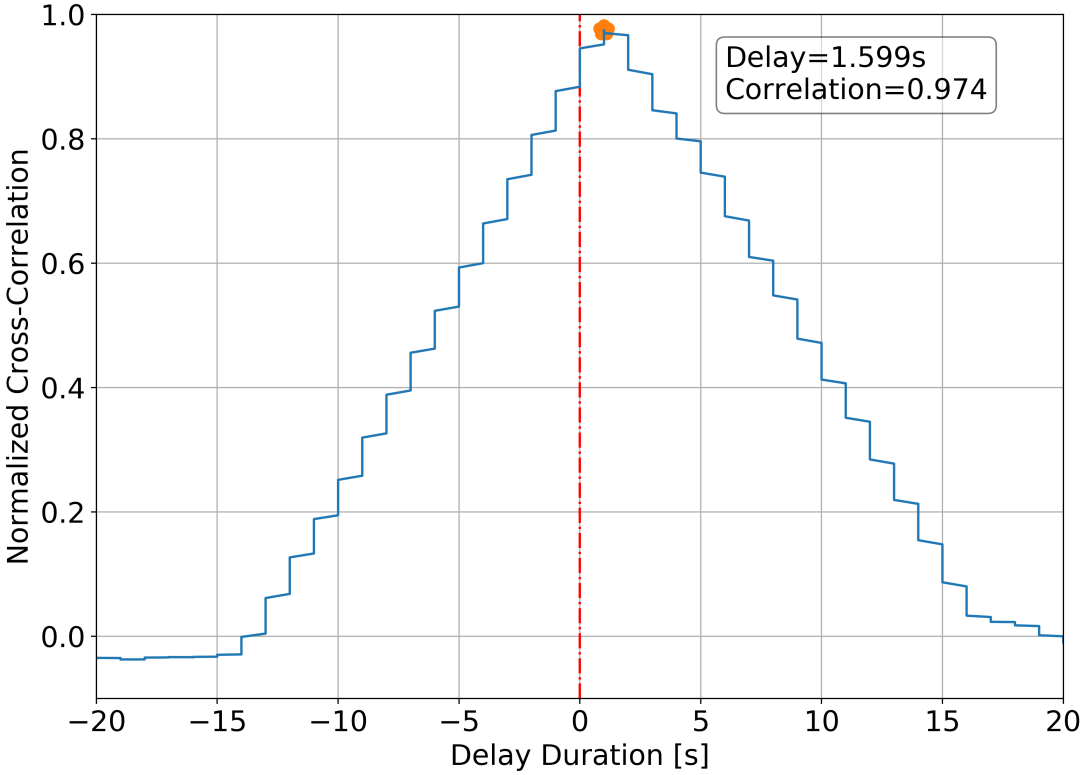

Figure 4.10: Normalized cross-correlation coefficient between NO_x emissions and torque request.

4.2 ANN Design

This section summarizes the process created to generate the neural networks models. From a software development point of view, the problem of developing neural network models is very generic, despite the project's goal. As a result, the process is designed carefully, in a modular way, in order to allow future projects to re-use the Python program developed and tackle any supervised learning regression problem training neural network model. This means that the method described below is not problem or data dependent, as soon as a set of minimum requirements are met:

- Data are timeseries, provided in csv files and each column includes one variable
- All columns have the same length
- Pyrenn module is required for model creation
- Pandas, Numpy, Matplotlib, scikit-learn modules were used

The method designed is using open-source and free Python modules and frameworks. If the aforementioned prerequisites are met, it can be used to generate ANN models with inputs delay and prediction feedback loop for any dataset.

4.2.1 Pyrenn: RNN and TDNN Python toolbox

Pyrenn is a recurrent neural network toolbox for Python and MATLAB developed by Dennis Atabay from Technical University of Munich (TUM) [22]. It is an open source

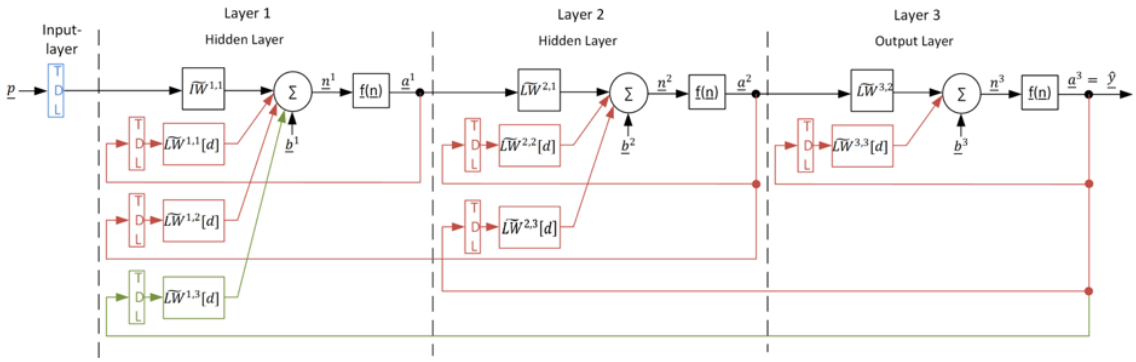

Figure 4.11: Neural Network Architecture guideline from [22].

neural network builder which enables fast prototyping of wide variety of Recurrent Neural Networks using a set of 5 commands. It differs from popular mainstream neural network development frameworks, because it is very simplistic using purely NumPy and Pandas to create, train and use neural networks. Other mainstream neural network builders have been studied, such Keras and Tensorflow, but they have been found cumbersome enough to produce RNN models for time-series predictions and at the same time they need a huge effort to transfer to a ECU prototyping board, i.e. compile their source code to C.

Regarding activation functions, tanh and linear functions are the only ready-to-use activation functions and the training algorithm is Levenberg-Marquardt (LM) [26], which converges much faster than gradient descent algorithms for time-series predictions due to its nature. The number of training epochs needed is lower than other applications might need, e.g. image recognition applications. The LM algorithm implementation for the initial Pyrenn toolbox was prone to freeze when it reached very small incremental improvements on the error minimization. That is why the author introduced a new input variable for the algorithm, i.e. the minimum error minimization step, which forces the training to stop if very small improvements are met, it has been submitted in GitHub stream and it is part of the main Pyrenn stream shown in Appendix A.

After model training, trained models are saved in Python dictionaries which can be used to make predictions by the Pyrenn prediction code. Models that are trained in Python can also be transferred to MATLAB through the Pyrenn MATLAB interface and vice versa. Fig. 4.11 illustrates how a typical RNN structure looks like for Pyrenn.

Overall, Pyrenn was selected for its portability and rapid prototying RNN capabilities. It takes small effort to build neural network models and this promotes quick testing of a lot of model architecture variations. At the same, models can be trained in Python algorithm and easily transferred in MATLAB/Simulink. This enables the portability of the trained models from Python to Simulink and as a last step to compile them for hardware boards, such as the dSPACE ECU prototyping board of the LME. As a result, model creation and portability is accelerated significantly with Python

4.2.2 Model development methodology

In this project, model development takes place by performing a full factorial design of experiments (DOE) looking for the minimum model architecture with the optimal accuracy. The same method is used for both TDNN and RNN models. The difference between the two models is the RNN's additional feature of the predictions feedback loop. This feature is characterized by its duration and that is why a design decision has been made. The additional decision for the RNN is the difference between the design of experiments for

the RNN and the TDNN models.

Specifically, a Python script is developed to carry out the two designs of experiments and decide on the final models architecture. It performs a full parametric study over the neurons, hidden layers, inputs and output delay duration. All models are multiple input and single output. The models are selected as candidate best models based on the R-squared (R^2) scoring. The final models are chosen based on the accuracy and stability of the predictions curve for the best candidate modes, i.e. the models with the higher R^2 results. All architectures are considered because in time-series predictions modelers have to visually select the model if the only metric available is R^2. The algorithm is designed as a generic software that can generated neural network models. That is why it uses a set of inputs, regardless of the problem tackled:

- Training data in csv files with each time-series in each column.
- Maximum number of hidden layers
- Maximum number of nodes
- Maximum inputs delay duration
- Maximum output feedback loop duration

In summary, the algorithm investigates what the feedback loops duration should be for inputs and output delays. Additionally it decides on the optimal number of hidden layers and nodes by creating, training and validating each model architecture one by one. After validation, the R^2 score is saved in a log file. Lastly, the log file helps to choose the best candidate models. These models results are inspected visually, i.e. the results are plotted and evaluated manually, and the final model is chosen based on the noise in the results and the lack of over-fitting on the training set.

The models are created with the Pyrenn module [22]. In the beginning, training data sets are created using the scripts's inputs. The data are inserted in csv files and they are reduced and normalized as mentioned previously. Each model is built within a 3-levels deep for loop that build models depending on the number of hidden layers, nodes per layer, input and output feedback duration. At each pass a new neural network architecture, i.e. TDNN or RNN, is created and it is populated with layers, neurons and activation functions. In this project, several network architectures have been tested and a full sweep of models up to 2 hidden layers, 20 neurons per layer and 12 samples memory of in- puts and output values were studied. Each model is trained for a maximum of 200 epochs with the Levenberg-Marquardt [26] algorithm. After each model is created, it is tested based on the training set for the last time and the R2 score of the predictions is calculated. If the R2 is higher than 0.925 then the model is considered as a best candidate model, the R2, the name and the trained model is saved in log file and in a folder in the working directory. The algorithm continues until all model architectures are created and evaluated. This can be a time-consuming approach and that is why each model is trained for a maximum of 200 epochs and the training stops, if the error minimization is low using the new input variable that was introduced by the author. As a result the training is sped up significantly and how the model architecture size affects the model efficiency to capture the system's behavior.

Lastly, the best model is qualified visually and manually. The results on the training set are plotted and they are inspected for noises, signs of over-fitting and inaccuracies in the most interesting operating areas of the marine engine.

To sum up, the Python script creates and tests all model architectures depending on the maximum number of hidden layers, the maximum number of nodes per layer and the

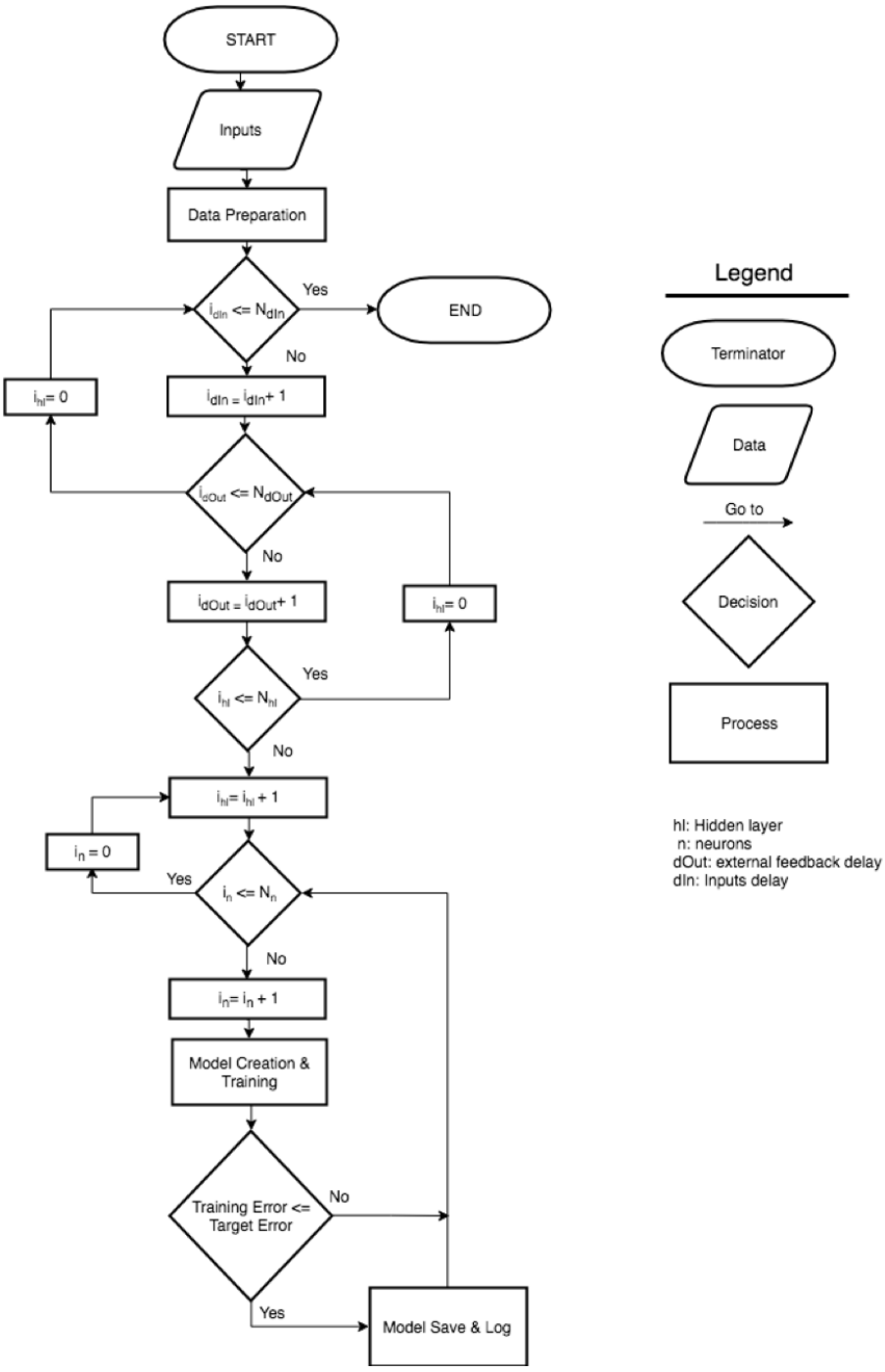

Figure 4.12: Model development flow chart.

output feedback duration as shown in Fig. 4.12. It saves the best models in the csv format, given an acceptable error. The final model selection takes place visually depending on how accurately reproduces the dynamic behavior of the system.

Chapter 5

Model Training

This chapter includes the results and notes from the training process of the TDNN and RNN models. Two models were developed for each ANN architecture, one for NO_x and one for λ value. That is why this chapter is split in two sections, one for the TDNN models and one for the RNN models.

5.1 TDNN models for λ value and NO_x

TDNN models are characterized by their capability of using input sensor data from current or previous instances in order to calculate their predictions. Such models have been developed using the method and the algorithm described in the previous chapter. So a full sweep of TDNN models were tested with up to 2 hidden layers, 20 neurons per layer and 12 steps input delay, i.e. 1200ms, memory of inputs were studied. Each model is trained for a maximum of 200 epochs with the LM algorithm. Multiple architectures have been evaluated for λ value and NO_x approximation and the best models are presented in Fig. 5.1 and Fig. 5.2.

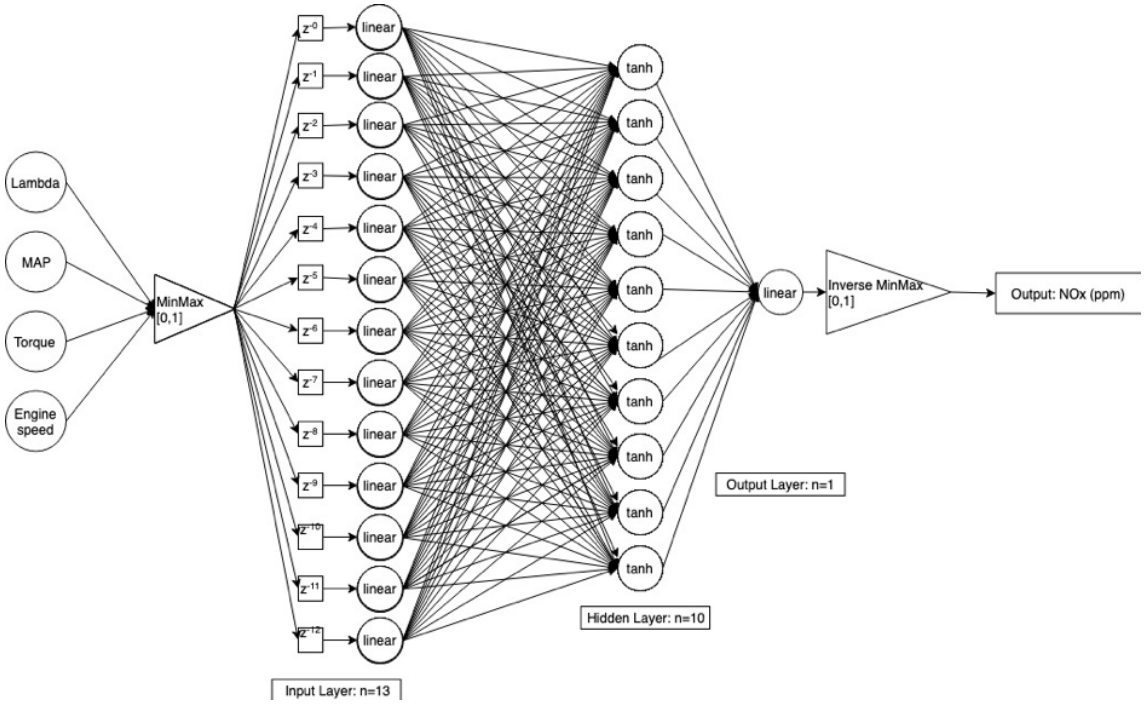

Figure 5.1: The resulting TDNN model for NO_x.

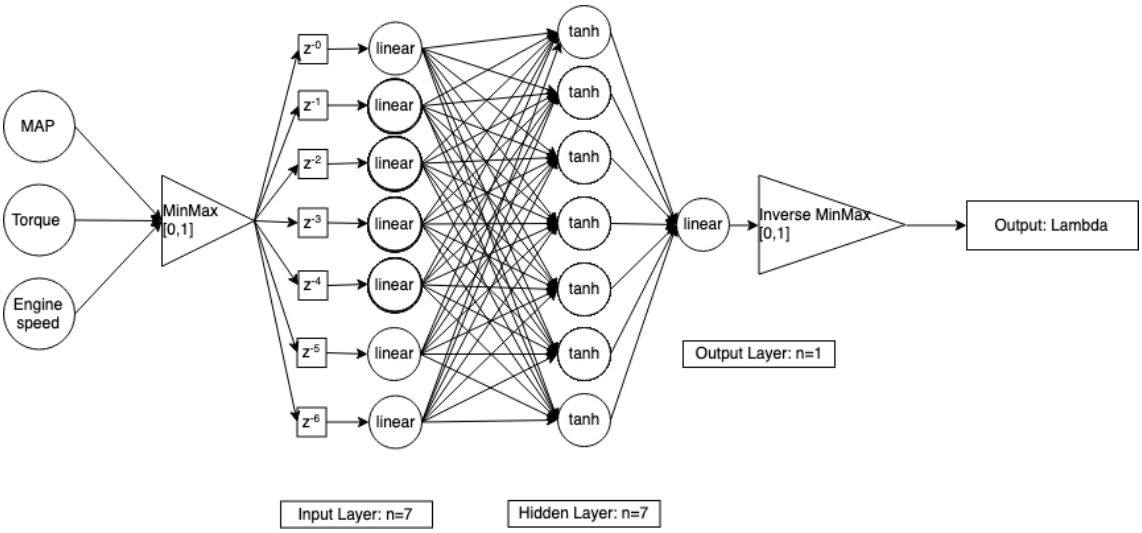

Figure 5.2: The resulting TDNN model for λ value.

The two models are chosen among the best candidate models based on the R^2, as described in the method description section. During training it was noticed that a single hidden layer is enough for the TDNN models to be accurate as they draw their prediction capabilities from the previous states of the inputs. Nevertheless, it was noticed that models with a second hidden layer score high R^2 values but their results are noisy and around the target values, which is a clear indication of overfitting the training set as show in Fig. 5.7. That is why only models with one hidden layer are considered. Additionally models with longer inputs delay duration are more accurate than models with smaller inputs delay duration as shown in Fig. 5.3. However, models with very high R^2 values are overfitting the training dataset. This is observed in the figure which shows that the over-fit model demonstrates high oscillations despite the highest R^2 over all the λ value models equal to 0.9883. A model with lower R_2 scoring, e.g. 0.9498, is producing more steady results with no significant oscillations over the steady state and the transient phenomena. That is why the models are chosen after plotting their results, compared with the target values and the choice of the best model for each application was done manually.

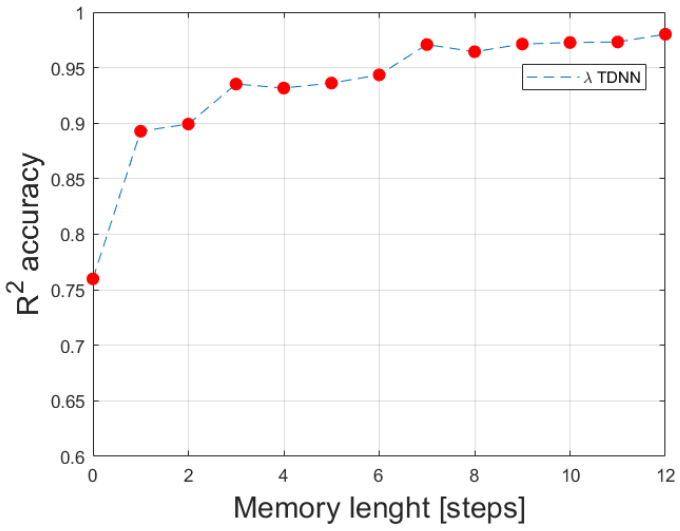

Figure 5.3: Effect of sensors inputs delay on R^2 for λ value TDNN models

For λ value, the TDNN models has 7 neurons and 6 steps input delay. For NO_x, the TDNN model has 10 neurons and 12 steps input delay. The results of both models are plotted on top of the training set. Additionally, the predicted values are compared with the target values in typical predicted versus measured plots.

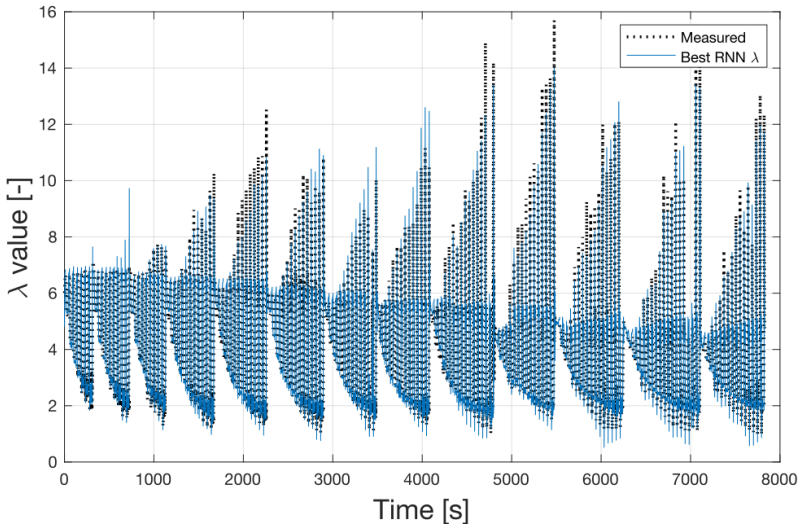

Figure 5.4: Training results for λ value TDNN model.

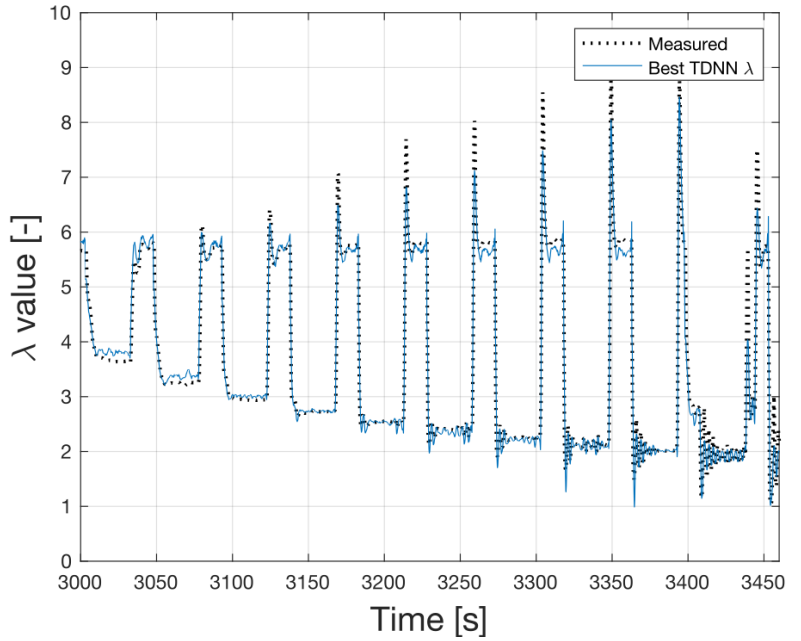

Figure 5.5: Training results for λ value TDNN model zoomed.

As shown in the figures, the best models results exemplify good agreement in the transient behavior of the training dataset, while at the same time they approximate steady state adequately. In some cases there is a steady state error, mainly close to the inputs range, but in most case the steady state results are very close to the target values.

