Politecnico di Torino

Department of Control and Computer Engineering
(DAUIN)

Master of Science in
Mechatronic Engineering

Modelling and Identification of a Dual Clutch Actuator

Supervisors:
Prof. Diego Regruto
Prof. Vito Cerone
Prof. Massimo Canale

Author:
Talal Almutaz Almansi Abdalla

April 2018

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

قَالُوا۟ سُبْحَٰنَكَ لَا عِلْمَ لَنَآ إِلَّا مَا عَلَّمْتَنَآ ۖ إِنَّكَ أَنتَ ٱلْعَلِيمُ ٱلْحَكِيمُ ﴿٣٢﴾

"They said, "Exalted are You; we have no knowledge except what You have taught us. Indeed, it is You who is the Knowing, the Wise.""

-Quran, 1:32

Acknowledgments

First, I would like to thank my beloved parents, brothers and my lovely sister, for always being caring, understanding and supporting me throughout my life.

I want to express my love and affection to my wife Mayada, having always been by my side, providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

I am deeply grateful to my thesis supervisor Prof. Diego Regruto. In spite of his great responsibilities, he managed to spare some of his valuable time to advise and guide me through out this work. I could not have imagined having a better supervisor for my thesis.

I would also like to express my sincere gratitude to Prof. Vito Cerone and Prof. Massimo Canale for the help and guidance I received from them throughout this journey. Without their passionate participation and input, this work could not have been successfully completed.

My personal and deep appreciation goes to Dr. Valentino Razza, my colleague Ahmed Elhaddad, the Centro Ricerche Fiat (CRF) team represented in Eng. Emanuel Corigliano, Eng. Simone Baliva and Eng. Sabrina Spagnolo.

Abstract

In this thesis we introduce the problem of identifying a model for a dual clutch actuator from experimentally collected data to be used to control the position of the clutch. However, since the measurement of the clutch position is not present in the real application, a *Virtual Sensor* has been implemented to provide this measurement.

The first stage of this thesis was to identify a model to produce the clutch actuator's output pressure from the input current. *Set-Membership* (SM) identification techniques were applied on the data sets to get a model able to cope with the uncertainties affecting the measurements. The data sets were examined and a dead zone was found present, this has motivated the need to use a Single-Input Single-Output (SISO) Linear Time Invariant (LTI) *Hammerstein* model structure. The collected experimental data has two signal profiles for the input current, ramp and step, and were collected at 3 different operating temperatures $-20°C$, $20°C$ & $60°C$. Applying the data sets with ramp profile on the identified model produced relatively good results.

Applying the data sets with step profile, the model was not able to cope up with the plant's dynamics that are changing in a rapid and unpredictable manner. This motivated the need to use *Recursive Estimation* for this identification problem. The *Recursive Estimation* algorithm was able to obtain good results in the sense that the absolute error between the measured and the estimated pressure (through the *Normalized Gradient* algorithm), was found to be bounded by $|2 \times 10^{-14}|$.

In the second stage of this thesis, the *Virtual Sensor* was implemented using two different structures. The first structure was a *Feedforward Neural-Network* trained for a model in the form of a *Moving-Average Model* (FIR) for each operating temperature. The other structure was a finite and known order polynomial modelled with *Curve Fitting* techniques.

Using the experimental data of the current and pressure as inputs and the position as output, the modelled virtual sensor was able to track the position measurements with glitches. The glitch phenomena was eliminated by introducing a *Median Filter* on the output of the *Virtual Sensor*.

The second phase of the *Virtual Sensor* modelling was motivated by the curve shape between the pressure and the position. The *Virtual Sensor* was modelled with both, a *Feedforward Neural-Network* and a *Curve Fitting Polynomial*, where both these structures had similar performances in terms of the *Mean Squared Error* (MSE).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Transmission Systems and DDTC

The transmission system is the component responsible for transferring the engine's generated power to the wheels of the vehicle and enhances the power flow through the required gear ratios. Currently, there are five main transmissions systems deployed to vehicles, those are Manual Transmission (MT), Automatic Transmission (AT), Automated Manual Transmission (AMT), Dual Clutch Transmission (DCT) and Continuously Variable Transmission (CVT). We will now go into details talking about the DCT.