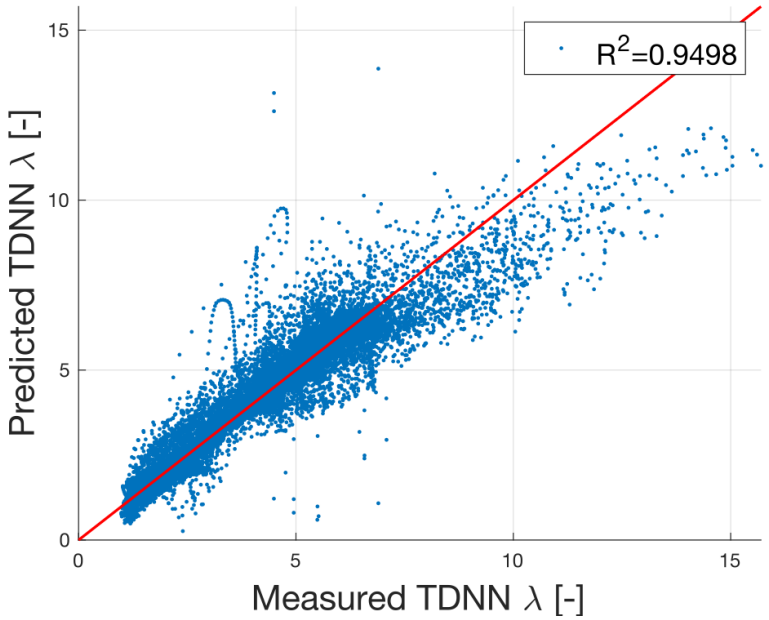

Figure 5.6: Measured versus Predicted plot for TDNN λ value model training.

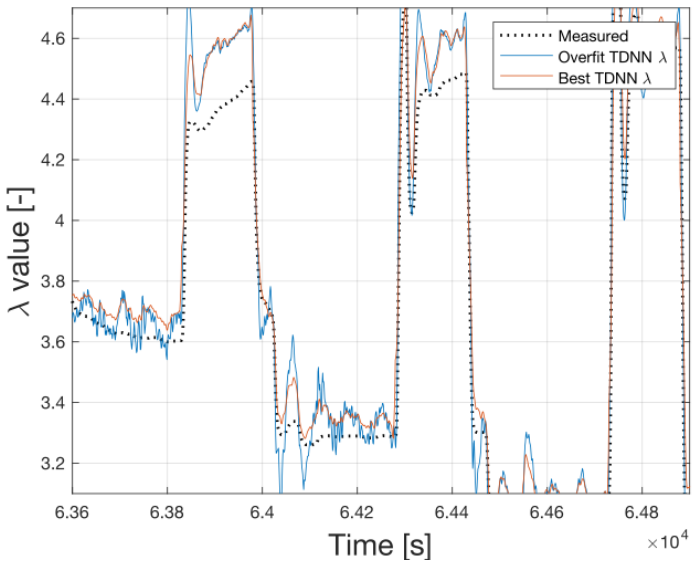

Figure 5.7: Overfit results for λ value TDNN model versus best model and measured values.

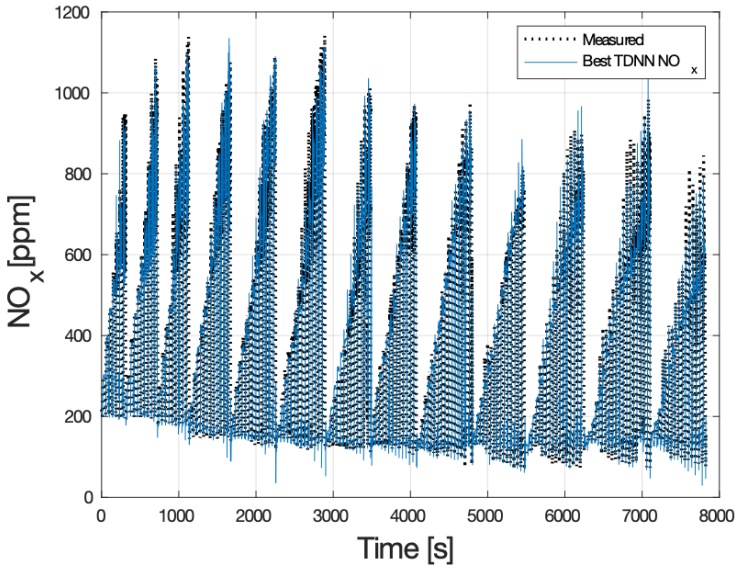

Figure 5.8: Training results for NO_x TDNN model.

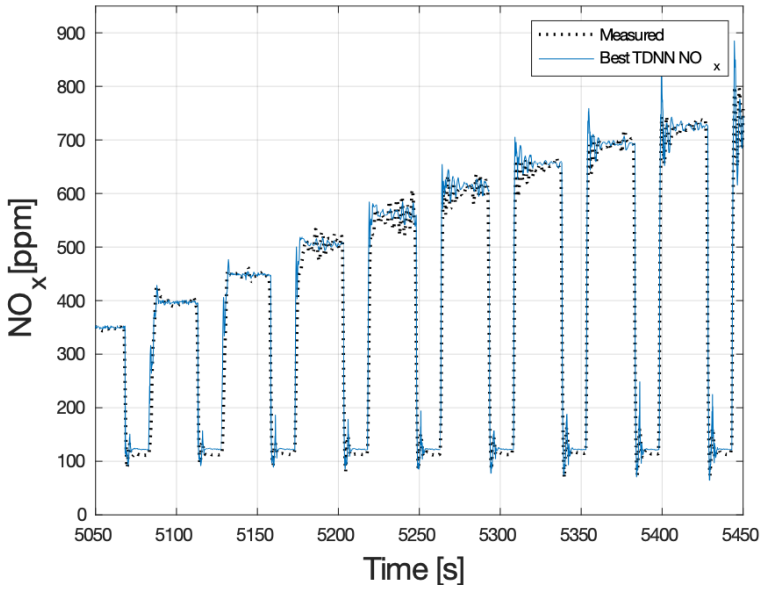

Figure 5.9: Training results for NO_x TDNN model zoomed.

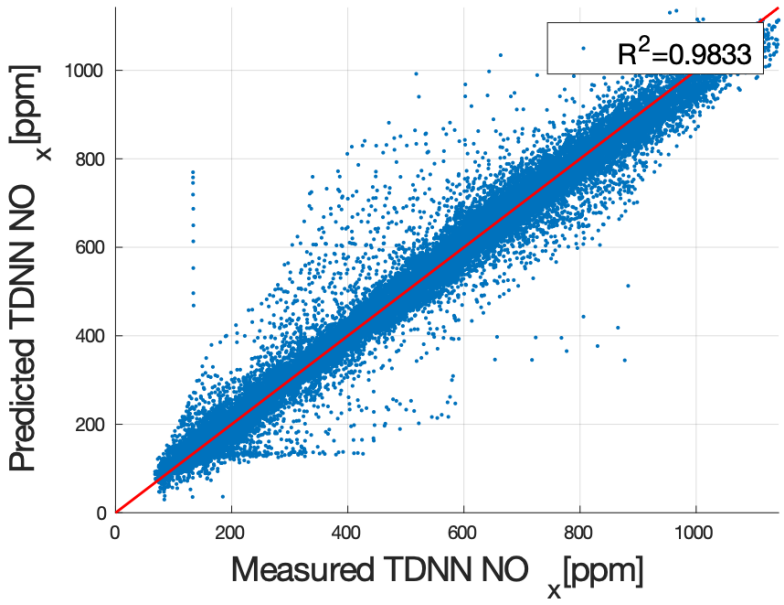

Figure 5.10: Measured versus Predicted plot for TDNN NO_x model training.

5.2 RNN models for λ value and NO_x

In addition to TDNN models, there were developed RNN models that use previous instances of both inputs and output in order to calculate their next output. This helps improve accuracy in the transient area of the training set with smaller model architectures. In other word, the RNN models developed are substantially smaller than the TDNN models for the same target values.

During training of multiple RNN architectures, it was noticed that increasing the output loop duration increased accuracy of the models, similarly to the effect of the inputs delay. Similarly to the TDNN models, it was noticed very early that a second hidden layer was not necessary because the deep architectures overfit the training dataset. It was decided to test architecture with a maximum 12 steps, i.e. up to 1200ms, inputs delay and 12 steps, i.e. up to 1200ms, output delay. So a full sweep of RNN models up to 2 hidden layers, 20 neurons per layer and 12 samples memory of inputs and output values was studied. Each model is trained for a maximum of 200 epochs with the LM algorithm. The candidate RNN models were chosen based on their R^2 score on the training set. If a model scored higher than 0.925 R^2 was saved and its predictions were inspected visually for signs of overfitting, i.e. noise, before the final models were selected. For most of the models the training converged in less than 75 epochs and the LM algorithm stopped for many models by the newly added minimum error stop criterium developed by the author. The two best RNN models for NO_x and λ value are shown in Fig. 5.11 and Fig. 5.12.

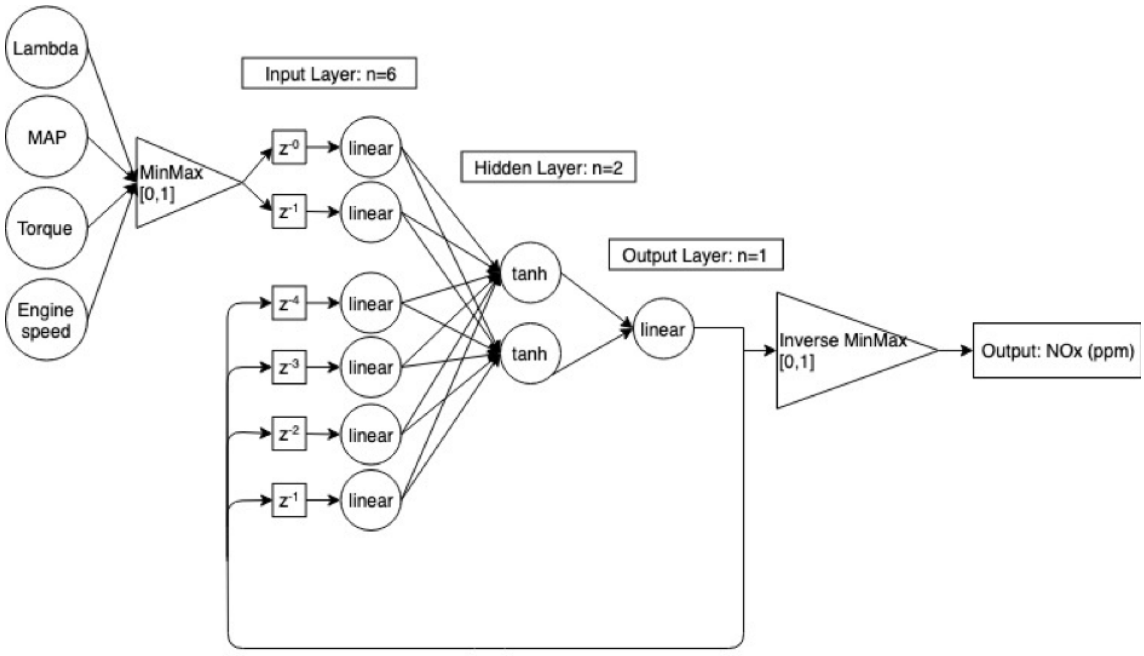

Figure 5.11: The resulting RNN model for NO_x.

Comparing the final TDNN and RNN models, it is noticed immediately that the RNN models are not equally complex but they are as accurate on the training set. This means that the RNN models can predict better results on the training set with less parameters. Specifically, they have smaller input delay duration for the input variables and the hidden layer has less nodes for both the NO_x and λ value models. This was expected because the RNN models are using their previous predictions to calculate the next. It has to be noted that the RNN models must also produce accurate predictions because bad predictions could worsen their next results even more as a chain reaction and the model will go into

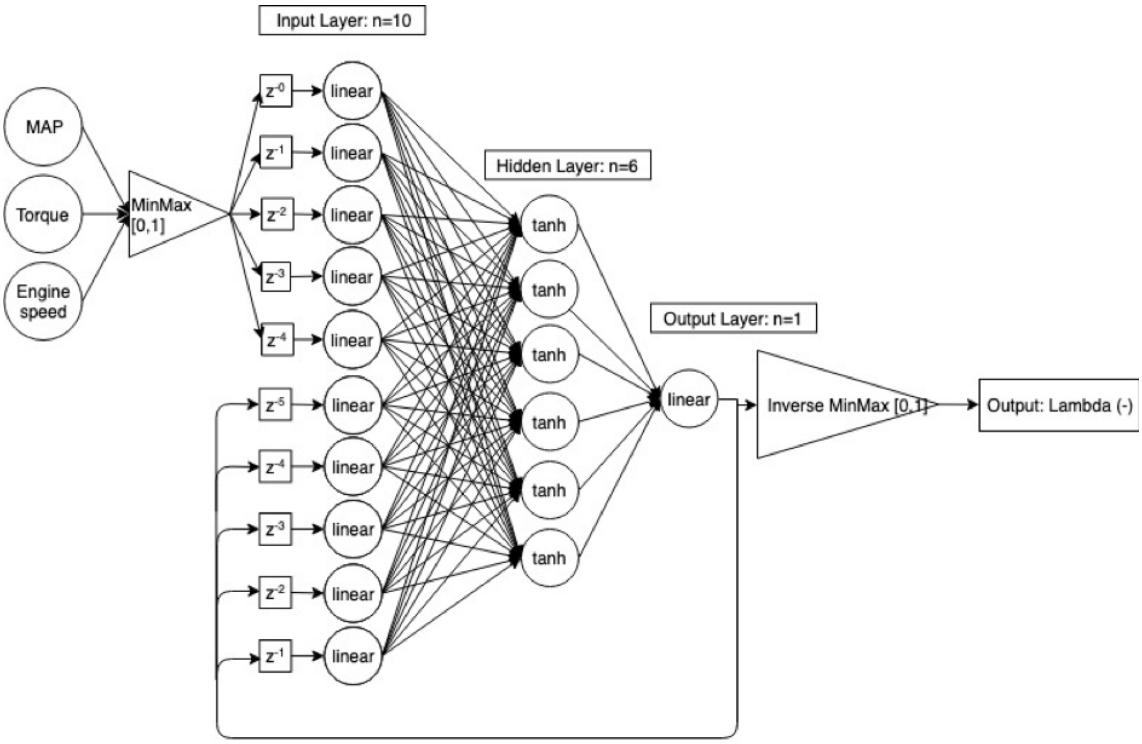

Figure 5.12: The resulting RNN model for λ value.

instabilities. This another reason why the training of RNNs must be as accurate as possible without overfiting at the same time. The RNN training results are show in the Fig. 5.13 and Fig. 5.16.

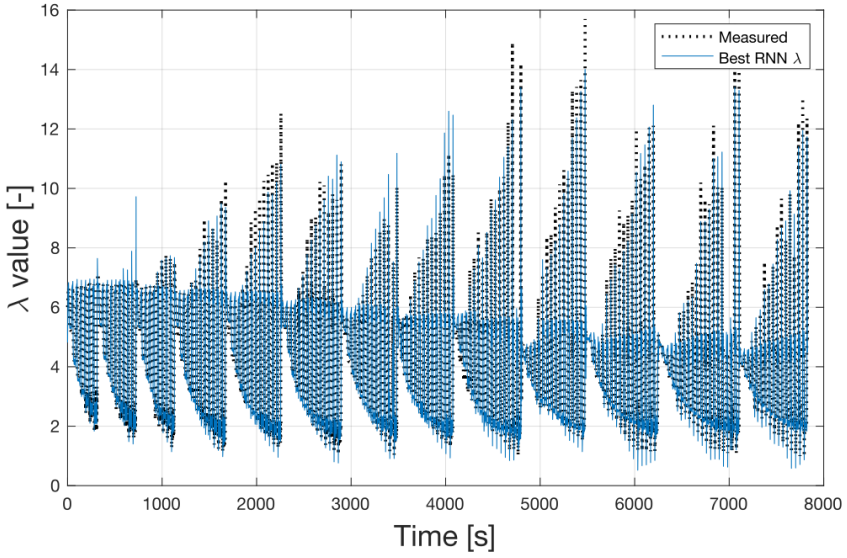

Figure 5.13: Training results for λ value RNN model.

The results for the RNN show better correlation with the target training data than the TDNN models. This is proved both by the R_2 and the predicted-measured plot, which is more compact in the 45^o line. The RNN results are less accurate close to the training set maximum and minimum limits than the results in the middle of the data range, which is

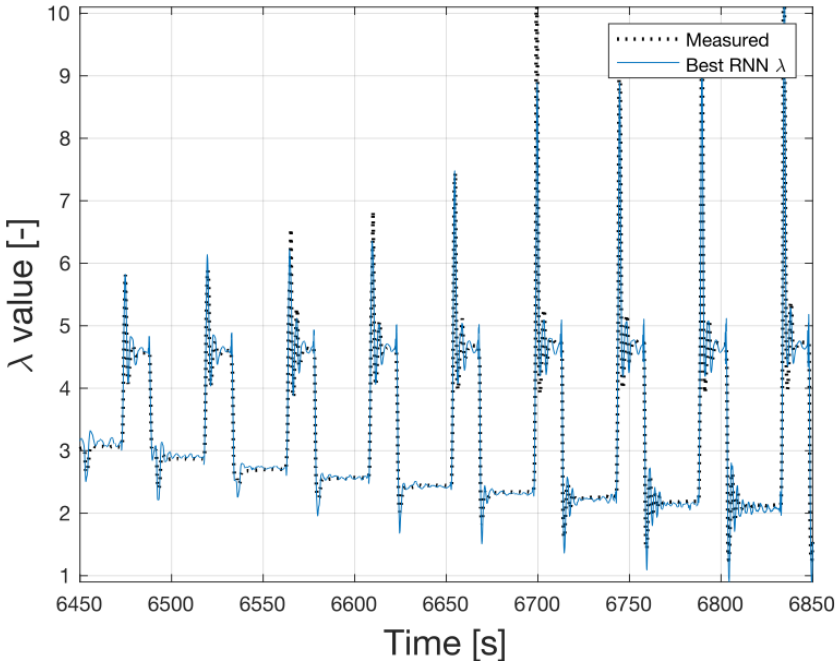

Figure 5.14: Training results for λ value RNN model zoomed.

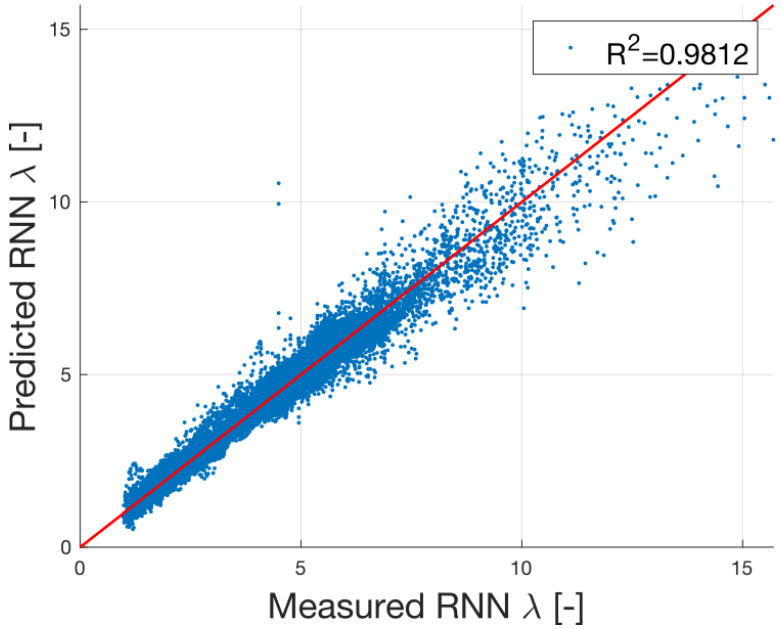

Figure 5.15: Measured versus Predicted plot for RNN λ value model training.

acceptable since in that area are usually non-interesting NO_x spikes. Additionally, this demonstrates the fit of the sigmoid activation function, which produces accurate results at its main lean area but less accurate at the range's limit. Regarding λ value RNN model, the λ value results are less accurate as the λ value values increase, which is acceptable since this represents the high λ value values spikes shown in the figure which is a less important area than the rest of the transient area that *lambda* is equal to 0.9-7.

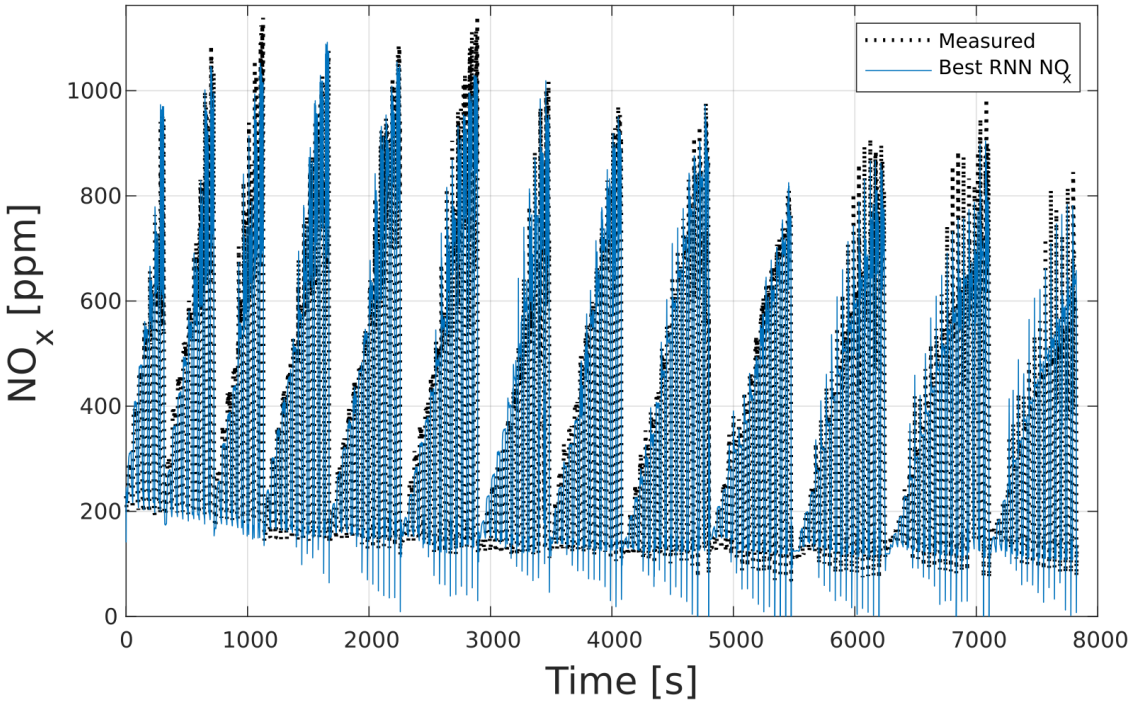

Figure 5.16: Training results for NO_x RNN model.

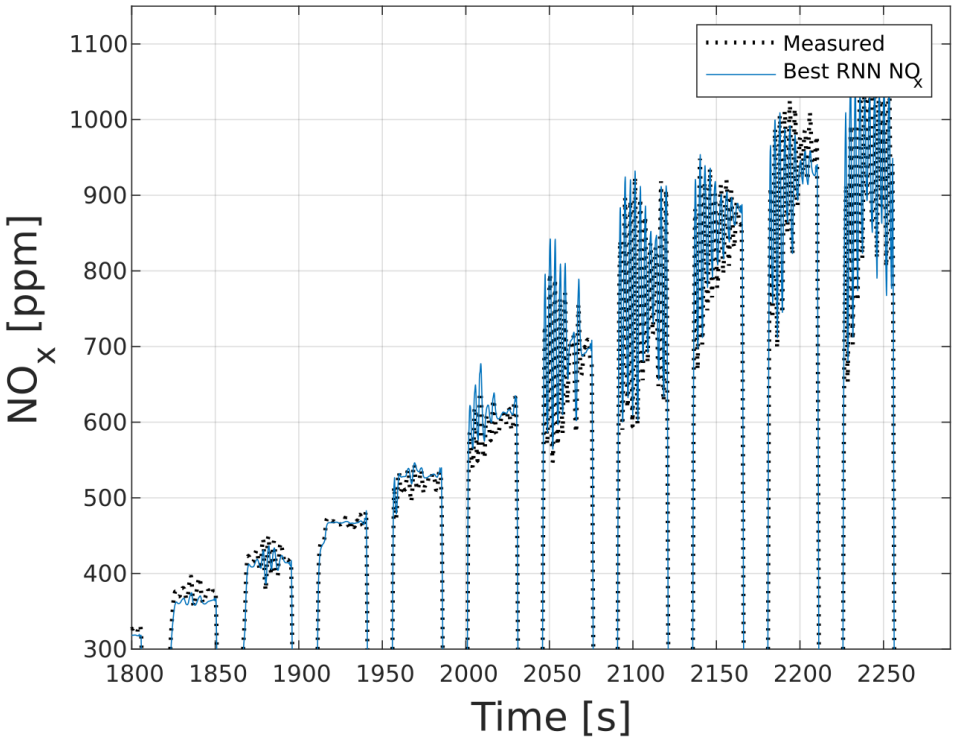

Figure 5.17: Training results for NO_x RNN model zoomed.

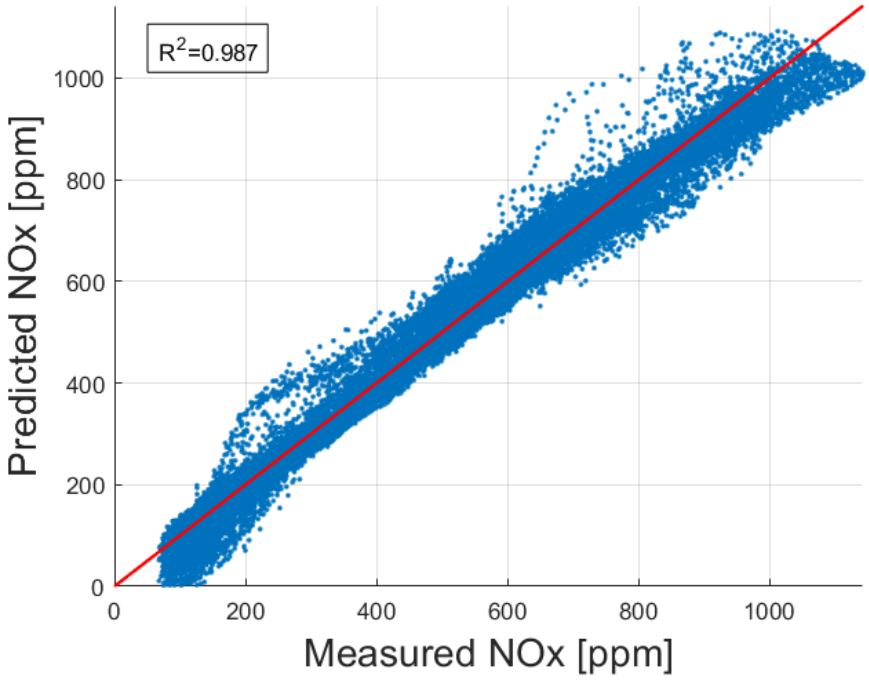

Figure 5.18: Measured versus Predicted plot for RNN NO_x model training.

5.3 Training conclusion

The four best models were chosen among all the different architecture permutations of the TDNN and RNN models. The structure of the final models for NO_x and λ value prediction is presented in table 5.1.

The training is considered successful due to multiple factors, such as the fact that the LM algorithm converged relatevily fast with high R_2 score for each model as shown in table 5.2, the visual inspection of the predicted-measured and the full training-predicted set plot, which shows that the models follow the measured data trajectories successfully in both transient and static areas. At the same time the input selection assumptions are considered equally successfull. In the next chapter, the four selected models are ported from Python to Simulink and then to the LME ECU prototyping system in real-time conditions.

Table 5.1: Internal architecture of the neural network models.

Parameter	NO_x RNN	NO_x TDNN	λ value RNN	λ value DNN
Hidden Layers	1	1	1	1
Nodes of hidden layer	2	10	6	7
Inputs size	8	48	17	18
Inputs Delay	1	12	4	6
Output Feedback Delay	4	-	5	-

Table 5.2: R^2 accuracy score of the NN models during training.

Dataset	NO_x RNN	NO_x TDNN	λ value RNN	λ value TDNN
Training data	0.987	0.9833	0.9812	0.9498

Chapter 6

Real-time Validation with ICE testbed

The trained models have been verified with the training set and they proved that their predictions are accurate enough to consider using them as the representative models of the system. However, the models need to be proved accurate in areas that it is not included in the training dataset but in the same range as the training set. In other words it has to be proved that the models did not overfit the training data and they can both reproduce the dynamic behaviour of the system and interpolate between different ICE engine speeds and torque commands.

That is why the models are ported from Python to MATLAB, then to a Simulink model and lastly code is generated and the models are built and run on an ECU prototype machine. The latter is controlling the test bed system during operation and at the same time it uses the data acquisition system to sense quantities from various locations, e.g. intake manifold or after the turbine. The measured values are used as inputs to the TDNN and RNN models and the model predictions are compared with the quantities from the sensors. This way the models are validated to patterns both similar and unknown to them.

6.1 Experimental Setup

The testbed facility equipment has been thoroughly described in paragraph earlier and that is why it is not going to be described in this one. As mentioned, the laboratory equipment includes a dSPACE DS1103 board which is responsible for data acquisition and control of the test-bed. The same dSPACE board is used to test control strategies harnessing its rapid control prototyping capability and MATLAB/Simulink compatibility. In order to do so, the Python models must be exported.

There were several ways investigated porting the Python models in Simulink, such as compiling the Python code from source, direct coupling between Python and Simulink via handwritten C code s-function. All of them have been proven inefficient due to the high effort needed to tune them up. However, the module used to create and build models, i.e. Pyrenn, can export a trained model as a comma-separated file (.csv). Additionally, it has a MATLAB portion that can interpret the saved CSV file in MATLAB variables for the MATLAB workspace. The loaded MATLAB variables can be indexed accordingly in order to reconstruct the neural network models in Simulink with simple Simulink blocks. This is possible because the neural networks model calculations can be easily reproduced by matrix calculations. The true value of the neural networks is their parameters after a successful training, i.e. model is producing accurate results. The resulting Simulink model are shown in Fig. 6.1.

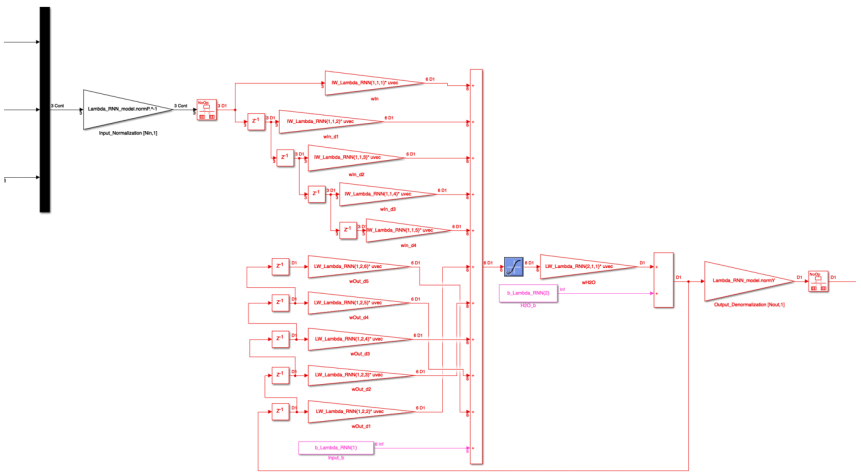

Figure 6.1: λ value RNN in simulink.

As a first step the Simulink model is validated with training inputs from Python and the model predictions are compared with the Python output values. This is necessary because it proves that no mistakes have been done during the porting of the four neural networks. All predicted Simulink results are exactly equal to the predictions from Python tests, as shown for example in Fig. 6.2.

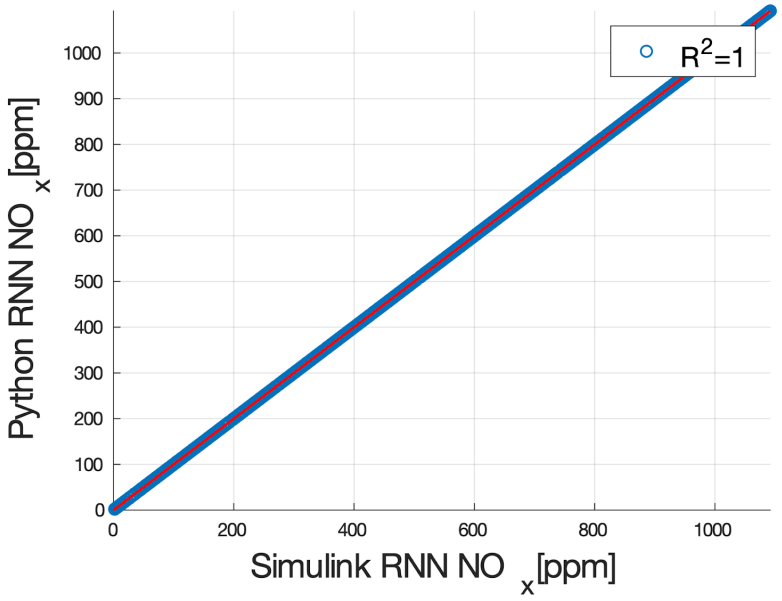

Figure 6.2: Validation of succesul porting NO_x RNN model to Simulink from Python.

While the investigation of porting the models to the board for Real-Time validation was time-consuming, once it was decided and validated, Python models can be easily transferred to Simulink.

Finally, the Simulink model is comprised of simple components that can be compiled via the Simulink Coder for the dSPACE target. As a result the neural network models were deployed for real-time validation tests on the dSPACE board. The tests were carried out in two series. The first is 6 validation sets that replicate the part load conditions that the four models were trained with. This means that both engine speed and torque change

in step command at different instances between each other. The second series of validation is two fully transient routes with step torque commands and constantly changing engine speed. In the first validation case, pulses occur for longer duration than the training pulses, e.g. the pulse height is 200-250 Nm whereas in training the typical width is 50 Nm. For this case, models exemplify that they can reproduce well transient response for NO_x and λ value during a pulse response, regardless of the pulse width. The second validation scenario includes two tests; the first is performed under arbitrarily alternating ICE torque and shaft speed demand between 1200 and 1630 RPM and 300-500 N-m, which is a focal operating point in many test for the ICE tested. The second test is a random case of alternating ICE torque and shaft speed demand, which intended to push the models to their limits. The two last scenarios are challenging because the transient patterns are unknown to the models, close and outside their training limits. Fig. 6.3 shows the validation test area, which is included in the training limits but the validation operating points do not overlap with training ones neither in dynamics or the torque command pulse height and duration. This is enough to test the performance of the NN models within the operating area of the marine ICE.

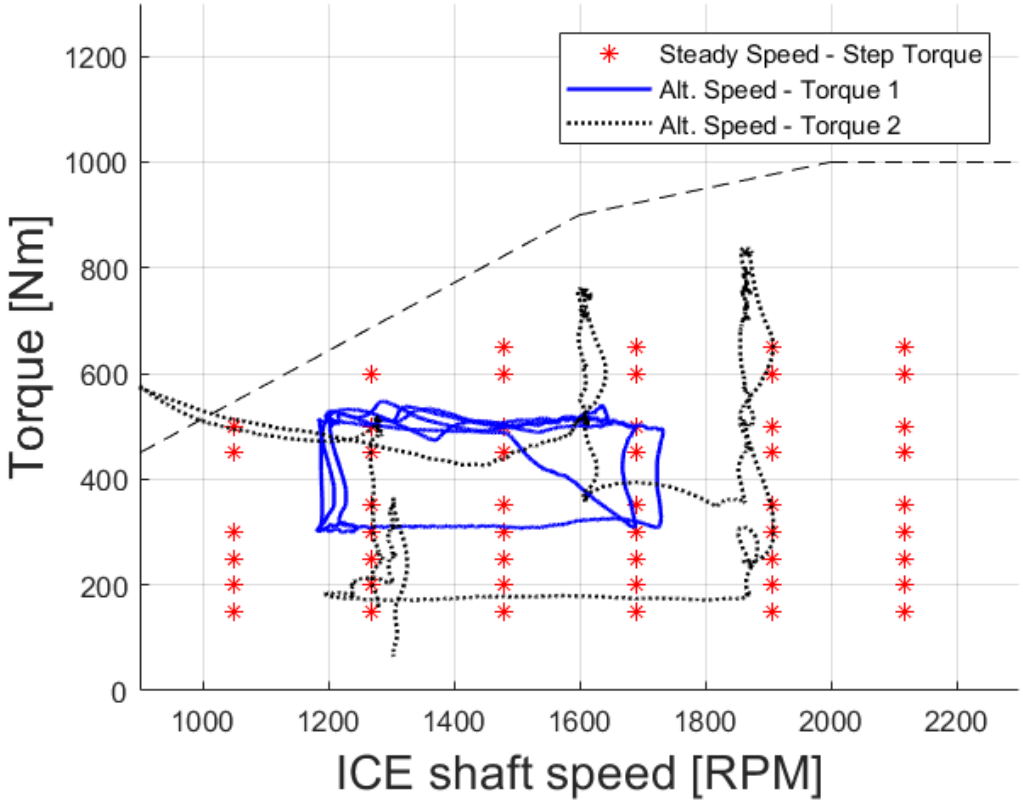

Figure 6.3: Validation area and scenarios for the NO_x and λ value models. The validation is performed within the training area but with different pulse width and more highly transient patterns

6.2 Validation Results

6.2.1 TDNN models for λ value and NO_x

NOx and Lambda TDNN Models The TDNN models follow the part load validation sets adequately but not very accurately. The results for the fully transient dataset are not accurate and they consider unacceptable. The main reason is because the models are trained in part load dataset that change in steps and remain constants until they change again, i.e. the engine speed and the torque command. It appears that once the models are stimulated differently, then their predictions are decrease significantly in accuracy and their R^2 score is very low. This means that the TDNN models not only need be used within the training area, as any data based model, but also they have to use inputs of similar dynamics. It is also important to point out that the TDNN model for λ valueis more accurate than the NO_x model and this is due to the fact that it is simpler and has not overfit the training set. This stems from the fact that the NO_x TDNN model produces very noisy results for areas and input data that it is not familiar with. All these are shown in the table 6.1 and the Fig. 6.4 from to Fig. 6.19.

Table 6.1: R^2 accuracy score of the TDNN models during validation experiments.

Dataset	NO_x TDNN	Cum. NO_x % Error	λ value TDNN
Valid. exp. 1	0.6873	-13.05%	0.7723
Valid. exp. 2	0.766	-12.36%	0.8818
Valid. exp. 3	0.79063	-2.58%	0.91646
Valid. exp. 4	0.835	-0.248%	0.8626
Valid. exp. 5	0.8148	-2.86%	0.9022
Valid. exp. 6	0.81414	1.816%	0.85697
Valid. exp. 7	0.3402	-3.7654%	0.73284
Valid. exp. 8	0.021	-31.8%	0.60167

Overall, the TDNN models while they can produce accurate results for the training set and similar areas like validation set 3 and 5, it is proven by the real operation validation that they cannot be trusted to reproduce the benchmark ICE behaviour. A comparison between the NO_x and the λ value model shows that the NO_x model produces worse results than the λ value model, despite the fact that during training it scores a higher R^2 and it occupies more parameters. As a result, they cannot generalize well enough to be used to drive a controls strategy or onboard diagnostics and that is why the focus is turned to the RNN models.

6.2.2 RNN models for λ value and NO_x

On the other hand the RNN models score relatively high R_2 values on the part load tests and they stay consistent in the two tests that are stimulated with unfamiliar to them inputs quantities patterns. Especially the NO_x model does not score less than 0.92 R^2 even for the validation test that is unknown to it. The λ value RNN is less accurate than then NO_x RNN and this is mainly because the NO_x model uses one more input with dynamic behaviour, i.e the λ value from sensor. Therefore the λ valuemodel depends in two static inputs, the engine speed and torque command pointers, and draws its dynamics partially from the manifold absolute pressure signal and the non-linear activation functions. To sum up, the table 6.20 includes all the R^2 scores for the two RNN models.

The NO_x model not only scores high on R^2 but also follows the dynamics of the NO_x curve from the sensor. The λ value model follows the dynamics adequately but its results

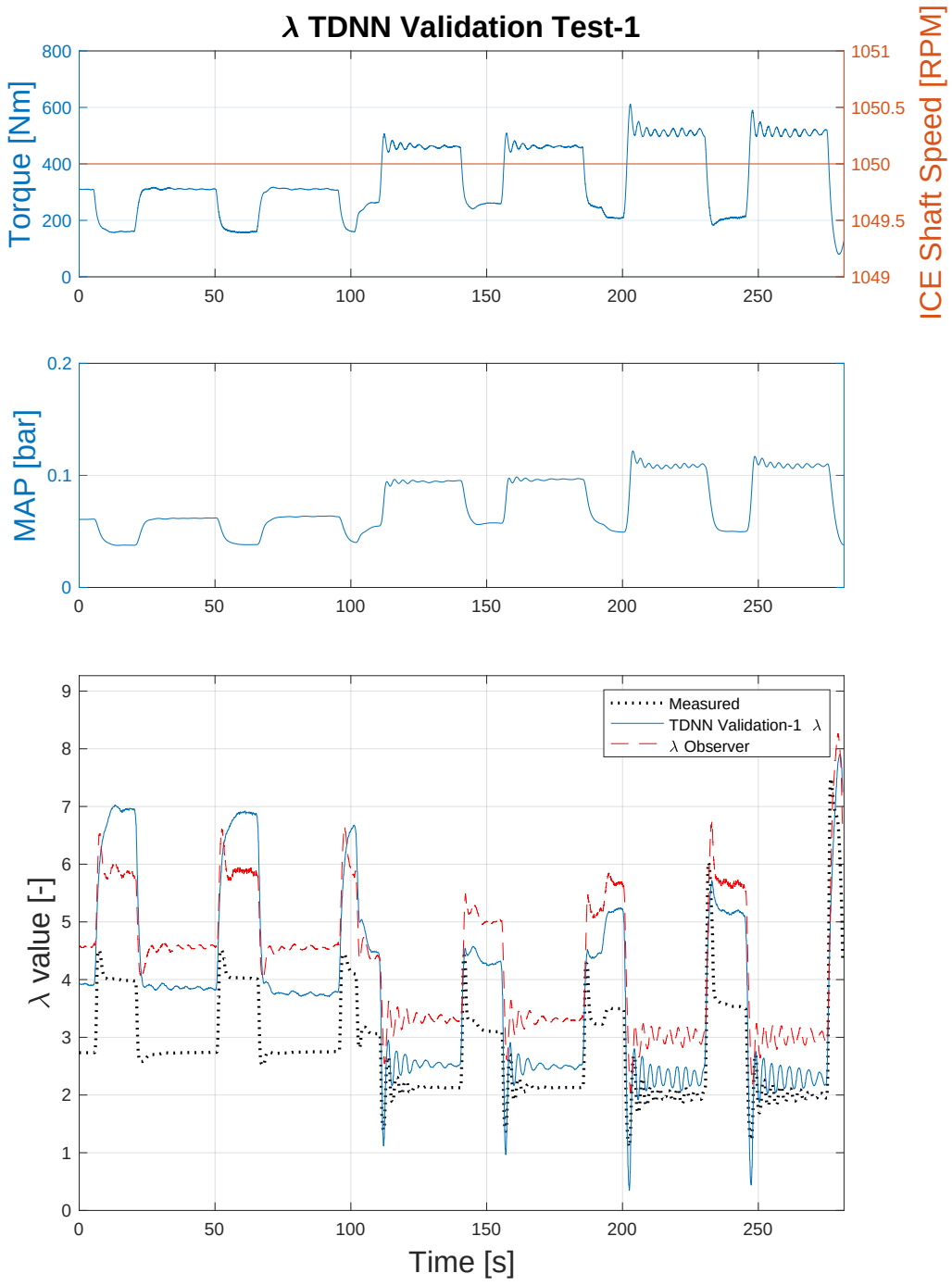

Figure 6.4: λ value TDNN results for validation test 1.

have a significant steady state error which affects negatively the R^2 scoring. The RNN models do not fall apart at the validation sets 7 and 8 in comparison to the TDNN models and it is possible to see that results get noisy during the engine speed and torque changes.

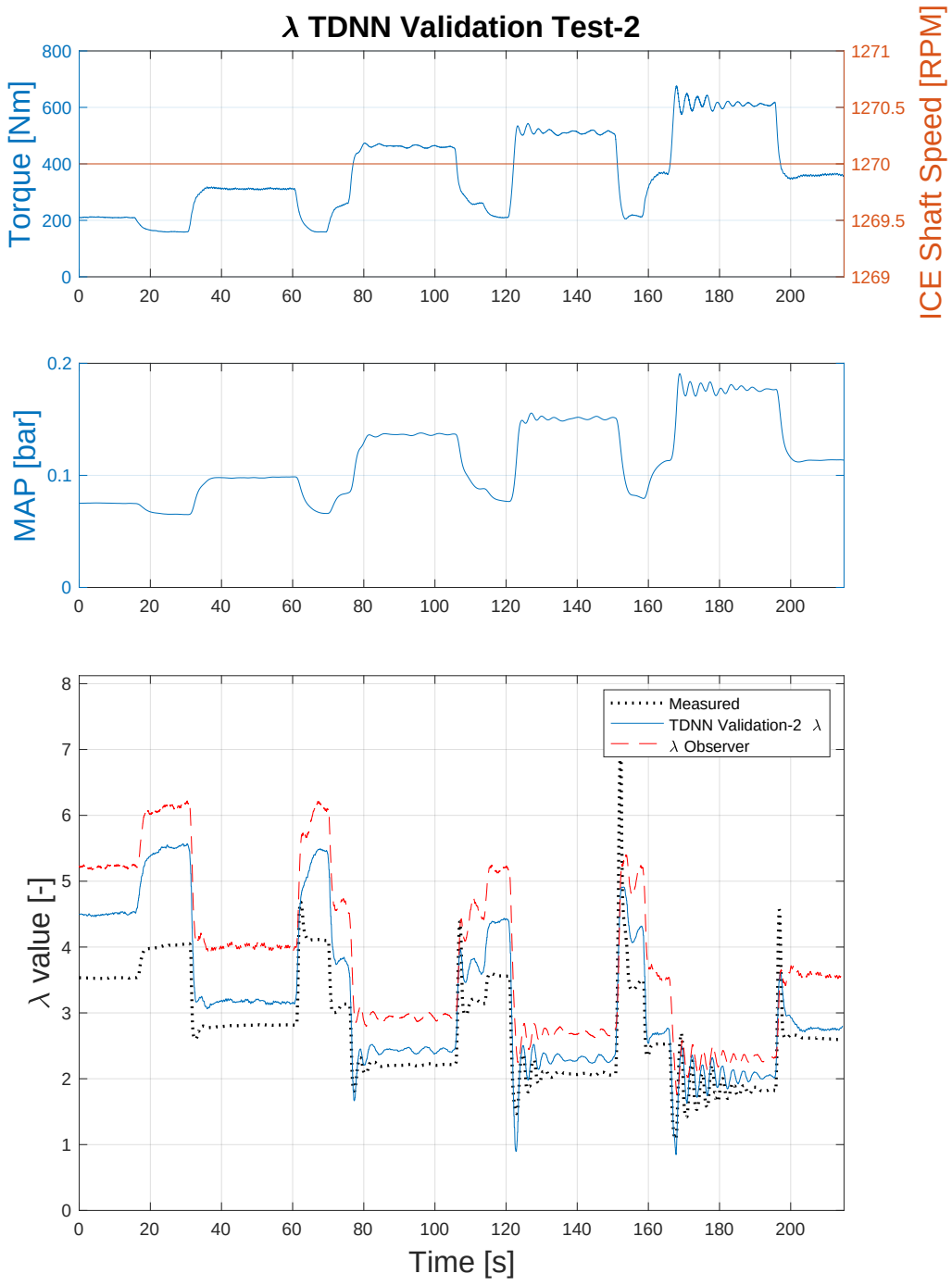

Figure 6.5: λ value TDNN results for validation test 2.

Specifically in the validation set 8, the two models show the same noisy results at 50s, which is an indication that both of the models results become unstable when all the input quantities defer from the training pattern. The same issue is also shown in NO_x model

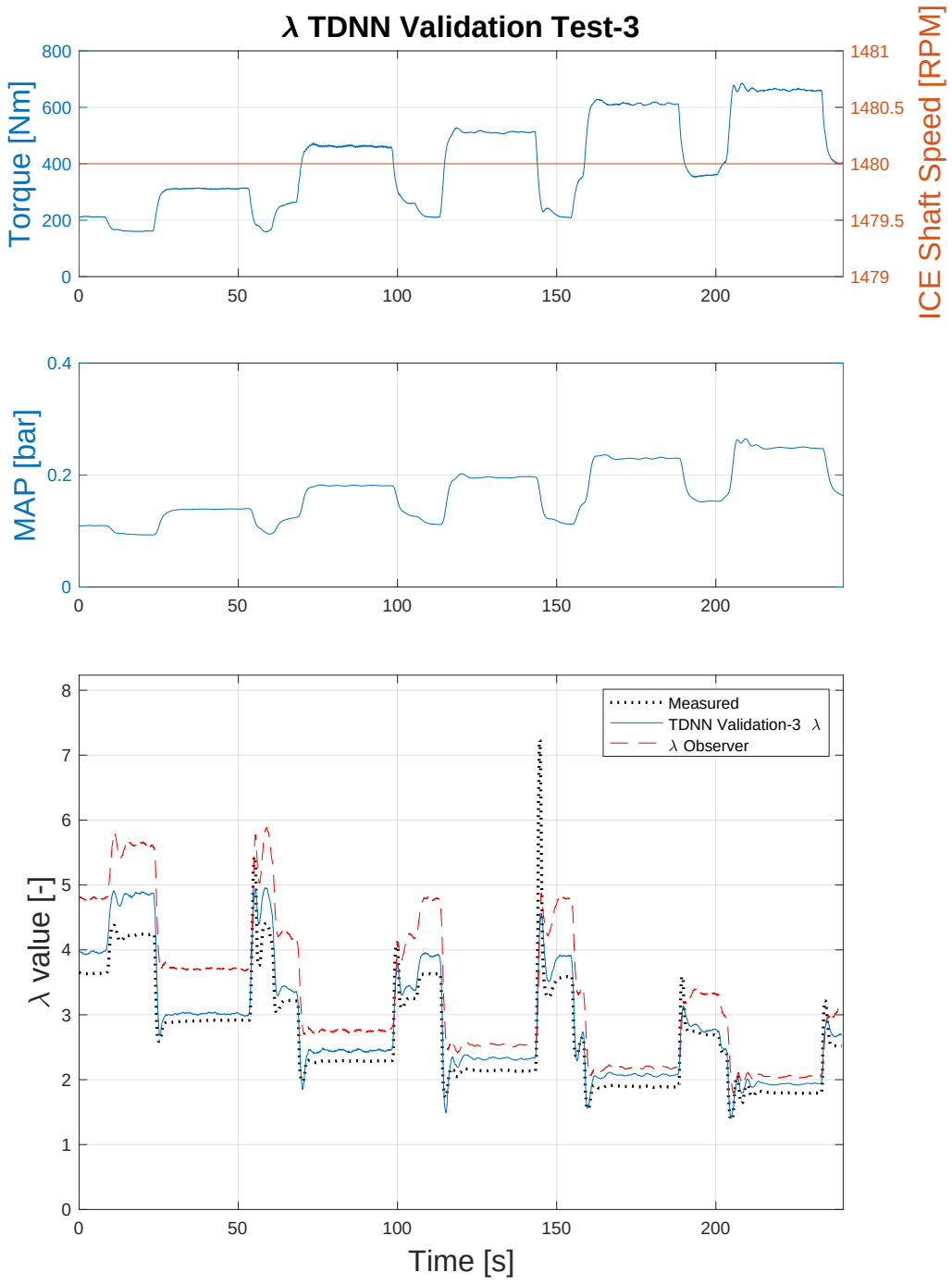

Figure 6.6: λ value TDNN results for validation test 3.

results for validation set 7 at 30s and 100s, but the model results stay accurate. Overall the models results are accurate and follow the dynamics adequately as shown in the figures from Fig. 6.21 to Fig. 6.36.