Dual clutch transmission is an automated manual transmission with two clutches, one for even gears and the other for odd gears. The advantage of using two separate clutches is to avoid the traction interruption when shifting from one gear to the other. This transmission allows near instant shifting between gears.

There are two different types of clutches in the DCT transmission system, wet (WDCT) and dry (DDCT). Wet clutches are used for high torque engines that can generate 350 Nm, while the dry clutches can reach up to a maximum of 250 Nm and is suitable for small vehicles. The dry clutch has an advantage over the wet one as it offers an increase in fuel efficiency.

In this thesis, we will be working on the on the Dual Dry Clutch Transmission developed by Fiat Powertrain Technologies [1]. This transmission is part of the C635 new family. This new transmission family consists of a range of manual, all wheel drive and DCT transversal, 6-speed transmissions with a maximum input torque of 350 Nm and output torque of 4200 Nm.

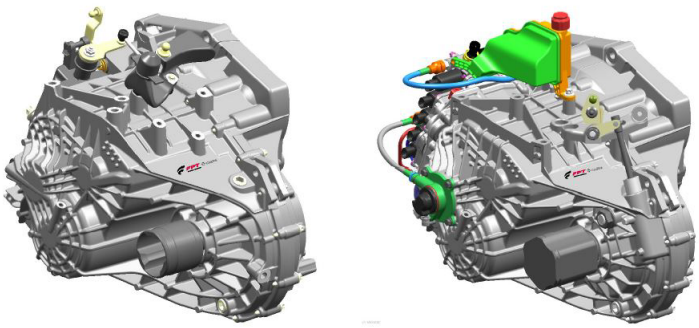


Figure 1.1: C635 MT and DDCT Versions

1.1.1 C635 DDCT Transmission

As shown in figure (1.2), the 3-shaft transmission architecture is contained in a two piece aluminum housing with an intermediate support plate for the shaft bearings. The gear set housing is characterized by reduced upper secondary shaft length, a feature which was also necessary in order to ensure packaging in the lower segment vehicles where the longitudinal crash beam imposes serious installation constrains.

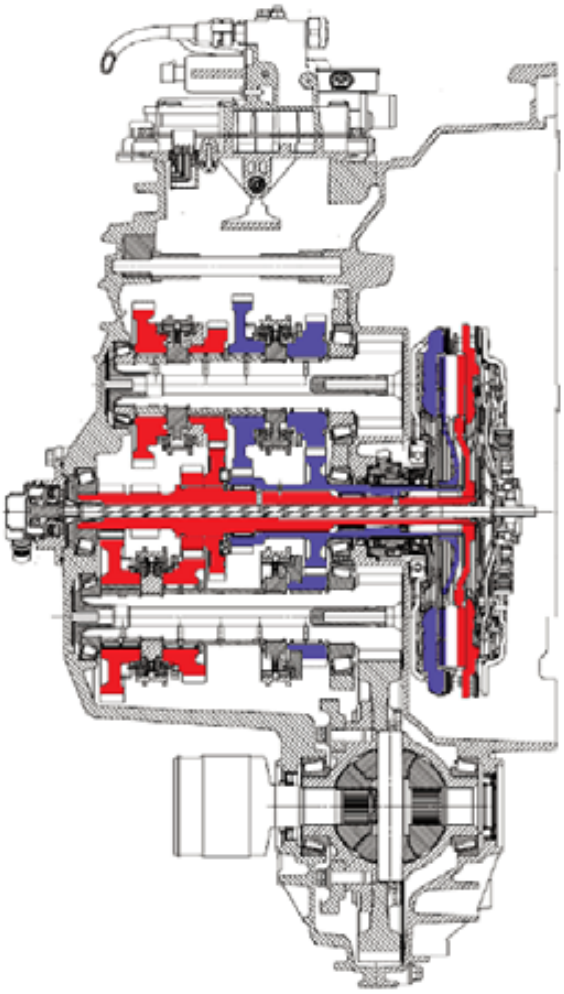


Figure 1.2: Cross Section of the C635 DDCT

The most important feature of this transmission in terms of packaging characteristics is the adoption of a coaxial pull-rod for the actuation of the odd-gear clutch (K1), while the even gear clutch (K2) is actuated with a rather conventional hydraulic Concentric Slave Cylinder (CSC). This pull-rod is connected to a hydraulic piston actuator located on the rear face of the transmission housing in a manner identical to the one adopted in the past in an earlier FPT technical demonstrator.