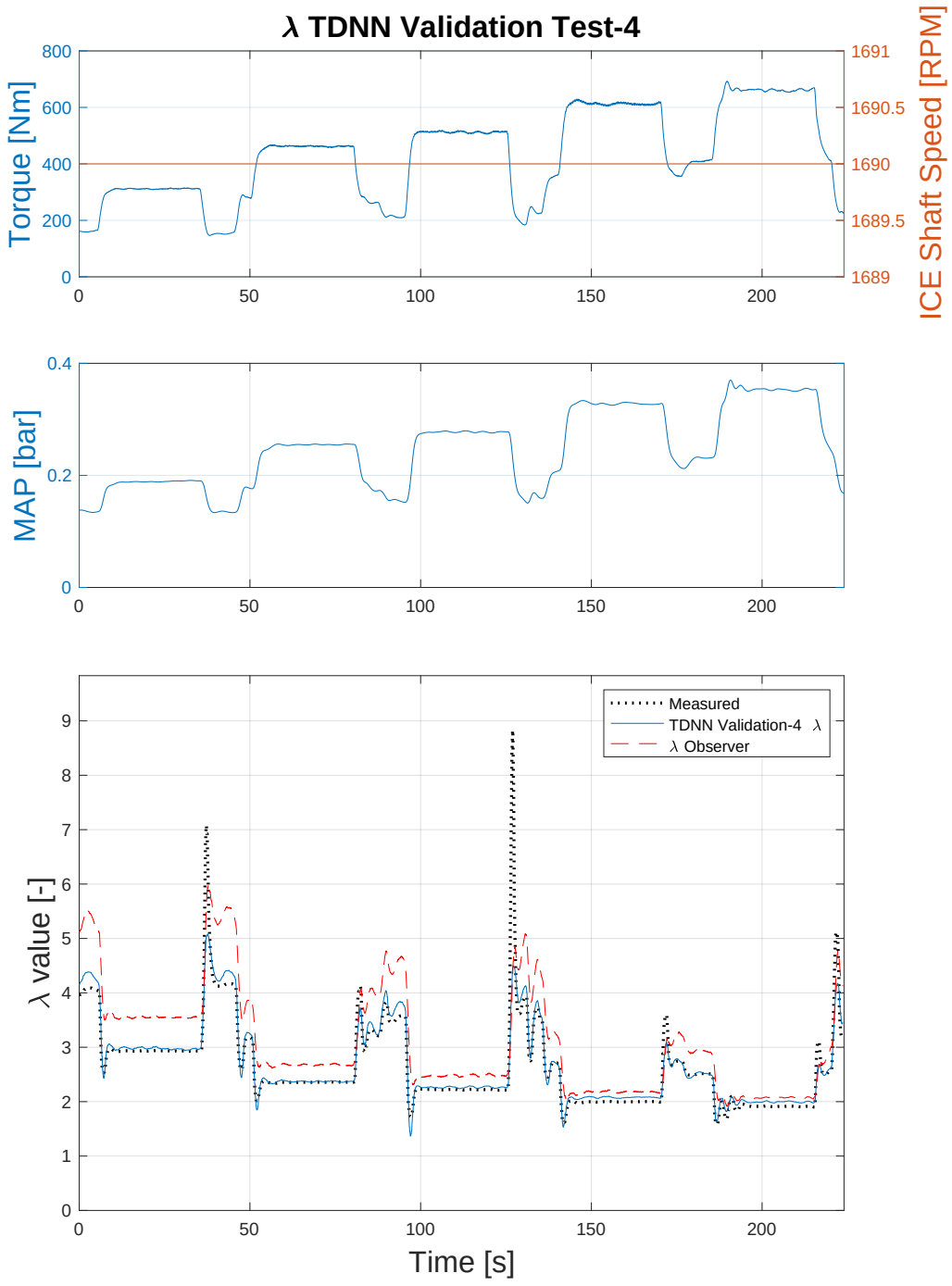

Figure 6.7: λ value TDNN results for validation test 4.

In conclusion, both of the models are affected by continuous changes to their static inputs and their accuracy deteriorates. The λ value is less accurate than the NO_x model, but it still follows the dynamic changes correctly. The RNN models are qualified to be

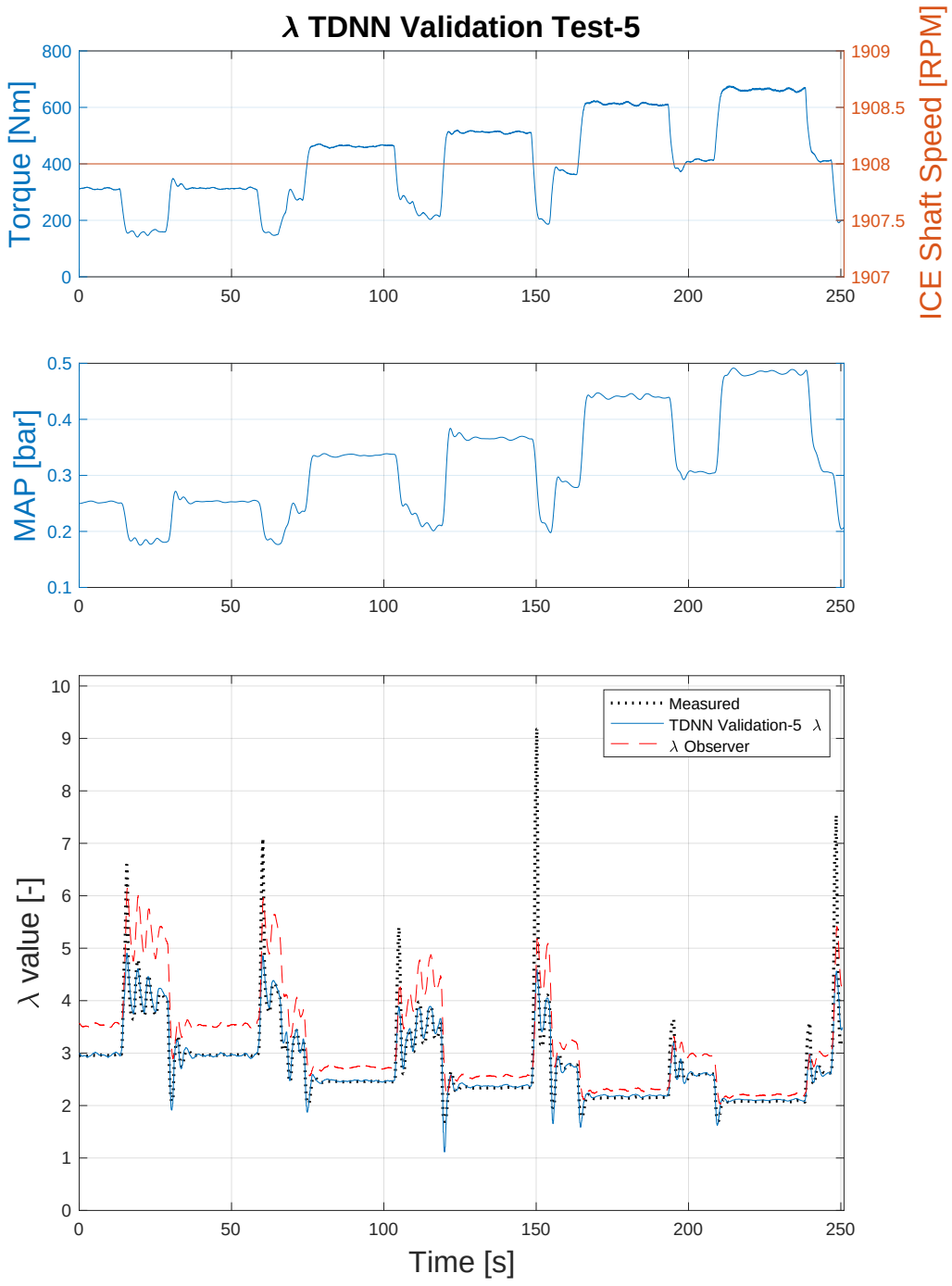

Figure 6.8: λ value TDNN results for validation test 5.

used as virtual sensor because their results are accurate and stay consistent with the target values non-linearities.

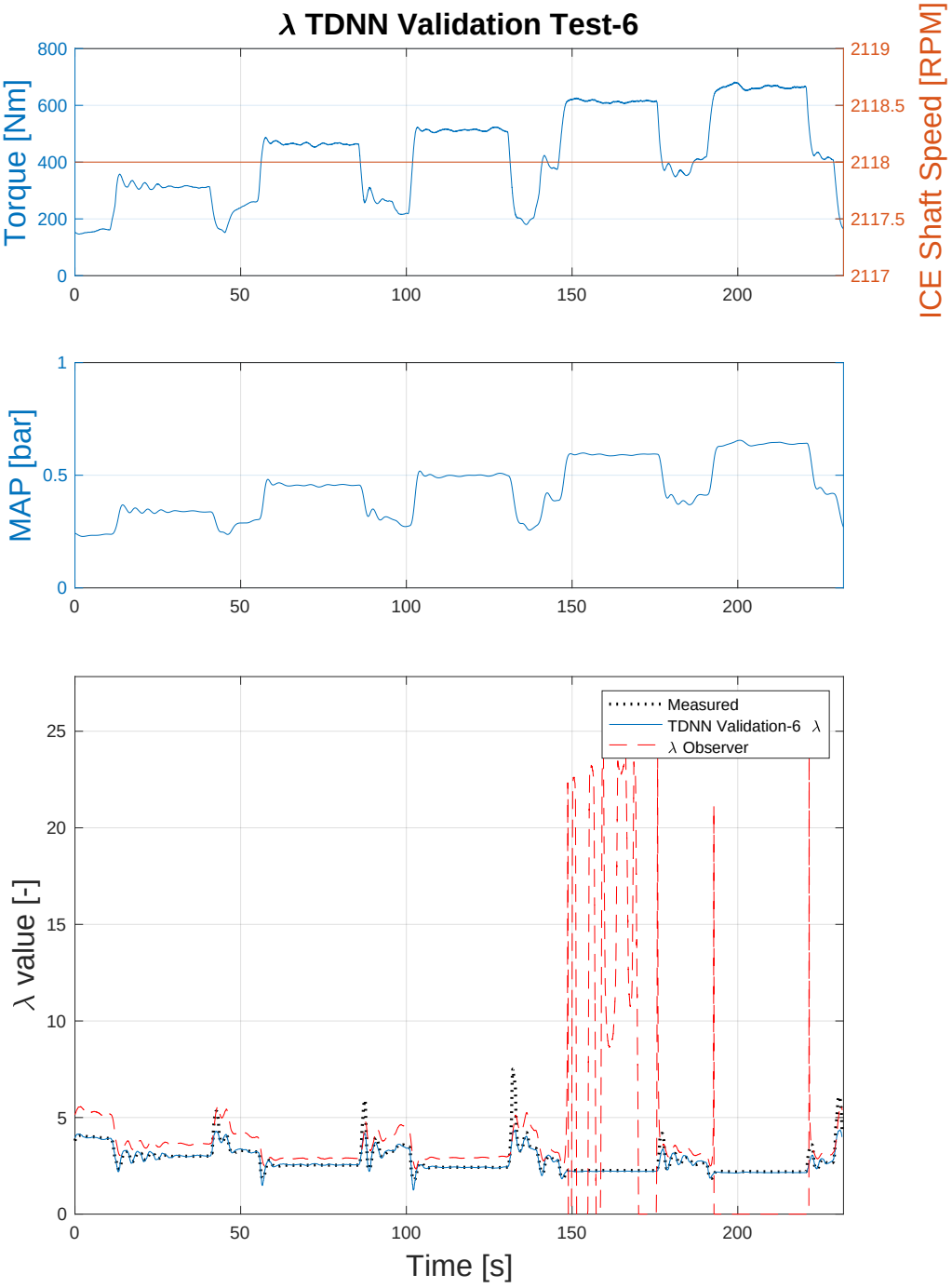

Figure 6.9: λ value TDNN results for validation test 6.

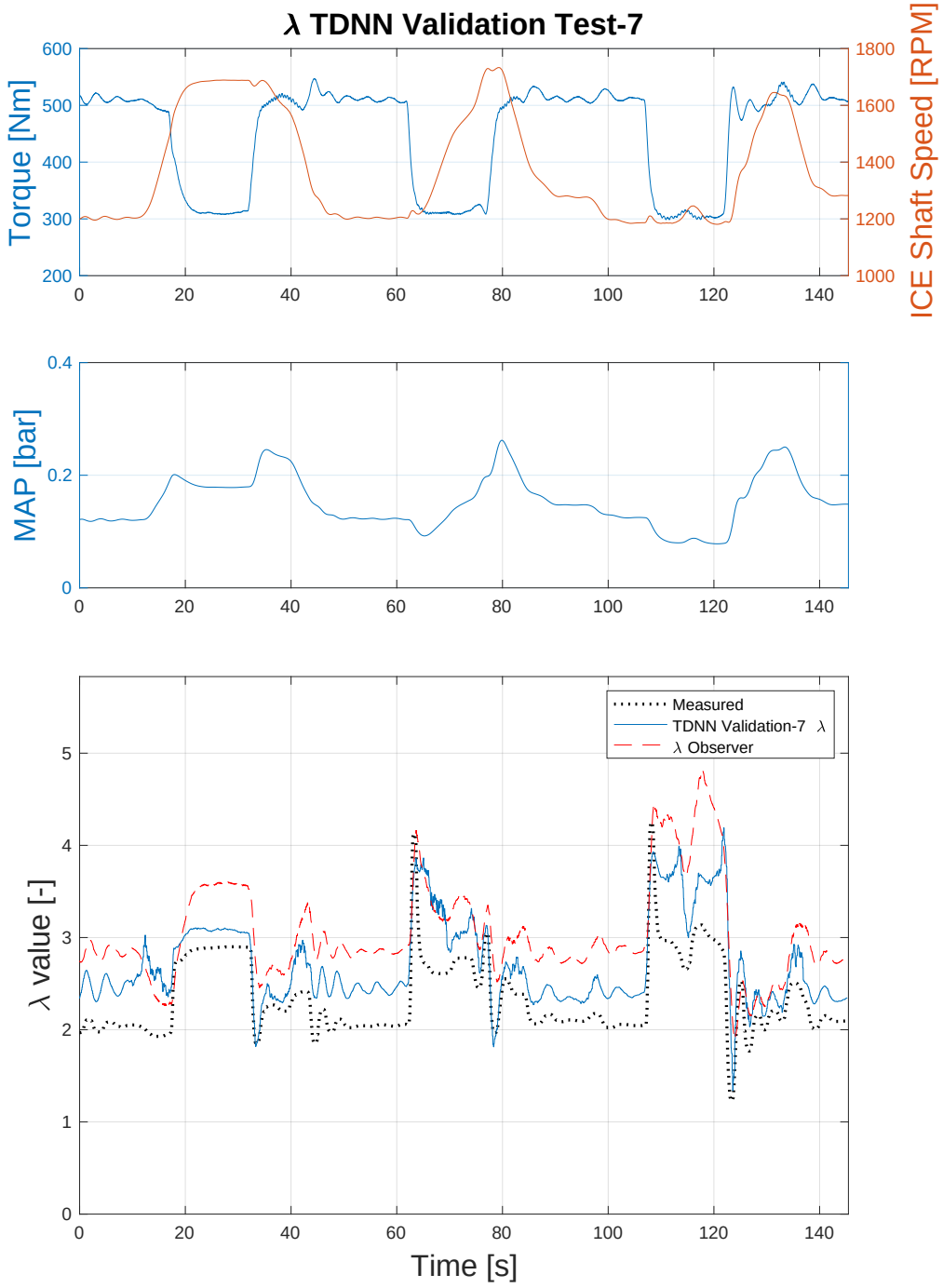

Figure 6.10: λ value TDNN results for validation test 7.

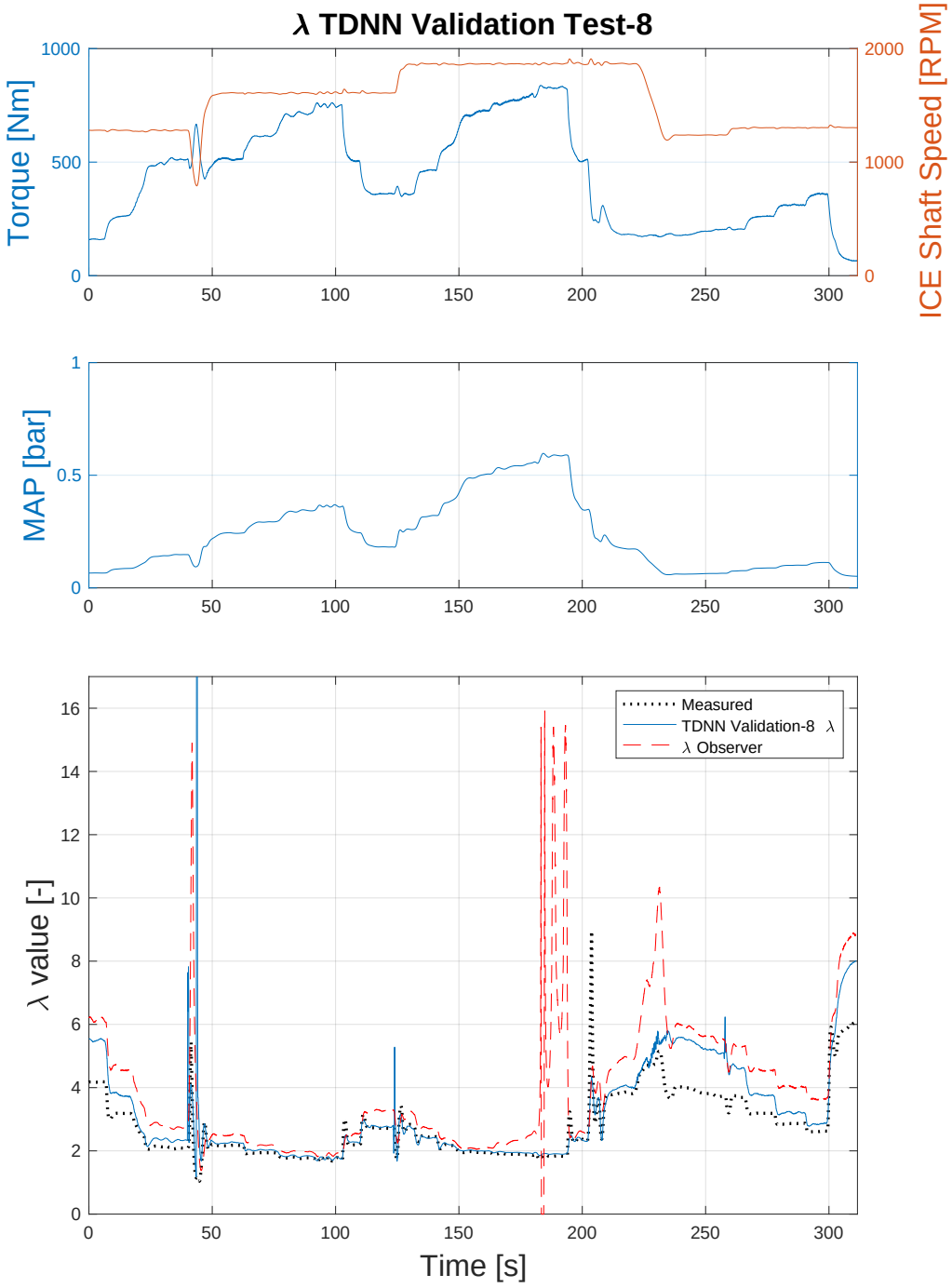

Figure 6.11: λ value TDNN results for validation test 8.

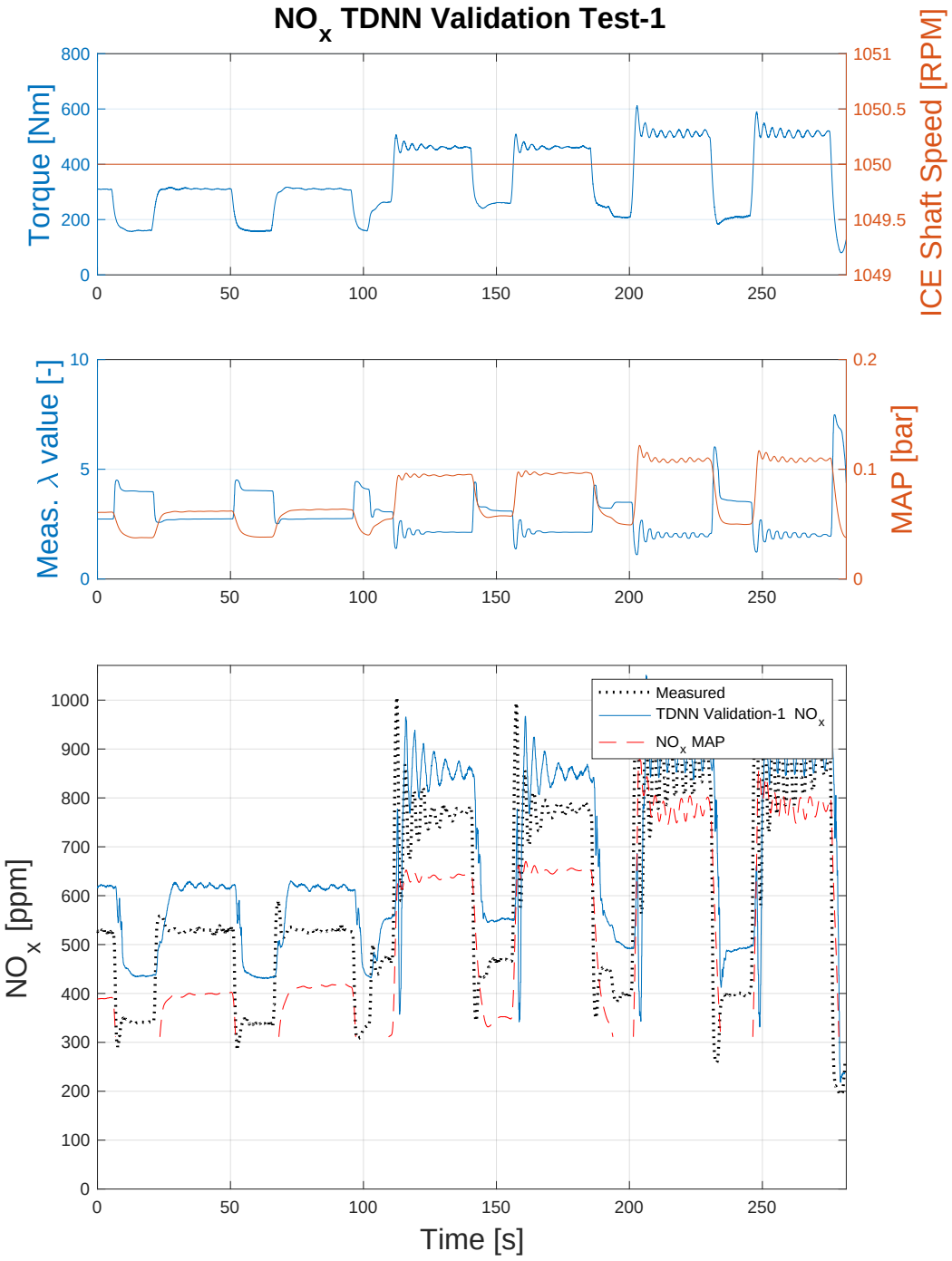

Figure 6.12: NO_x TDNN results for validation test 1.

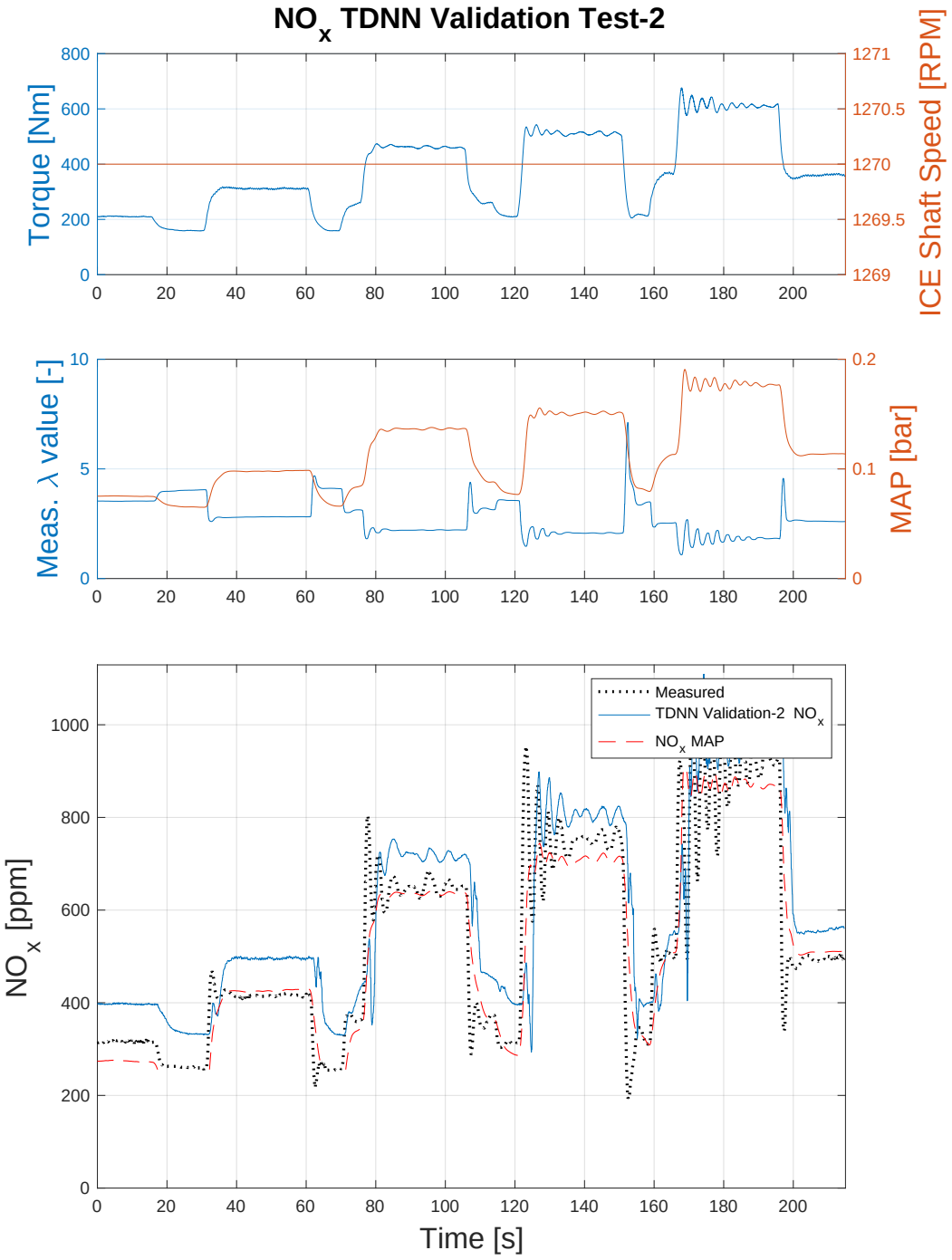

Figure 6.13: NO_x TDNN results for validation test 2.

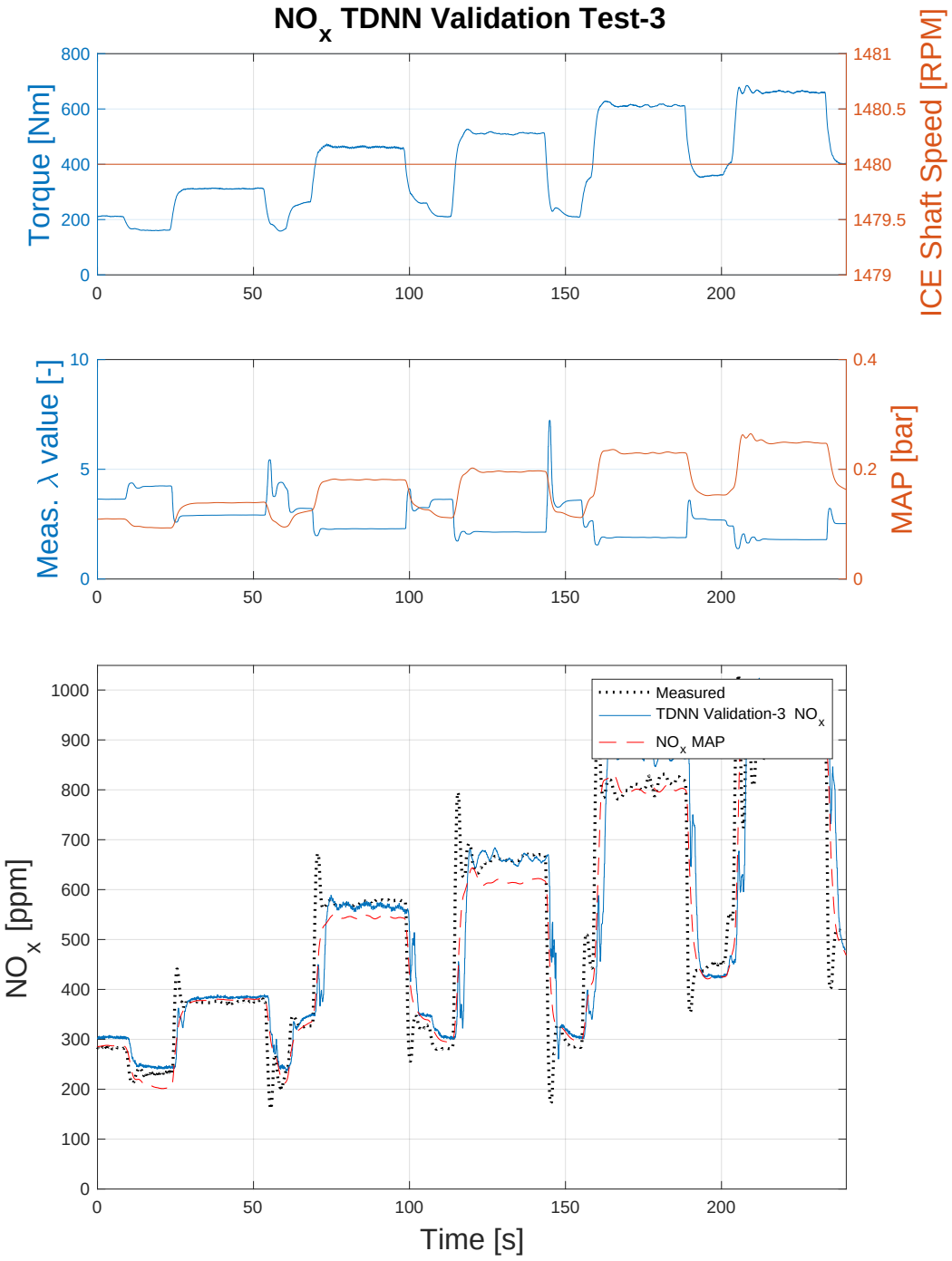

Figure 6.14: NO_x TDNN results for validation test 3.

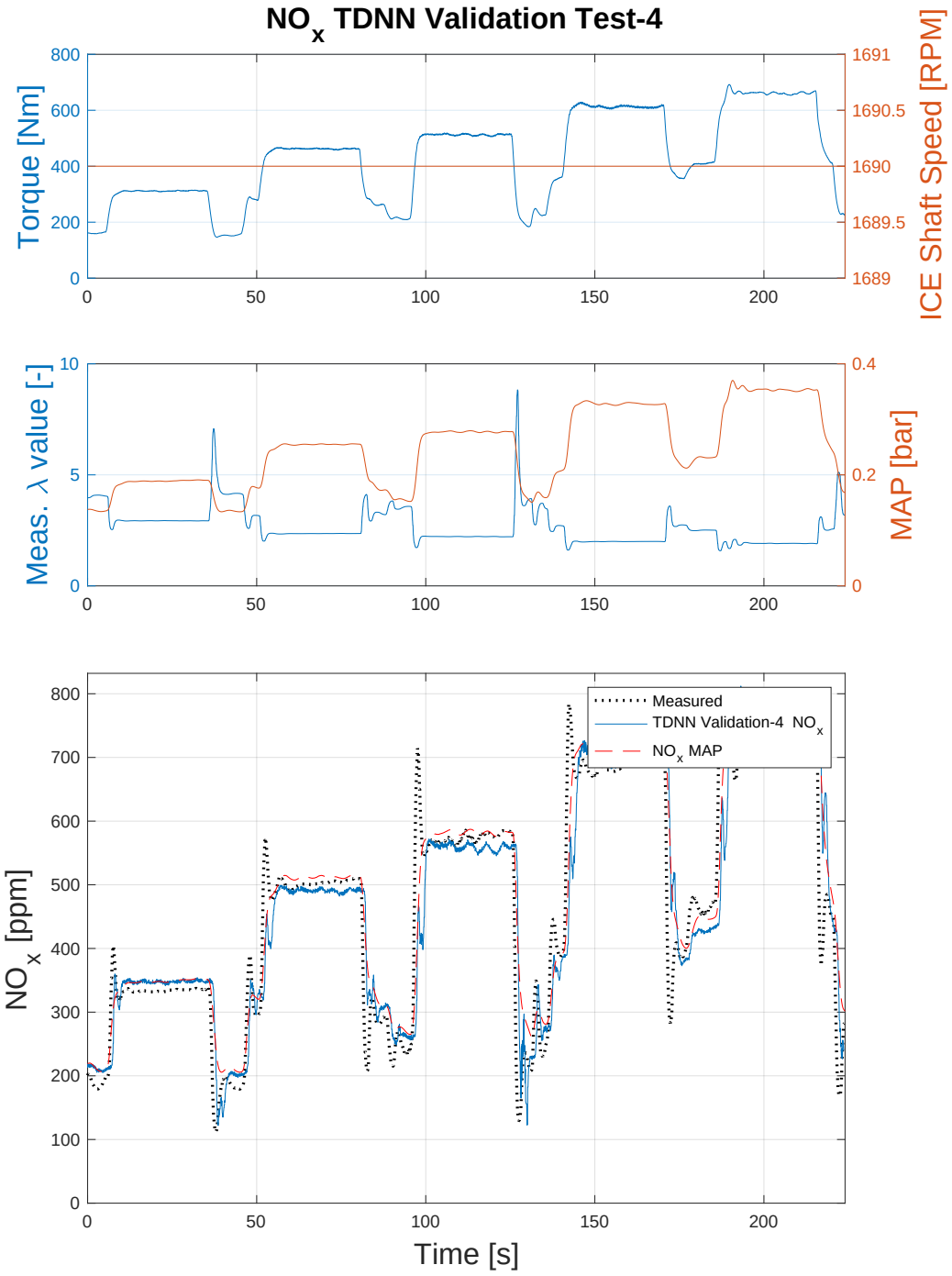

Figure 6.15: NO_x TDNN results for validation test 4.

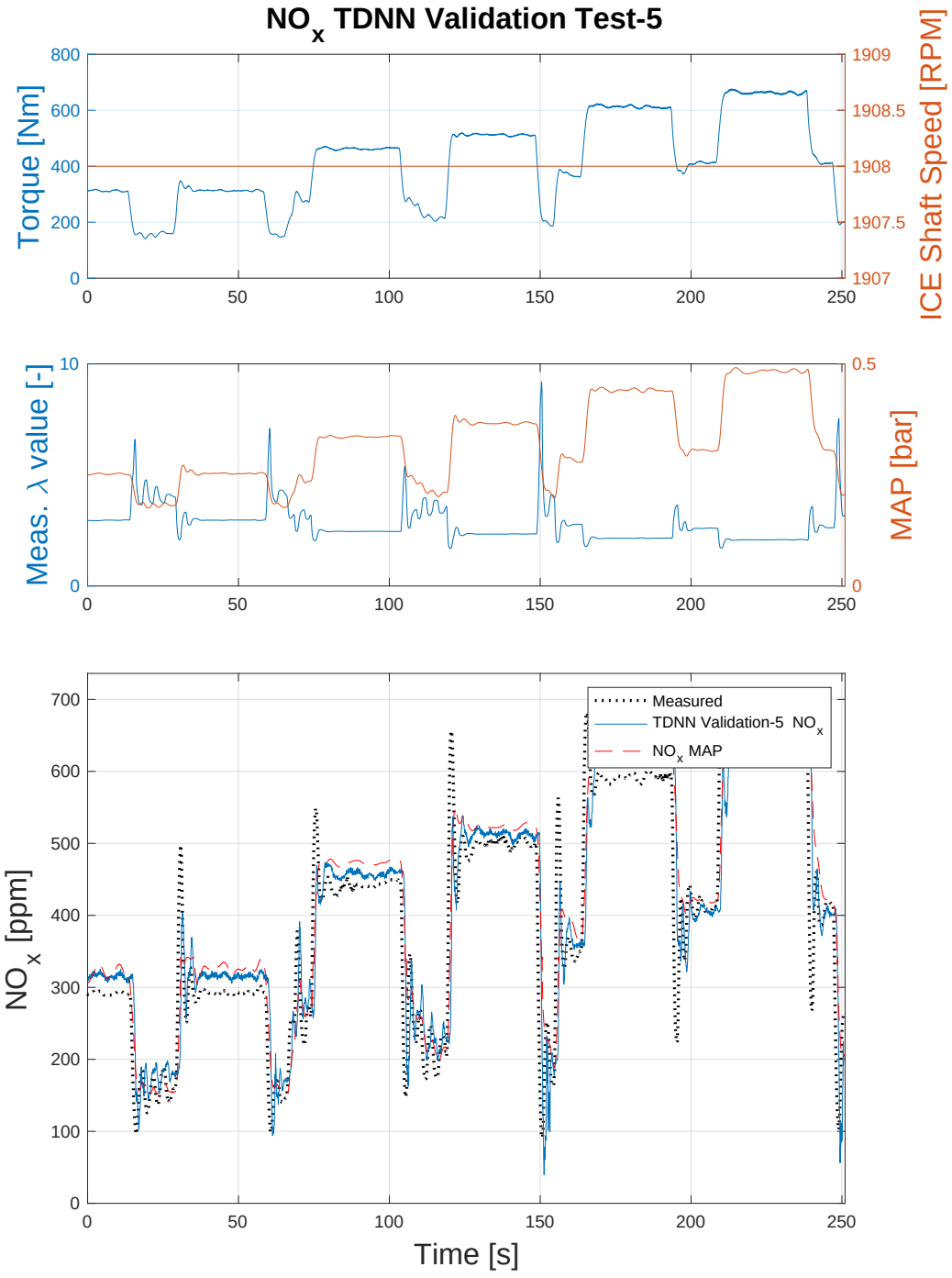

Figure 6.16: NO_x TDNN results for validation test 5.

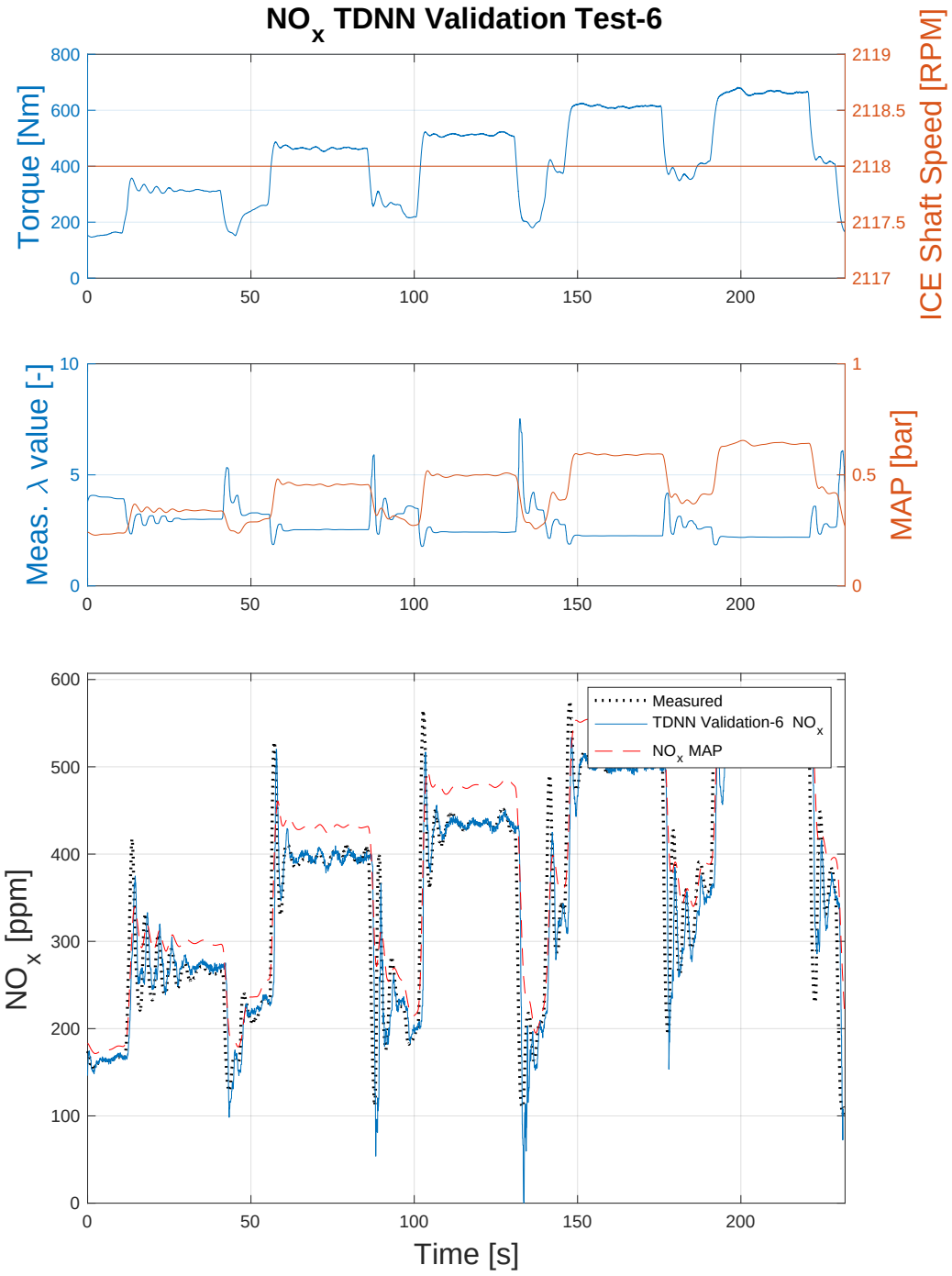

Figure 6.17: NO_x TDNN results for validation test 6.

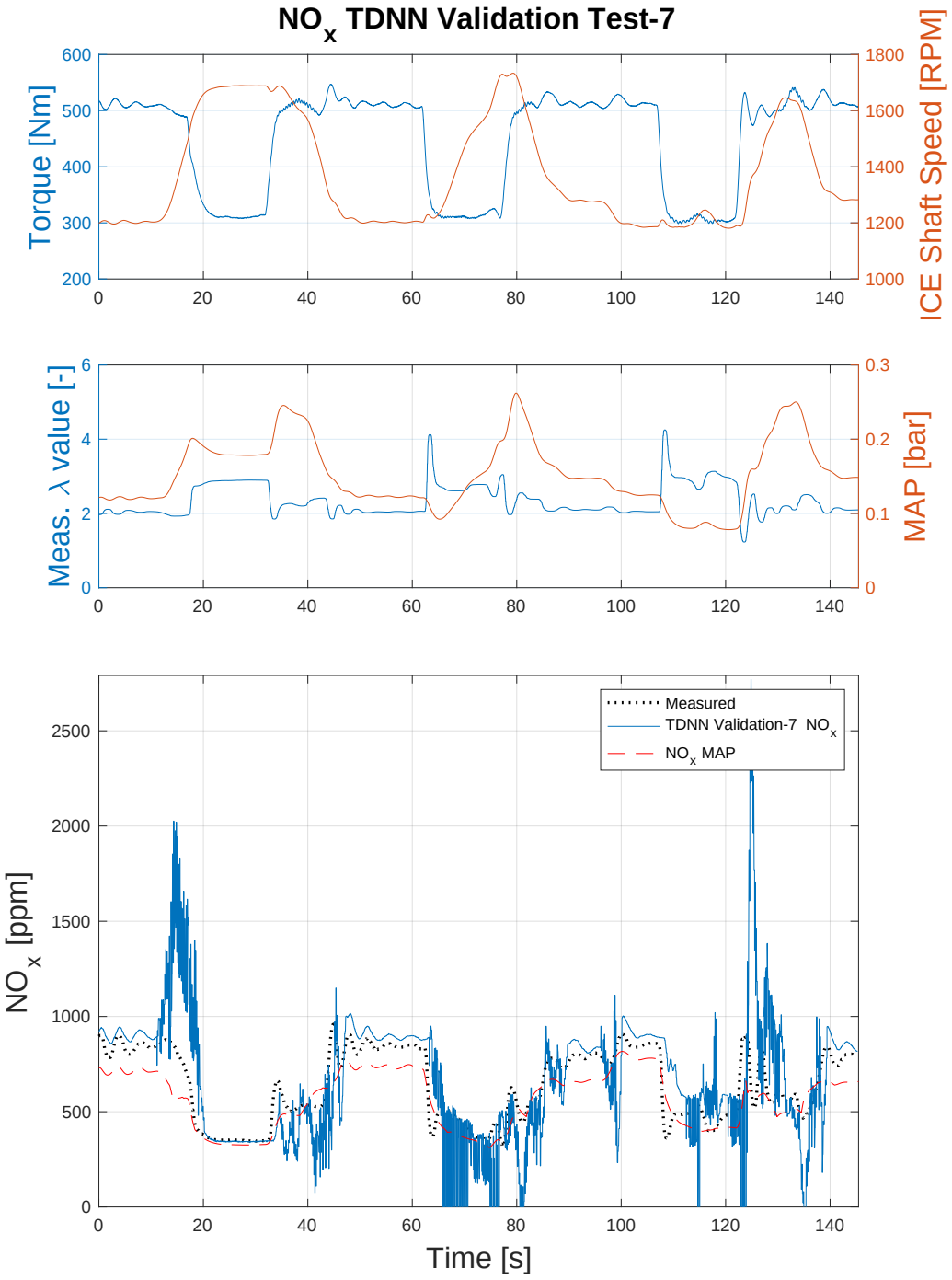

Figure 6.18: NO_x TDNN results for validation test 7.

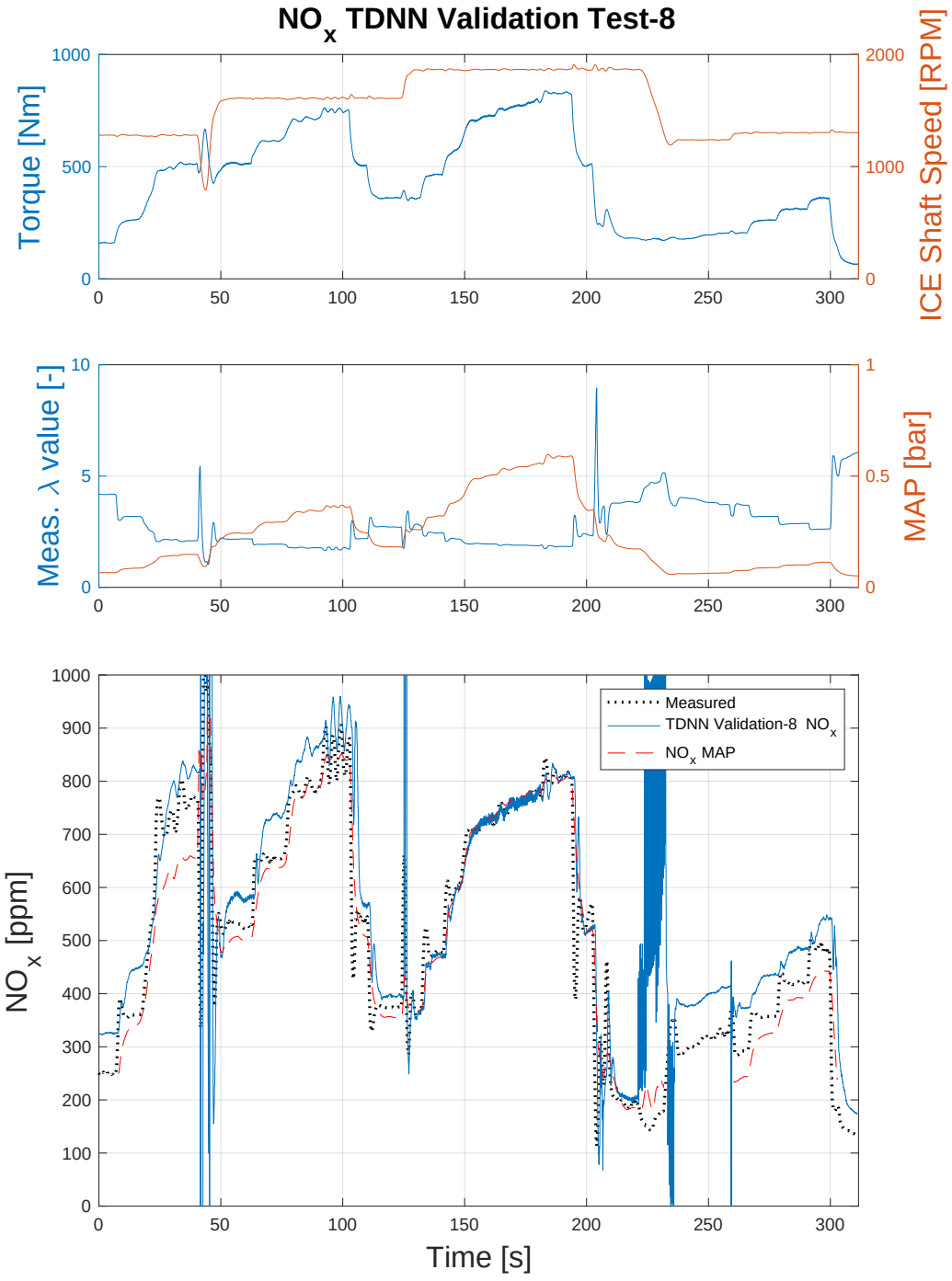

Figure 6.19: NO_x TDNN results for validation test 8.