1.1.2 Dry Dual Clutch Unit

In order to motivate the model decomposition into the different subsystems it is important to describe first the main operations and elements of a DCT. During the shifting process the transmitted torque is obtained by the overlap of the engagement of the closing clutch and the release of the opening clutch. The clutch K1 is normally closed as in conventional manual transmissions and its position is controlled by means of a contact-less linear position sensor integrated in the rear hydraulic piston actuator. The even gear clutch K2 is normally open and is controlled in force, i.e., through hydraulic pressure. The two clutches act on a center plate together with the two pressure plates. The entire dual clutch unit is mounted on the clutch housing by means of a single main support bearing. This compact mounting solution was developed thanks to the adoption of the specific actuation system of the clutch K1 which allows the space for such a bearing to be installed [2].

1.1.3 Electro-Hydraulic Actuation System

The hydraulic circuit of the actuation system is represented in figure (1.3), it consists of 4 double action pistons, 5 non-contact linear position sensors (one for each shifting piston and one for the shifter spool), 2 speed sensor for the two primary shafts, 5 solenoid valves (4 pressure proportional (PPV), 1 flow proportional (QPV)). The QPV is used for the position control of clutch K1.

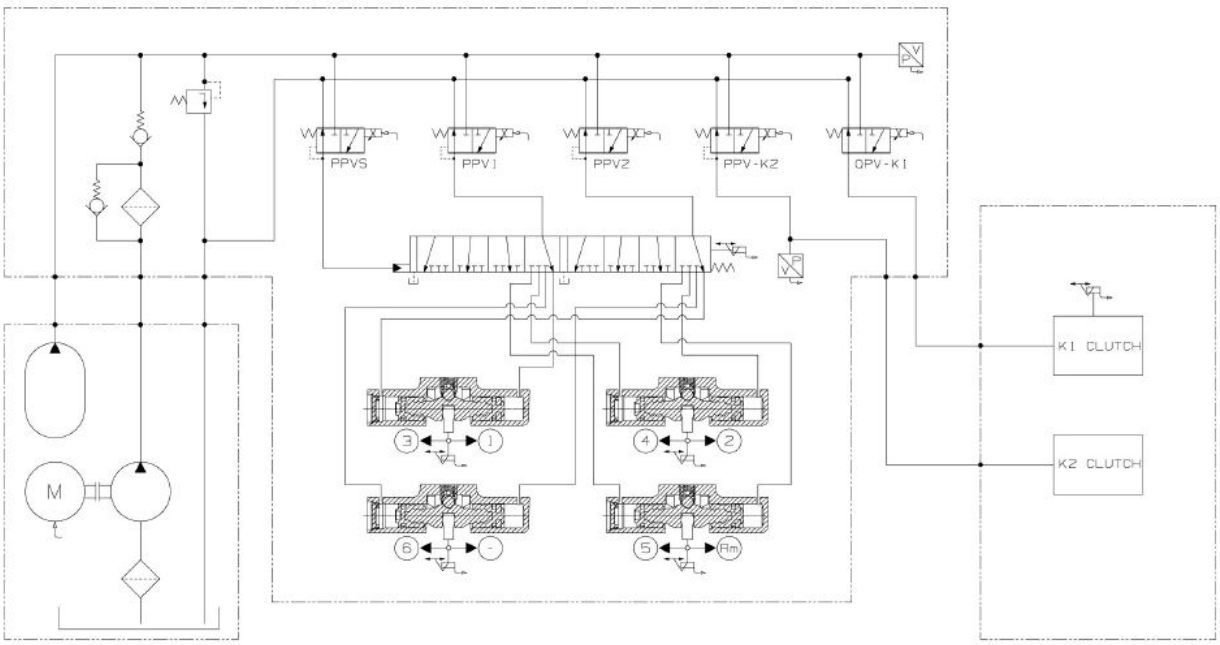


Figure 1.3: Complete Actuation System (CAS) Hydraulic Circuit

The system is composed of a hydraulic power unit (PU, Figure 1.4, left picture), consisting of an electrically driven high pressure pump and accumulator, and an Actuation Module (CAM, Figure 1.4, right picture) which includes the control solenoid valves, the gearshift actuators and the sensors.

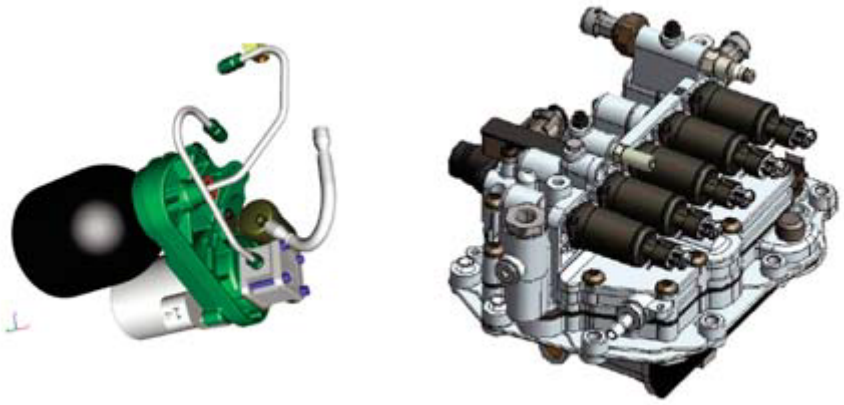


Figure 1.4: The Hydraulic Power Unit (PU) and The Complete Actuation Module (CAM)

1.1.4 C635 DDCT Control Unit

The C635 DDCT has been designed as a standalone unit. The control strategies run in a multitasking environment, preserving the resources of the Main Microcontroller, these strategies can be grouped as:

- Actuator Control:
  - Engagement Actuators Control.
  - Shifter (Selector) Control.
  - Odd Gears Clutch (K1) Control: The normally closed clutch (K1) is controlled by a position closed loop. This is the clutch of the first and of the reverse gear,therefore this control strategy is essential also for the vehicle starting performance.
  - Even Gears Clutch (K2) Control: The normally open clutch (K2) is controlled in force with a pressure feedback signal delivered by one of the CAM sensors.
- Self-Tuning Control: The main self-tuning control algorithms concern the conversion of the requested clutch transmitted torque to K1 position and K2 pressure.
- Launch and Gear Shift Strategies: The C635 DDCT implements various driving modes, depending on the desired performance and Brand/OEM requirements, both in manual and in automatic mode. In effect, three different modes of shift patterns

in automatic and two different ones in manual (tip) mode are contemplated and are accomplished also by specific control strategies and calibrations on the engine side. Vehicle creeping on brake release is also implemented, in collaboration with the braking systems hill holding functions.

1.2 Objective

The objective of this thesis can be addressed in the following two points:

- System Identification for the K2 Actuator: Finding a suitable model for the K2 actuator to be used to control the clutch position.
- Virtual Sensor Modelling: To design a Virtual Sensor able to provide the K2 clutch position in real-time operation.

1.3 Thesis Organization

In this chapter we gave a brief overview of vehicle transmission systems and the Dual Dry Clutch Transmission (DDCT) developed by Fiat Powertrain Technologies.

In Chapter 2 we introduce the theory behind the *Set-Membership* framework (SM) used in system identification of *Linear Time-Invariant* (LTI) systems. This framework will be used to identify a model for the K2 actuator plant to estimate the system's output pressure.

Chapter 3 introduces the system identification process followed to model the K2 actuator. First we start by identifying a model in the *Set-Membership* framework able to provide reasonably good estimation for some sets of the experimental data, but fails to cope up with the system's varying dynamics for the other sets of the experimental data. The latter mentioned point was the main motivation to move to a recursive system identification approach. The recursive approach proved to be successful for this application as demonstrated by the achieved results.

Chapter 4 highlights the main theory and concepts behind *Neural Networks*. The chapter briefly reviews the structure of a *Neural Network*, its activation functions and the learning processes.

In chapter 5 we apply the concepts introduced in chapter 4 to model a *Virtual Sensor* able to provide a real-time estimate of the clutch position given the current and pressure measurements. We use the *Multilayer Feedforward Neural Network* to model three sensors, corresponding to three different operating condition of the clucth, characterized by three different operating temperatures ($-20°C$, $20°C$ & $60°C$).

The last chapter, chapter 6, gives a summary about the work completed in this thesis and the proposed future work.