Figure 6.20: R^2 accuracy score of the NN models during validation experiments.

Dataset	NO_x RNN	Cum. NO_x % Error	λ value RNN
Valid. exp. 1	0.9315	-8.49%	0.7723
Valid. exp. 2	0.9859	-9.13%	0.83527
Valid. exp. 3	0.99138	-3.71%	0.9026
Valid. exp. 4	0.9888	-4.75%	0.9044
Valid. exp. 5	0.9854	-1.26%	0.9023
Valid. exp. 6	0.9777	-0.19%	0.94257
Valid. exp. 7	0.91801	-0.88%	0.7078
Valid. exp. 8	0.9466	-7.78%	0.58854

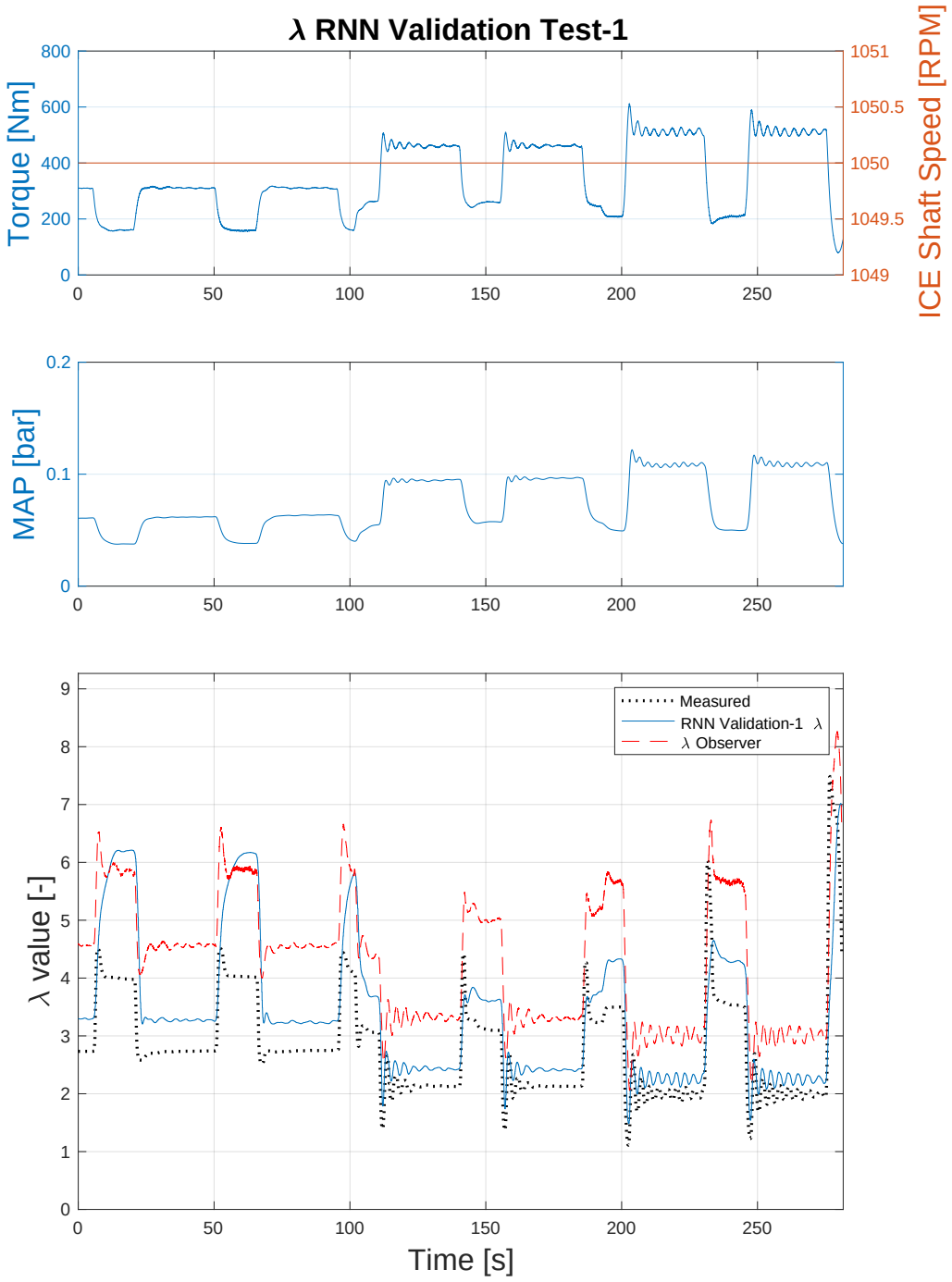

Figure 6.21: λ value RNN results for validation test 1.

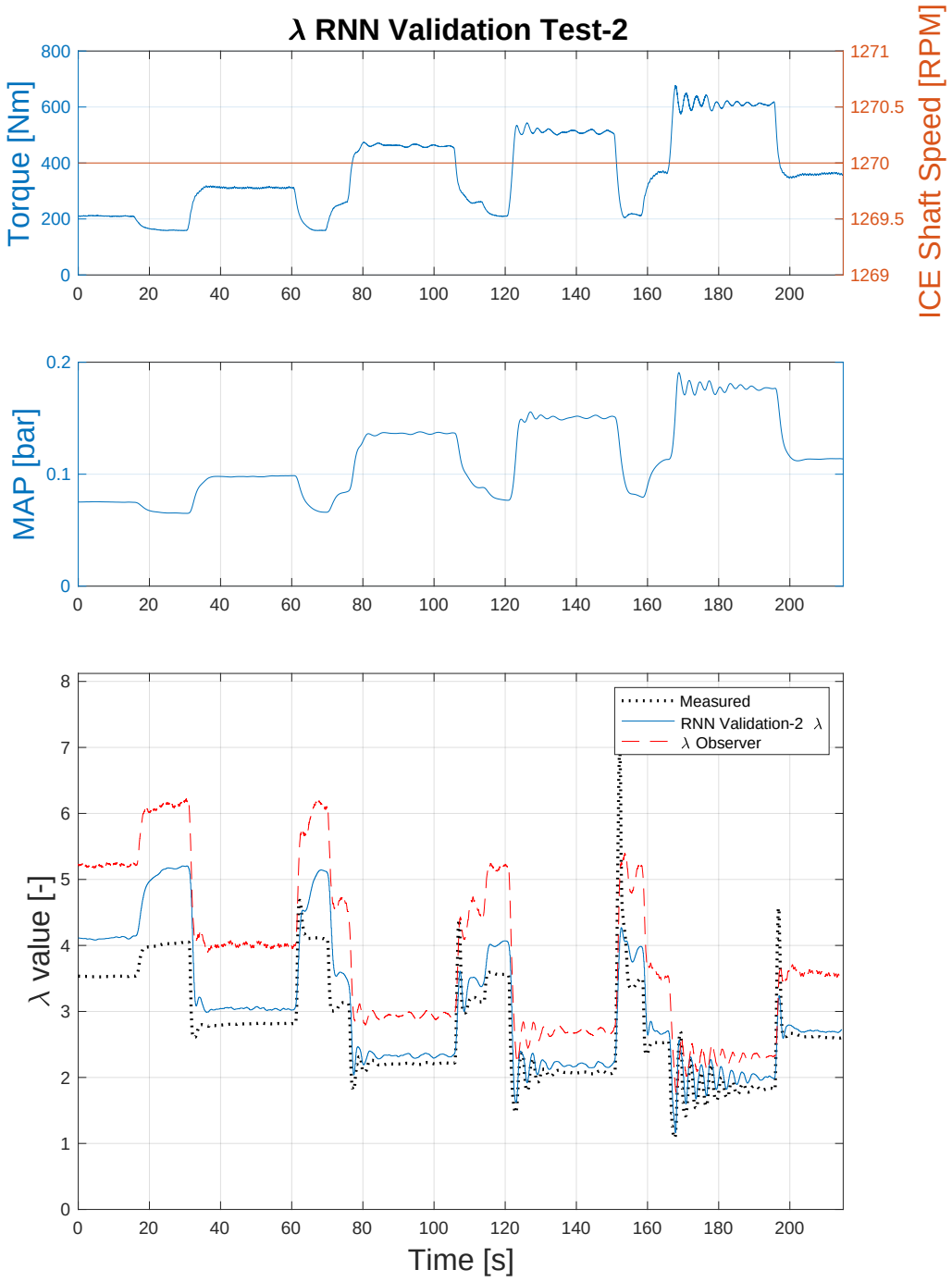

Figure 6.22: λ value RNN results for validation test 2.

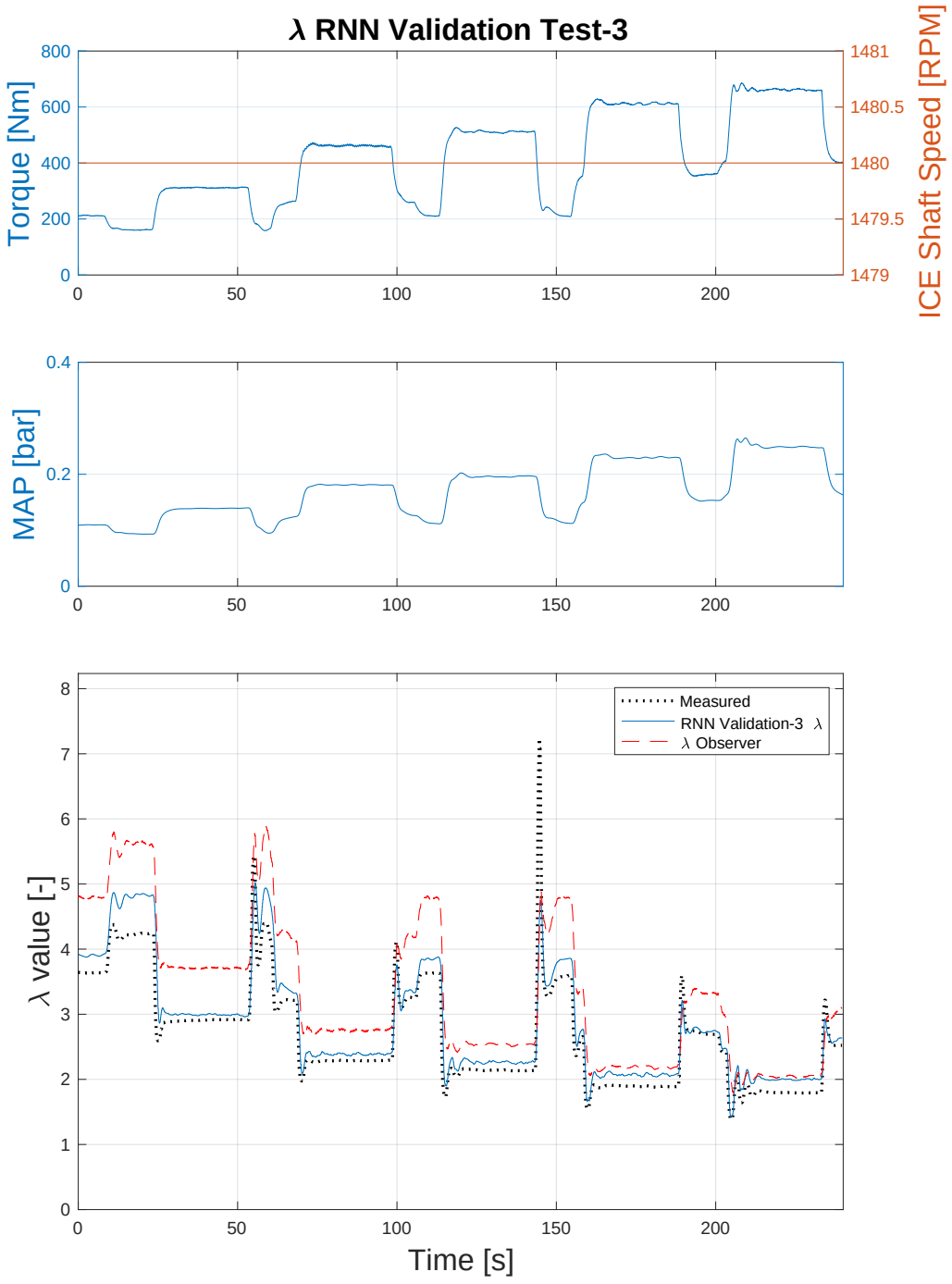

Figure 6.23: λ value RNN results for validation test 3.

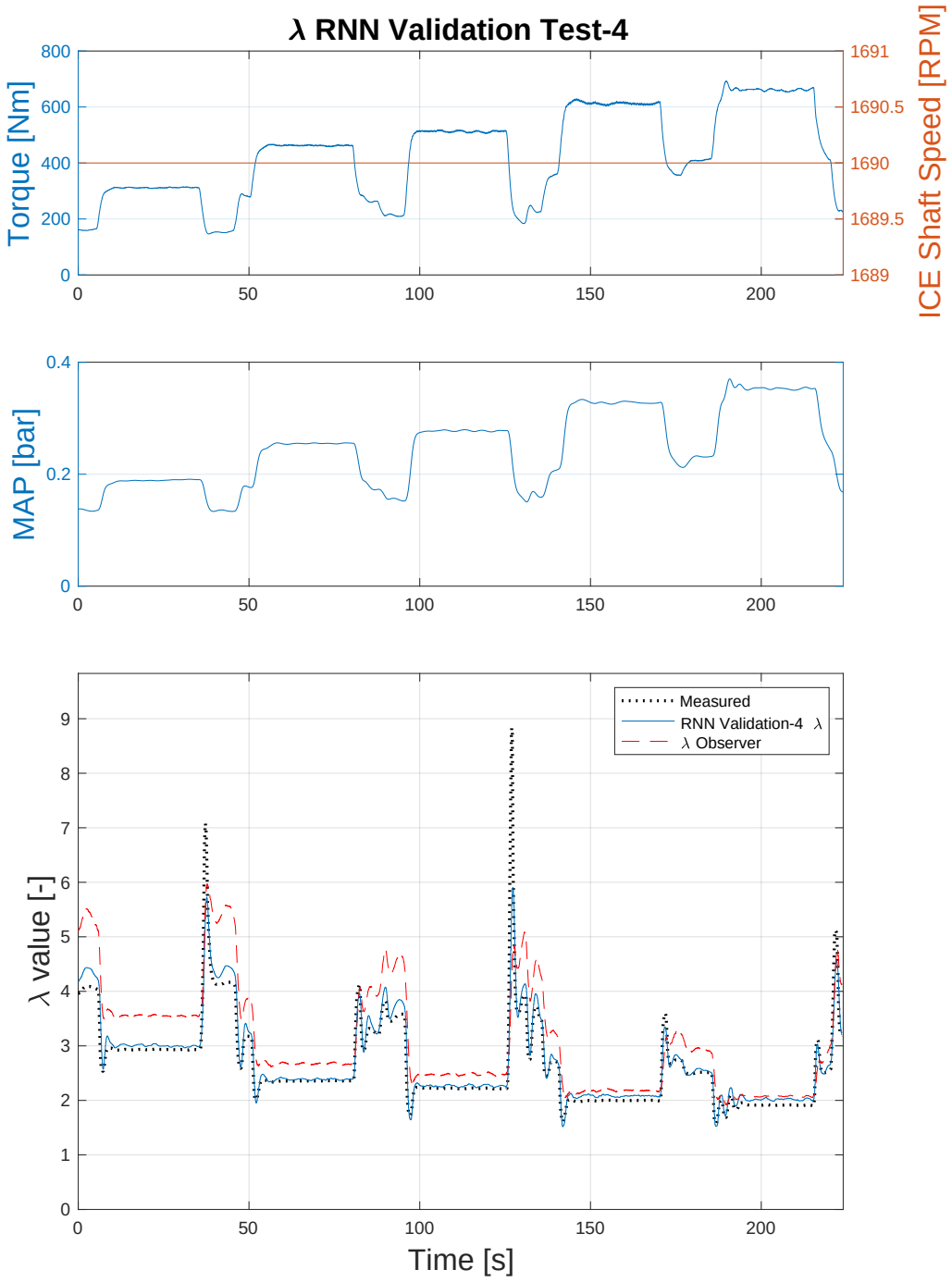

Figure 6.24: λ value RNN results for validation test 4.

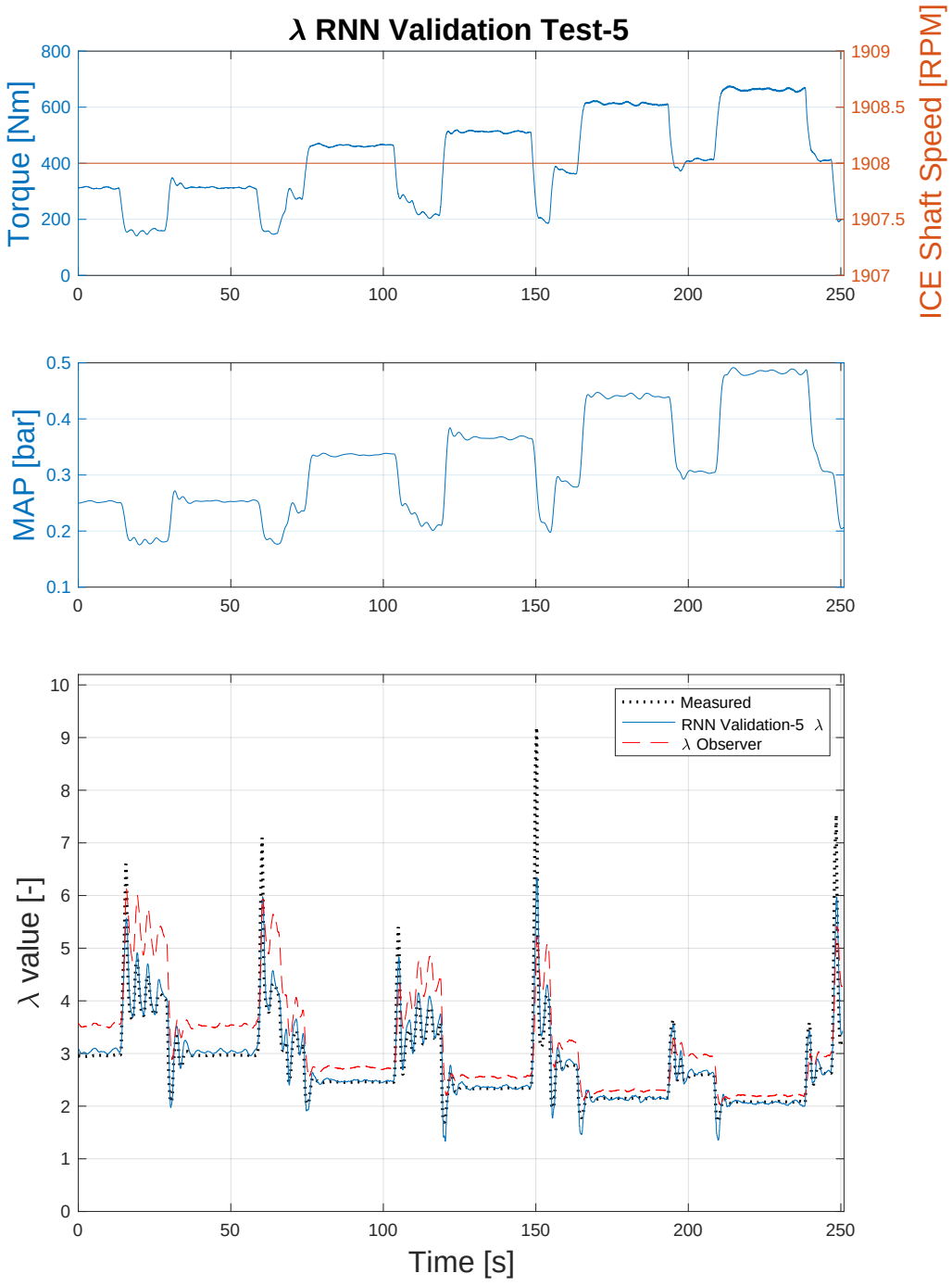

Figure 6.25: λ value RNN results for validation test 5.

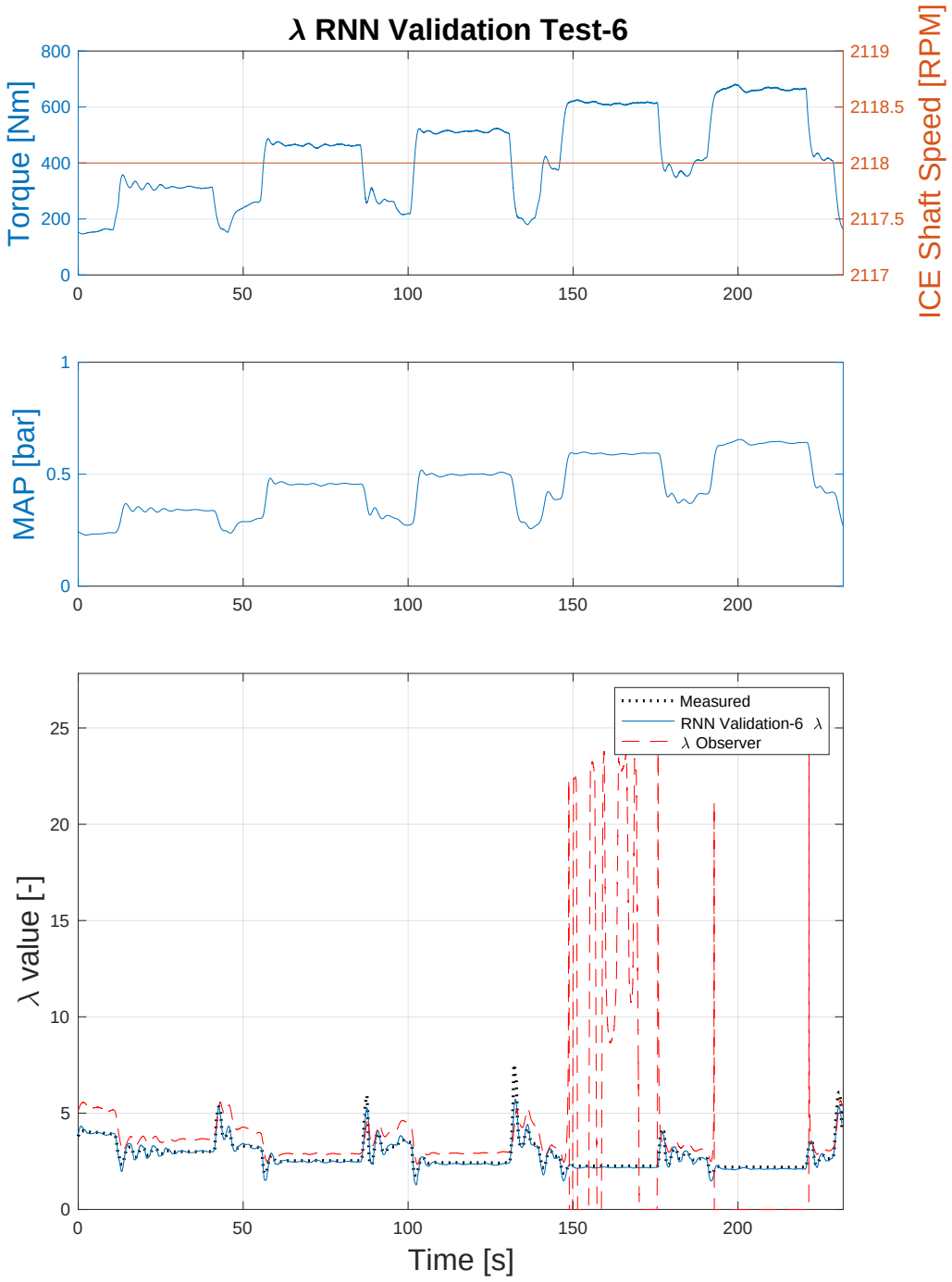

Figure 6.26: λ value RNN results for validation test 6.

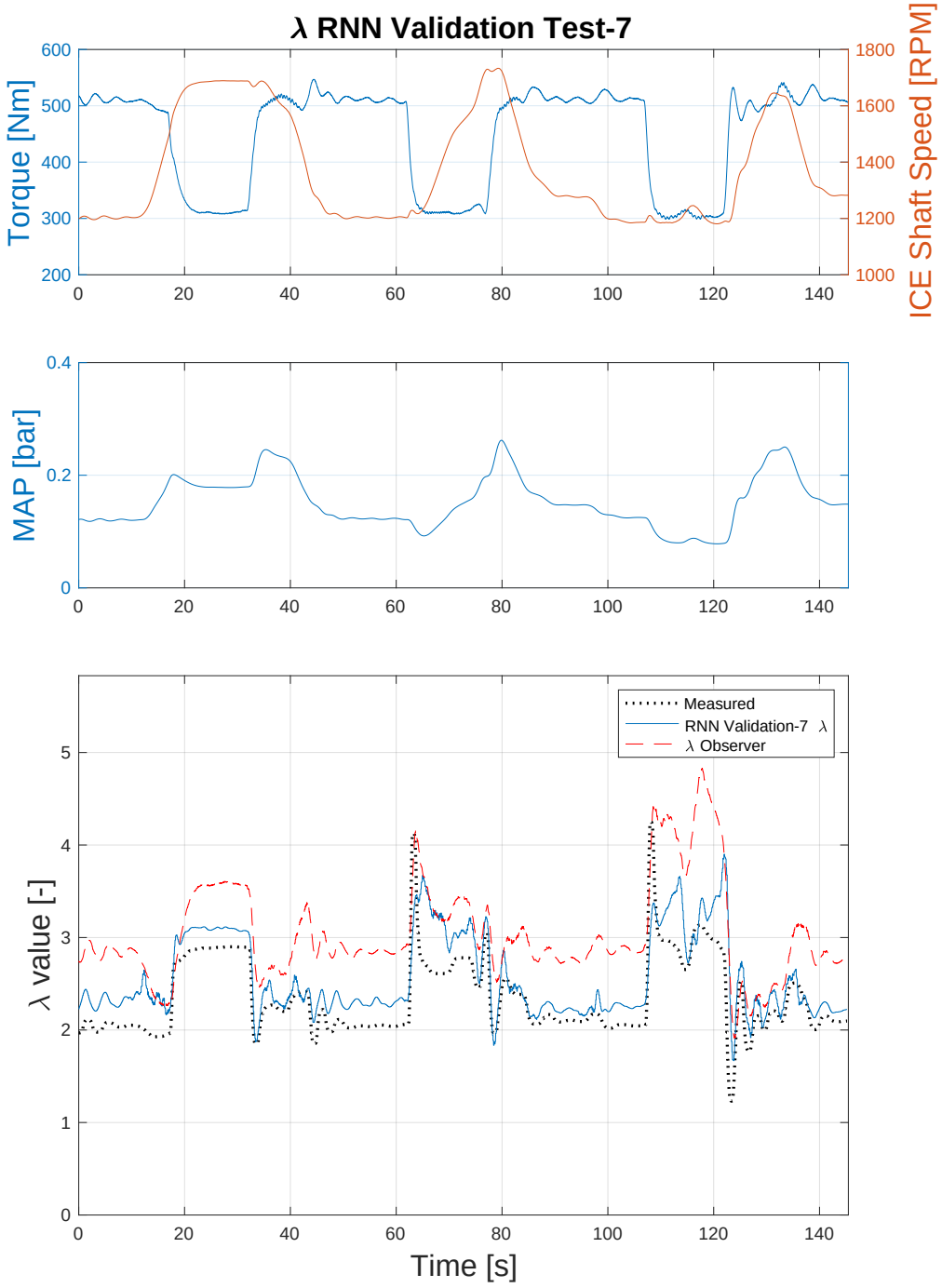

Figure 6.27: λ value RNN results for validation test 7.

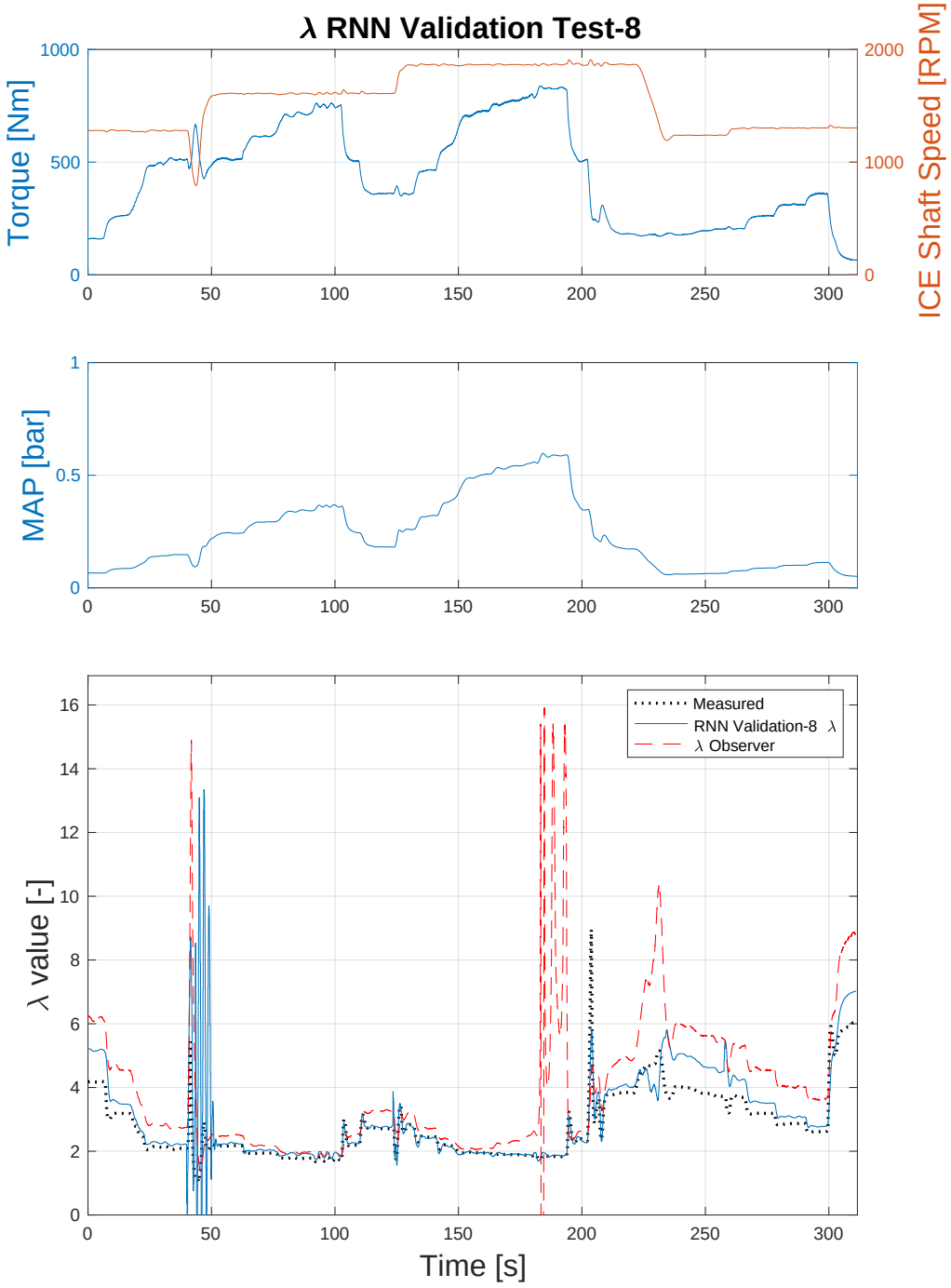

Figure 6.28: λ value RNN results for validation test 8.

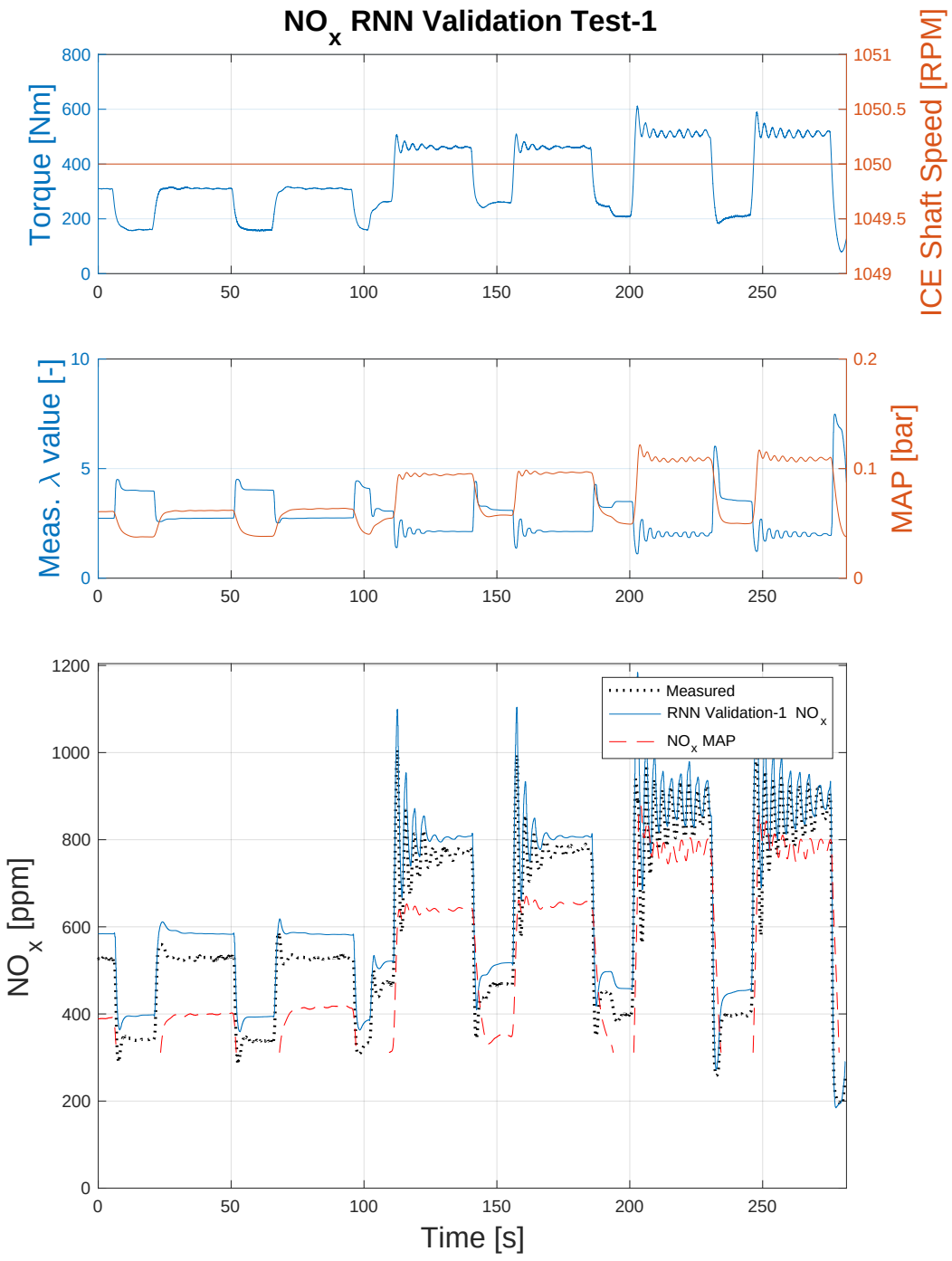

Figure 6.29: NO_x RNN results for validation test 1.

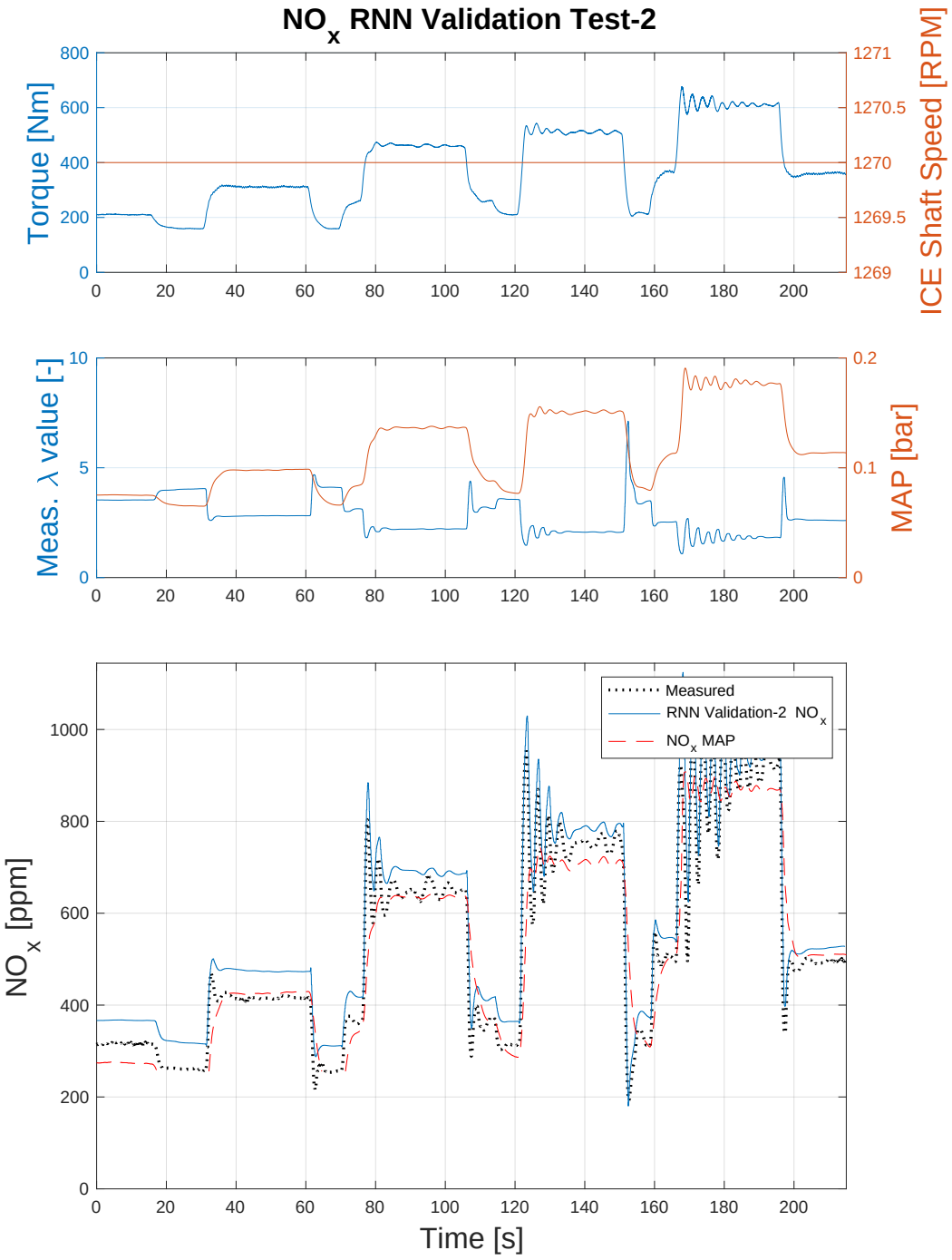

Figure 6.30: NO_x RNN results for validation test 2.

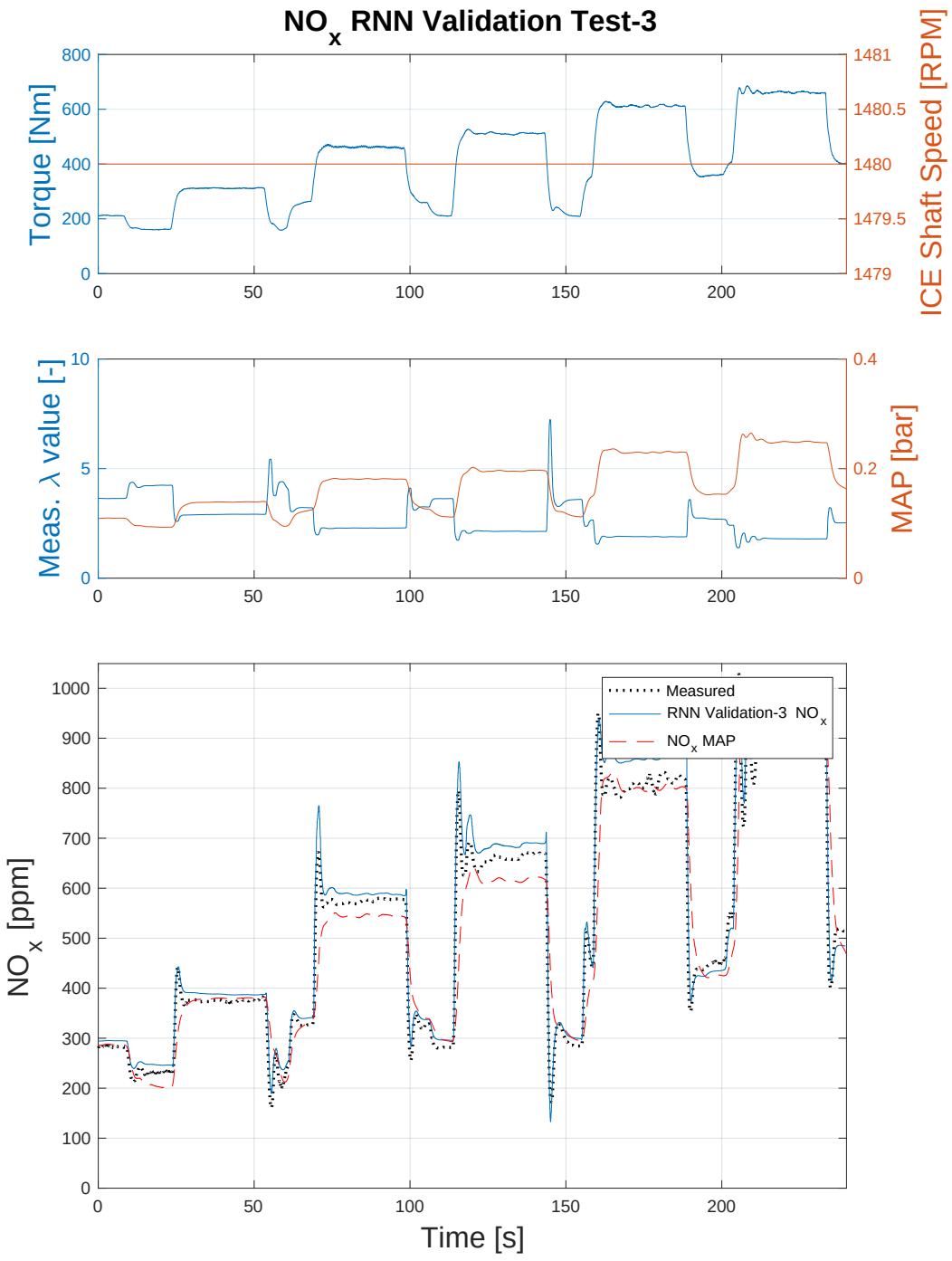

Figure 6.31: NO_x RNN results for validation test 3.

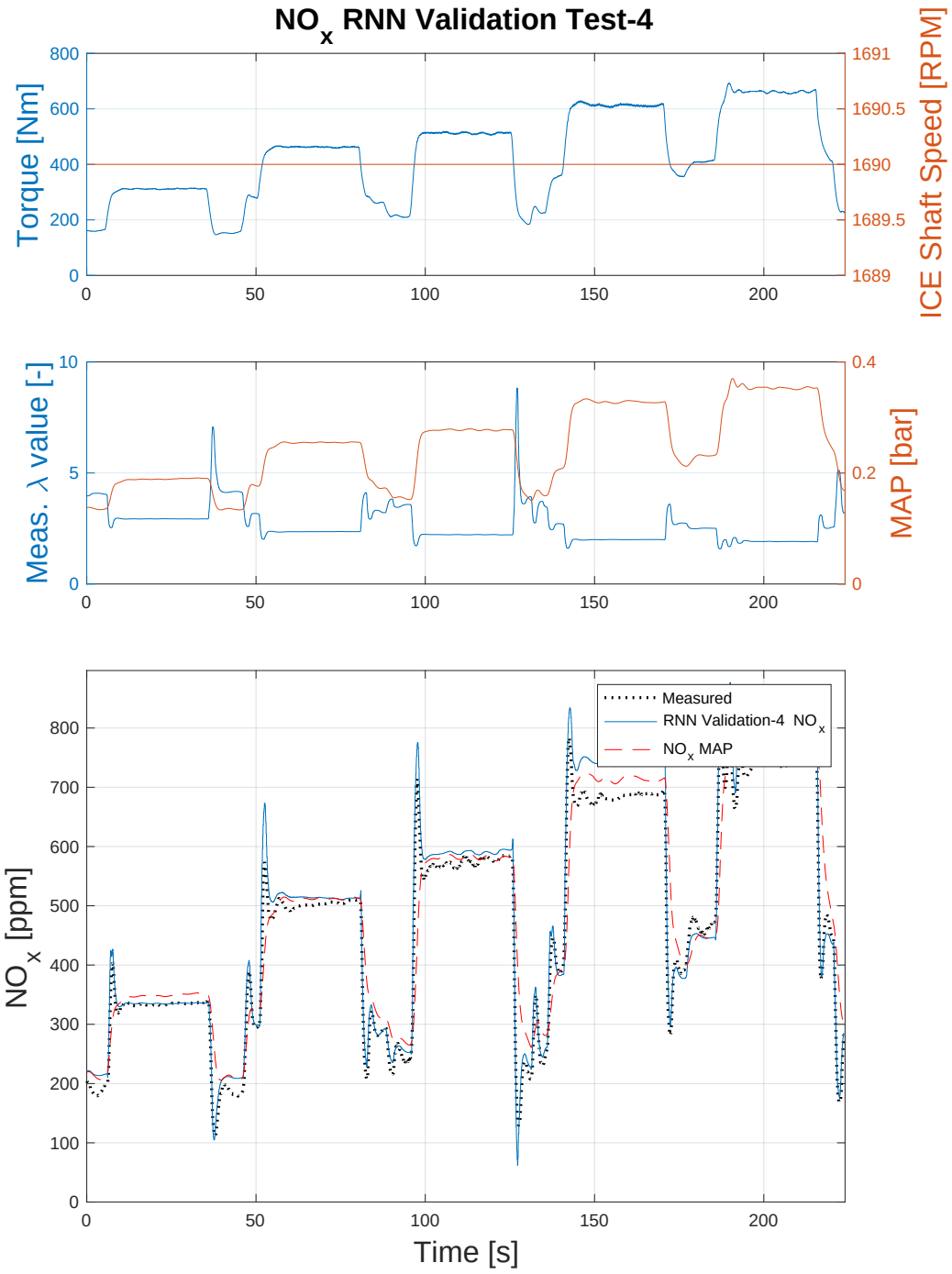

Figure 6.32: NO_x RNN results for validation test 4.

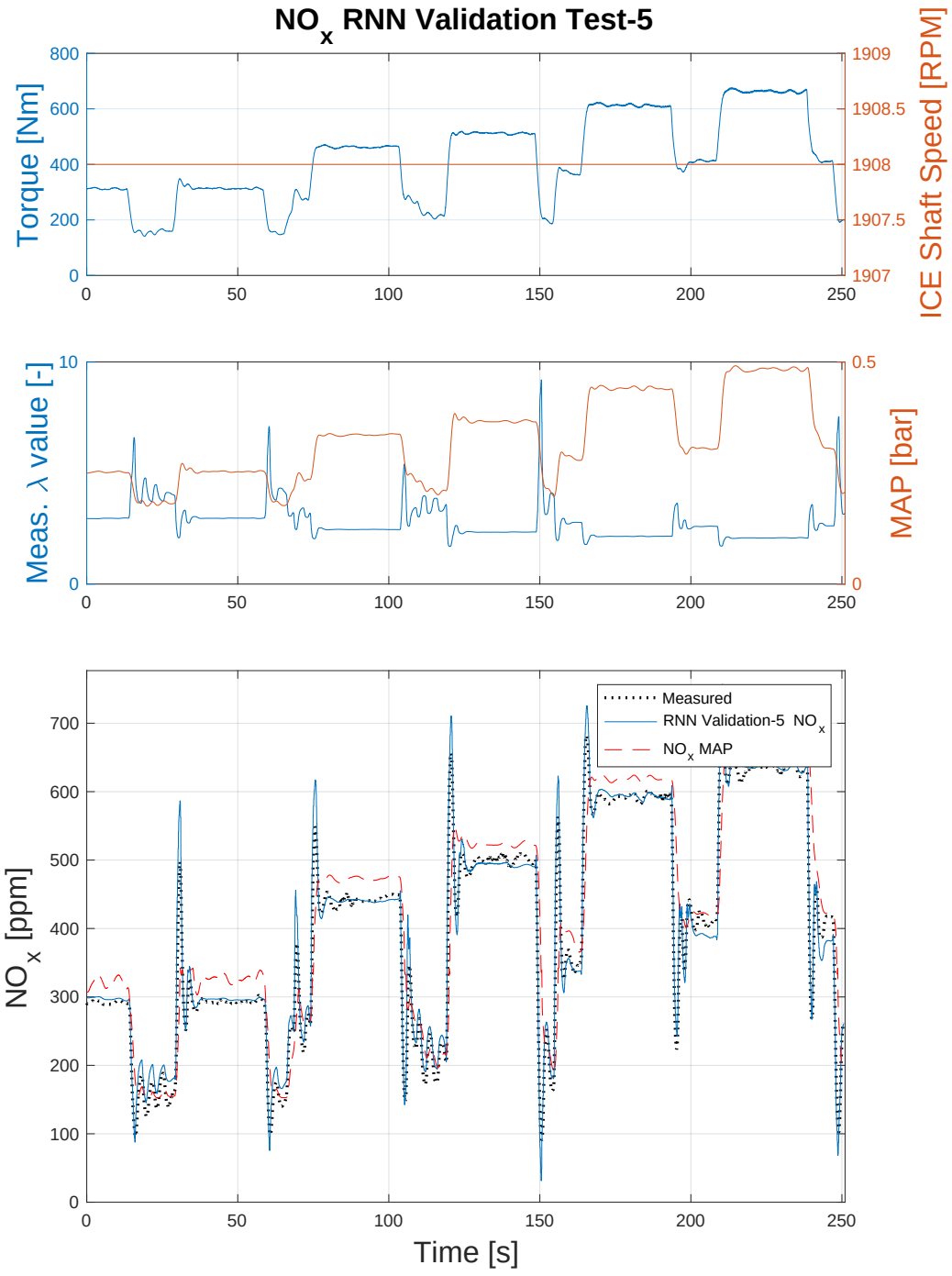

Figure 6.33: NO_x RNN results for validation test 5.