1.4 Non-Disclosure Agreement

As per the Non-Disclosure Agreement signed between **Fiat Chrysler Automobiles** (FCA) and me the author of this thesis, **Talal Almutaz Almansi Abdalla**, the measurement data sets used in all the modelling processes presented in this thesis have been normalized between 0 and 1.

Chapter 2

Set-Membership

2.1 Overview of System Identification Frameworks

The most popular system identification approaches are based on the assumption that the measurement uncertainties entering the identification problem are random values with a known probability density function (for more details, see [3, 4, 5]).

Throughout the years, many powerful estimation methods have been developed, as an example, Least Squares, Gauss-Markov, Maximum Likelihood and Bayesian estimators. Even though the previously mentioned methods are used in a wide range of applications, they suffer from a major drawback. These method need a large enough number of collected input-output data samples to in order check a posteriori the correctness of the statistical hypotheses and to evaluate reliable confidence intervals for the estimated parameters. It is important to note that it is not always possible to observe a large number of samples, such as in biological systems. This drawback is the motivation for the need to introduce the Set-Membership estimation theory.

2.2 Set-Membership Framework

Set-Membership (SM) Estimation, also known as *Bounded-Error Estimation*, is an alternative stochastic estimation theory, where the noise entering the identification problem is assumed to be unknown but bounded.

2.2.1 Problem Formulation in SM Framework

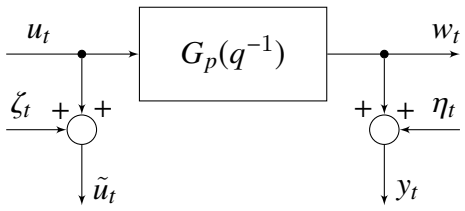


Figure 2.1: Errors-in-Variables Model Structure

We can look at the problem formulation of systems described by the following Single-Input Single-Output (SISO) Linear Time-Invariant (LTI) model in the form of an Errors-

in-Variables (EIV) structure as in figure (2.1). The noise-free output w_t at time t is given by,

$$w_t = G_p(q^{-1})u_t \tag{2.1}$$

where u_t is the noise-free input at time t and $G_p(q^{-1})$ is a discrete-time in the backward shift operator q^{-1}, which transforms u_t into w_t,

$$G_p(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})} \tag{2.2}$$

$A(\cdot)$ and $B(\cdot)$ are polynomials in the backward shift operator q^{-1},

$$A(q^{-1}) = 1 + \alpha_1 q^{-1} + ... + \alpha_{n_a} q^{-n_a} \tag{2.3}$$

$$B(q^{-1}) = \beta_0 + \beta_1 q^{-1} + ... + \beta_{n_b} q^{-n_b} \tag{2.4}$$

$$n_a \geq n_b$$

where $n_a \geq n_b$
Equation (2.2) can be rewritten as,

$$G_p(q^{-1}) = \frac{\beta_0 + \beta_1 q^{-1} + ... + \beta_{n_b} q^{-n_b}}{1 + \alpha_1 q^{-1} + ... + \alpha_{n_a} q^{-n_a}} \tag{2.5}$$

The noise corrupted input $\tilde{u_t}$ and output y_t are given by,

$$\tilde{u_t} = u_t + \zeta_t \tag{2.6}$$

$$y_t = w_t + \eta_t \tag{2.7}$$

where:

ζ_t is the input measurement uncertainty at time t,

η_t is the output measurement uncertainty at time t.

As stated in the beginning of this section, the noise entering the problem is assumed to be unknown but bounded,

$$|\zeta_t| \leq \Delta_\zeta \tag{2.8}$$

$$|\eta_t| \leq \Delta_\eta \tag{2.9}$$

Now we can define the Feasible Parameter Set (FPS), the Extended Feasible Parameter Set and the Parameter Uncertainty Intervals (PUI) for this modelling problem.

2.2.1.1 Feasible Parameter Set

Definition: The Feasible Parameter Set is the set of all the possible parameter values which are consistent with the equation describing the assumed model structure, noise structure and noise bound.