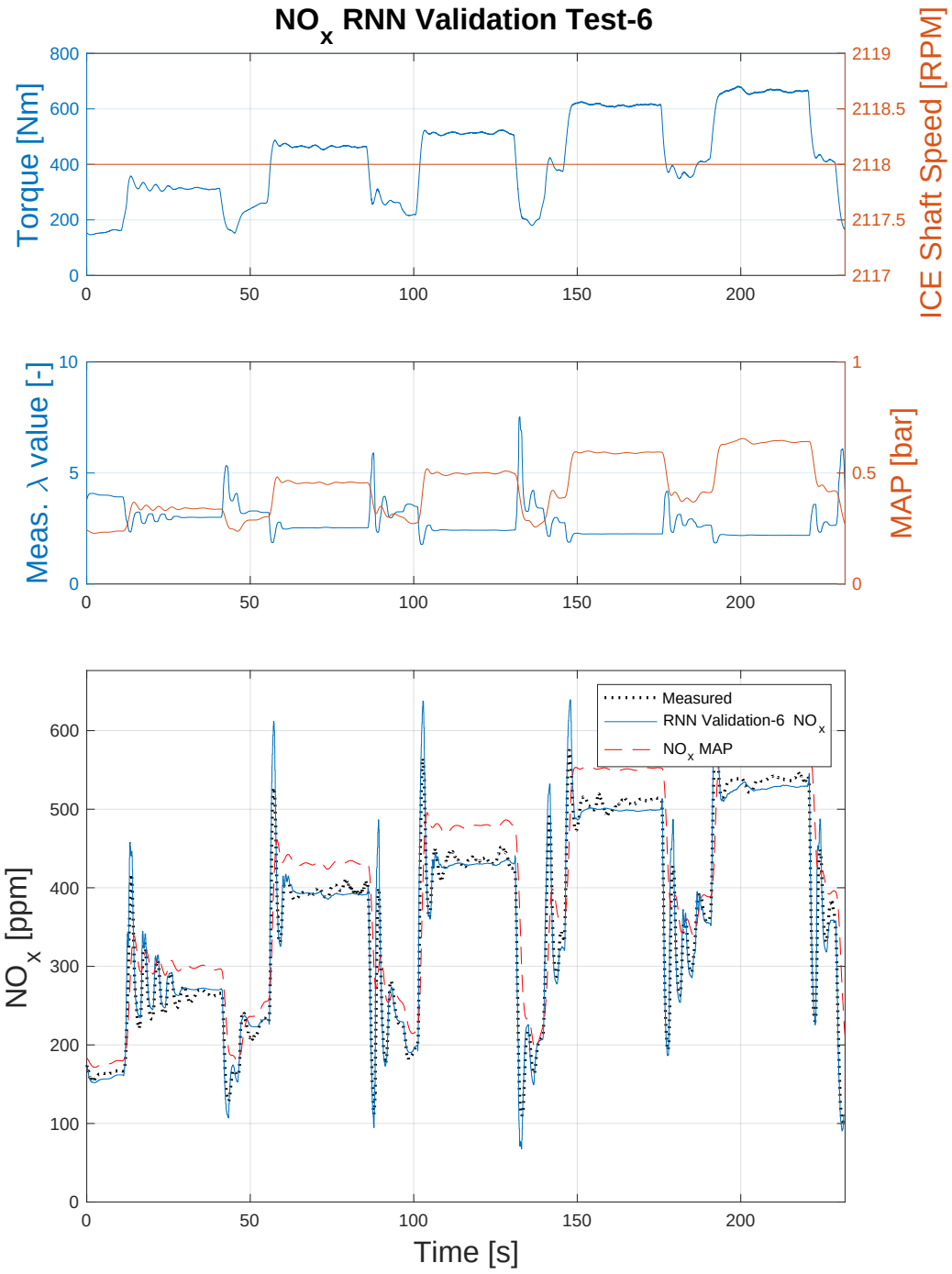

Figure 6.34: NO_x RNN results for validation test 6.

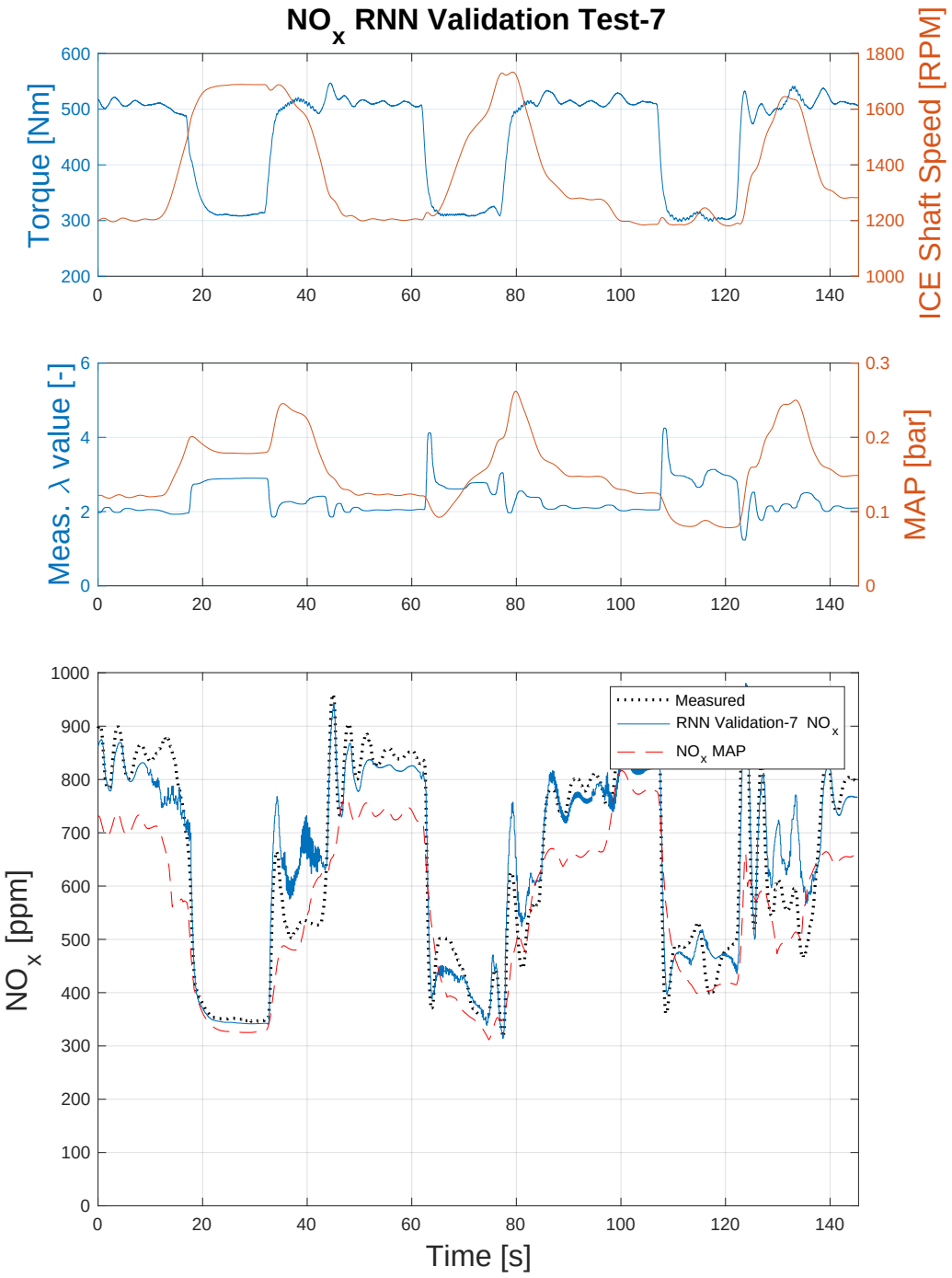

Figure 6.35: NO_x RNN results for validation test 7.

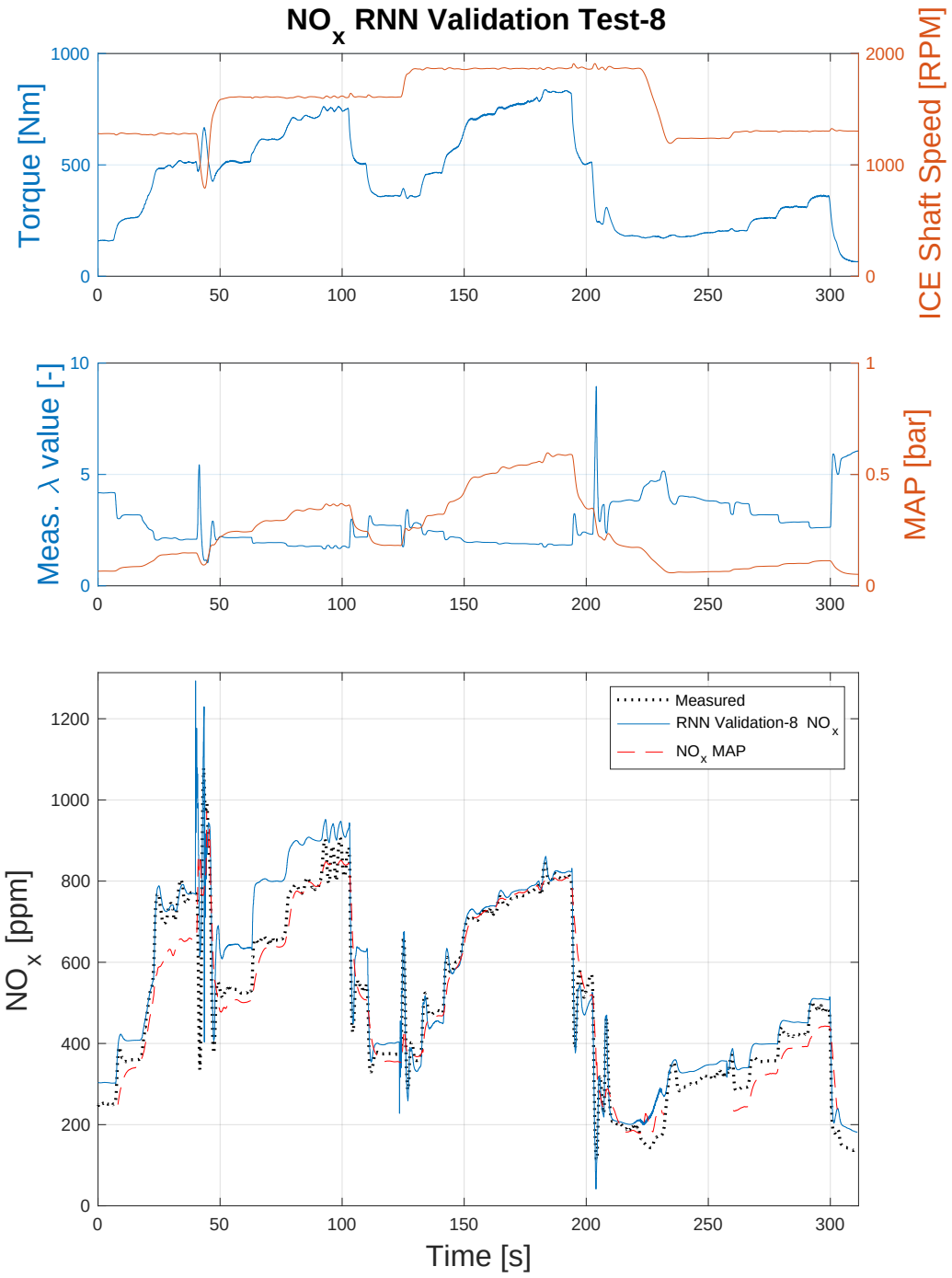

Figure 6.36: NO_x RNN results for validation test 8.

6.3 Validation Evaluation

Overall, the real-time validation proves that the RNN models can generalize in patterns similar but different than their training dataset. On the other hand, the TDNN models for both λ value and NO_x cannot keep up well with input stimulus different that those they were trained with, they are very sensitive to the static engine speed and torque input signals and fail to draw the dynamics from the other dynamic inputs, such as the intake manifold absolute pressure. Both TDNN and RNN models input ports rely on the pattern they were trained at and they do not demonstrate good interpolation capability between their input quantities. The difference between these two is that the RNN models results become a little noisy and follow at the same curve as the target values, while the TDNN models results are completely off.

At the same time the RNN models perform relatively better than existing map-based and first principles models for NO_x emissions and λ value predictions respectively. In some cases both the first principles λ value model and the RNN model are producing unstable results which is an indication that both models suffer from the input stimulation at that particular operating area. On the other hand, the map-based NO_x model is not following the measured NO_x trajectory as close as the RNN model and the same time it misses all the spikes as shown in Fig. 6.32 and transient phenomena due to its static nature.

Conclusively, as stated previously, the RNN models can be qualified to be used as virtual sensor instead of NO_x maps because their results are accurate and stay close with the measured NO_x emissions.

Chapter 7

Conclusions and Future Work

Conclusions

In this thesis, the feasibility of using artificial neutal networks for λ value and NO_x emissions predictions has been investigated. Initially, it was decided to use models that take into account the previous states of a system to calculate their predictions. That is why the recurrent neural network and the time-delay architectures were selected for the models. The models should take advantage of already installed sensors to draw their inputs, but the inputs selection has been decided based on first principles thermodynamics and the ome of the inputs correlation to output quantities. Several model architectures have been identified and evaluated. However, only one model is selected for each output variable and each architecture, resulting in four multiple-input-single-output (MISO) models. Each one of the four models has been evaluated based on its performance on the training data. The accuracy of the trained models was verified experimentally under realistic operating conditions on the hybrid diesel-electric testbed at LME. Experimental results showed accurate results at step load operation and during transients for the RNN models. The ANN models results are compared to existing map-based and first principles models and the comparison shows that the ANN predictions are better or equally accurate.

Conclusively, it can be derived that ANN models with calculations using previous states can successfully predict the NO_x and λ value emissions inside their training range for known load-speed patterns while some of them, i.e. the RNN models, can produce accurate results even in patterns that are uknown to them.

Suggestions for Future Work

In this work, ANN models are occupied for the NO_x and λ value approximation during real operating conditions.

Regarding the models accuracy, it can be suggested that the models are trained based not only at part-load operation data but also during transient data, i.e. a propeller load test or similar to a "driving" cycle. This could lead in more accurate models during transient operation. Additionally, the usage of internal state feedback for each neuron can be investigated as it was completely ignored in this thesis and it could lead to more stable results. Regarding the models usage, the models could be attached to MPC controllers or in other applications in order provide more accurate results for the testbed variables and therefore lead to more effective control strategy.

Using the same principles, models for new variables can be created, on top of NO_x and λ value, as the model creation methodology described in this thesis can be used for any of the LME engine variables that come as time-series, based upon reasonable inputs selection, e.g. prediction of fuel flow or exhaust gases opacity.

Appendices

Appendix A

Pyrenn module improvement

A.1 Levenberg-Marqaurdt Implementation modification

In this thesis, there were developed TDNN and RNN models. In order to create and train the models, Pyrenn module was used. It is a recurrent neural network toolbox for Python and MATLAB developed by Dennis Atabay from Technical University of Munich (TUM) [22]. It is an open source neural network builder which enables fast prototyping of wide variety of Recurrent Neural Networks using a set of 5 commands. The training algorithm is Levenberg-Marquardt (LM) [26], which converges much faster than gradient descent algorithms for time-series predictions due to its nature. The number of training epochs needed is lower than other applications might need, e.g. image recognition applications. The LM algorithm implementation for the initial Pyrenn toolbox was prone to freeze when it reached very small incremental improvements on the error minimization. That is why the author introduced a new input variable for the algorithm, i.e. the minimum error minimization step, which forces the training to stop if very small improvements are met, it has been submitted in GitHub stream and it is part of the main Pyrenn stream.

These changes solve an open issue with the module. Specifically the author forked the Pyrenn repository and applied changes to fix issue with tittle *neural network training 6* which was reported more than a year ago from the time of fix. The fix was submited as *Update Train LM with minimum error step criterium* [27] and was accepted in the main stream of the module for every user to utilize it. The code changed is the LM function shown in A.1, while the rest of the code can be found in GitHub[1].

```
.
.
early=0 #early stopping counter
while True:
 #run loop until either k_max or E_stop is reached
  JJ = np.dot(J.transpose(),J) #J.transp * J
  w = net['w'] #weight vector
  while True:
   #repeat until optimizing step is successful
   #calculate gradient
   g = np.dot(J.transpose(),e)

     #calculate scaled inverse hessian
     try:
```

[1] https://github.com/yabata/pyrenn/blob/master/python/pyrenn.py

```
        G = np.linalg.inv(JJ+dampfac*np.eye(net['N']))
      except np.linalg.LinAlgError:
        # Go small step in gradient direction
        w_delta = 1.0/1e10 * g
      else:
        # calculate weight modification
        w_delta = np.dot(-G,g)

        net['w'] = w + w_delta #new weight vector

        Enew = calc_error(net,data)

        if Enew<E and abs(E-Enew)>=min_E_step:
                    #Optimization Step successful!
                    dampfac= dampfac/dampconst
                    early=0 #reset the early stopping counter
                    break #go to next iteration
        else:
                    #Optimization Step NOT successful!
                    dampfac = dampfac*dampconst
                    if abs(E-Enew)<=min_E_step:
                     early=early+1 #increase early stopping counter
                     if verbose:
                            print('E-Enew<=min_E_step Encountered!!')
                            if early>=5.0:
                             print('5 Times * E-Enew<=min_E_step Encountered!!')
                     break

    #Calculate Jacobian and Error
    J,E,e = RTRL(net,data)
    k = k+1
    ErrorHistory[k] = E
    if verbose:
     print('Iteration:',k,'Error:',E,'scale factor:',dampfac)

                    #Ceck if termination condition is fulfilled
    if k>=k_max:
     print('Maximum number of iterations reached')
                        break
    elif E<=E_stop:
     print('Termination Error reached')
     break
.
.
```

As a result, if a training of a model is stuck in very small incremental changes of the error, now the algorithm breaks the loop via the early stopping criteria that was added. This change is helpful for this thesis because there were tested a lot of model architectures - in order to decide the optimal architecture - and often was the case where the automated model creation and training script was stuck without obvious reason.

Bibliography

[1] Heywood J.B.: 'Internal Combustion Engine Fundamentals', (McGraw-Hill, New York, 1988)

[2] Hinton, G., Osindero, S. and Teh, Y.: 'A Fast Learning Algorithm for Deep Belief Nets.' *Neural Computation, 18(7), pp.1527-1554*, 2006

[3] Gambarotta, A., Luchetti, G., Fiorani, P., de Cesare, M. et al.: 'A thermodynamic Mean Value Model of the intake and exhaust system of a turbocharged engine for HiL/SiL applications', *SAE Technical Paper 2009-24-0121*, 2009, https://doi.org/10.4271/2009-24-0121.

[4] Hagena, J., Filipi, Z., and Assanis, D.: 'Transient Diesel Emissions: Analysis of Engine Operation During a Tip-In' *SAE Technical Paper 2006-01-1151*, 2006, https://doi.org/10.4271/2006-01-1151

[5] Papalambrou, Georgios Samokhin, Sergey Topaloglou, Sotirios Planakis, Nikolaos Kyrtatos, Nikolaos Zenger, Kai.: 'Model Predictive Control for Hybrid Diesel-Electric Marine Propulsion' *20th IFAC World Congress*, 2017, **50**, pp. 11064-11069. https://doi.org/10.1016/j.ifacol.2017.08.2488.

[6] Li, H., Butts, K., Zaseck, K., Liao-McPherson, D. et al.: 'Emissions Modeling of a Light-Duty Diesel Engine for Model- Based Control Design Using Multi-Layer Perceptron Neural Networks', *SAE Technical Paper 2017-01-0601*, 2017. https://doi.org/10.4271/2017-01-0601.

[7] Ivan Arsie, Dario Marra, Cesare Pianese, Marco Sorrentino: 'Real-Time Estimation of Engine NOx Emissions via Recurrent Neural Networks', *6th IFAC Symposium Advances in Automotive Control Munich*, Germany Munich, July, 2010 https://doi.org/10.3182/20100712-3-DE-2013.00117

[8] Kamat, S., Diwanji, V., Smith, J., Javaherian, H. et al., 'Virtual Sensing of SI Engines Using Recurrent Neural Networks', SAE Technical Paper 2006-01-1348, 2006, https://doi.org/10.4271/2006-01-1348.

[9] Arsie, I., Pianese, C., and Sorrentino, M., 'Development of recurrent neural networks for virtual sensing of NOx emissions in internal combustion engines', SAE Int. J. Fuels Lubr. 2(2):354-361, 2010, https://doi.org/10.4271/2009-24-0110.

[10] Amin, S., Gerhart, V., and Rodin, E., "System Identification via Artificial Neural Networks: Applications to On-line Aircraft Parameter Estimation*," SAE Technical Paper 975612, 1997, https://doi.org/10.4271/975612.

[11] Wang, H., Chi, J., Wu, C., Yu, X. et al., "A Multi-Objective Recognition Algorithm with Time and Space Fusion," SAE Technical Paper 2019-01-1047, 2019, https://doi.org/10.4271/2019-01-1047.

[12] Arsie, I., Cricchio, A., De Cesare, M., Pianese, C. et al., "A Methodology to Enhance Design and On-Board Application of Neural Network Models for Virtual Sensing of Nox Emissions in Automotive Diesel Engines," *SAE Technical Paper 2013-24-0138, 2013*, https://doi.org/10.4271/2013-24-0138.

[13] De Cesare, Matteo Covassin, Federico: 'Neural Network Based Models for Virtual NOx Sensing of Compression Ignition Engines', *SAE International*, USA, September 2011. https://doi.org/10.4271/2011-24-0157

[14] Ivan Arsie, Anrea Cricchio, Matteo De Cesare, Franceso Lazzarini, Cesare Pianese, Marco Sorentino: 'Neural network models for virtual sensing of NOx emissions in automotive diesel engines with least square-based adaptation', *Control Engineering Practice*, April 2017 ,**61** , pp 11-20. https://doi.org/10.1016/j.conengprac.2017.01.005.

[15] Spara, H., Milink, G., Klass, V., Douwe, S., Dijktsra,C.: s'Experimental and simulation-based investigations of marine dieselengine performance against static back pressure', *Applied Energy 204 pp78-92*, Delft University of Technology, The Netherlands, 2017. http://dx.doi.org/10.1016/j.apenergy.2017.06.111

[16] Arsie, I., Di Iorio, S., Pianese, C., Rizzo, G. and Sorrentino, M.: 'Recurrent Neural Networks for air-fuel ratio estimation and control in SI Engines', I*FAC Proceedings Volumes, 41(2), pp.8508-8513.*, 2008

[17] Orr, G.: 'Neural networks.', *Springer, pp.9-50*, Berlin, 1998.

[18] Patterson, D. W.: 'Artificial Neural Networks – Theory and Applications', *Prentice Hall.*, 1995

[19] Chaichan, Miqdam Abass, Qahtan. (2010). Study of NOx Emissions of S.I. Engine Fueled with Different Kinds of Hydrocarbon Fuels and Hydrogen. Al-Khwarizmi Eng J. 6. 11-20.

[20] Rosenblatt, F.: 'The perceptron: A probabilistic model for information storage and organization in the brain', *Psychological Review, 65(6), pp.386-408*, 1958

[21] Papalambrou, G. Samokhin, S. Topaloglou, S. Planakis, N. Kyrtatos, N. Zenger, K.: 'Model Predictive Control for Hybrid Diesel-Electric Marine Propulsion.', IFAC-PapersOnLine. 50. 11064-11069, 2017. 10.1016/j.ifacol.2017.08.2488.

[22] Dennis Atabay: 'Pyrenn: A recurrent neural network toolbox for python and matlab', *Institute for Energy Economy and Application Technology*, Technische Universität München.

[23] Hornik, K.: 'Approximation capabilities of multi-layer feed forward networks.' *Neural Networks, 4(2), pp.251-257.* 1991

[24] Géron, A.: 'Hands-on machine learning with Scikit-Learn and TensorFlow.' *1st ed. O'Reily.*, 2017.

[25] Orr, G. and Muller, K.:'Neural Networks: tricks of the trade', Springer, 1998

[26] Williams, R., Zipser, D.: 'A Learning Algorithm for Continually Running Fully Recurrent Neural Networks.' *Neural Computation, Nummer 2, Vol. 1,S. 270-280* , 1989

[27] Dimitrakopoulos, P.: 'Update Train LM with minimum error step criterium 8', 2019. https://github.com/yabata/pyrenn/pull/8/files.

[28] S. Topaloglou, G. Papalambrou, N. Kyrtatos. 2016. Energy Management Controller Design for Hybrid Ship Propulsion During Transient Operation. *CIMAC congress, Helsinki, 6-10 June 2016, paper No. 50.*

[29] L. Guzzella, C. Onder. 2004. Introduction to Modeling and Control of Internal Combustion Engine Systems. *Springer, 2nd edition.*