From equations (2.1) to (2.9), the Feasible Parameter Set can be formulated as follows,

$$\begin{aligned}\mathcal{D}_\theta = \{\theta \in \mathbb{R}^{n_a+n_b+1} : w_t = -\alpha_1 w_{t-1} - \alpha_2 w_{t-2} - ... - \alpha_{n_a} w_{t-n_a} + \beta_0 u_t + \beta_1 u_{t-1} + \\ +... + \beta_{n_b} u_{t-n_b}; y_t = w_t + \eta_t; \tilde{u}_t = u_t + \zeta_t; |\eta_t| \leq \Delta_\eta; |\zeta_t| \leq \Delta_\zeta; \forall t = n_a + 1, ..., N\}\end{aligned} \tag{2.10}$$

where N is the number of input-output data samples.

$$\theta^T = [\alpha_1, \alpha_2, ..., \alpha_{n_a}, \beta_0, \beta_1, ..., \beta_{n_b}] \tag{2.11}$$

2.2.1.2 Extended Feasible Parameter Set

As it can be seen in the FPS represented in equation (2.10), the noise entering the problem (both ζ_t and η_t) are unknown variables as well, hence they have to be included in the FPS. Now we will introduce the Extended FPS to include these unknown variables,

$$\begin{aligned}\mathcal{D}_{\theta,\zeta,\eta} = \{\theta \in \mathbb{R}^{n_a+n_b+1}, \zeta \in \mathbb{R}^N, \eta \in \mathbb{R}^N : y_t - \eta_t = -\alpha_1(y_{t-1} - \eta_{t-1}) - ... - \alpha_{n_a}(y_{t-n_a} - \eta_{t-n_a}) + \\ +\beta_0(\tilde{u}_t - \zeta_t) + ... + \beta_{n_b}(\tilde{u}_{t-n_b} - \zeta_{t-n_b}); |\eta_t| \leq \Delta_\eta; |\zeta_t| \leq \Delta_\zeta; \forall t = n_a + 1, ..., N\}\end{aligned} \tag{2.12}$$

$$\eta^T = [\eta_1, \eta_2, ..., \eta_N] \tag{2.13}$$

$$\zeta^T = [\zeta_1, \zeta_2, ..., \zeta_N] \tag{2.14}$$

2.2.1.3 Parameter Uncertainty Intervals

Definition: The Parameter Uncertainty Interval (PUI) in the Set-Membership Framework, is defined as the region which bounds a parameter's value by its maximum and minimum intervals.

For a parameter $\theta_i \in \theta$, the PUI can be defined as follows,

$$PUI_{\theta_i} = [\theta_i^{min} \quad , \quad \theta_i^{max}] \tag{2.15}$$

$$\theta_i^{min} = \min_{\mathcal{D}_\theta} \theta_i \quad , \quad i = 1, 2, 3, ..., n_a + n_b + 1 \tag{2.16}$$

$$\theta_i^{max} = \max_{\mathcal{D}_\theta} \theta_i \quad , \quad i = 1, 2, 3, ..., n_a + n_b + 1 \tag{2.17}$$

The problem described in (2.12) is a Polynomial Optimization Problem (POP) and is a nonconvex problem.

Convex Relaxation techniques are applied to avoid the optimization problem to get trapped in a local minima and to assure the parameter's convergence to the global minima (see [6] for more details).

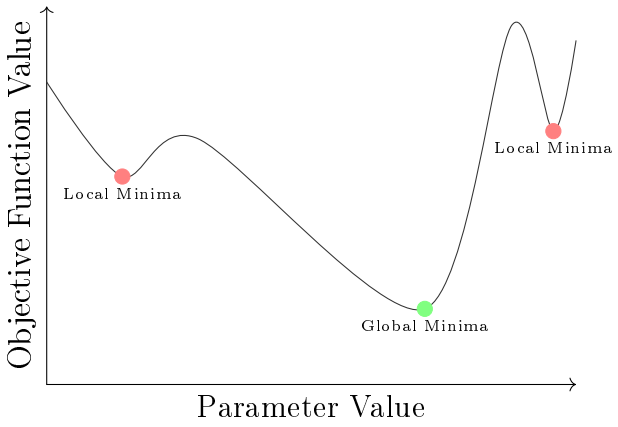


Figure 2.2: Global and Local Minima

Chapter 3

System Identification

3.1 Introduction

In this chapter, we introduce the system identification procedure followed in order to identify a model for the K2 actuator. The data sets use in this identification process have been provided by Centro Ricerche Fiat (CRF) and are divided into 2 sets of data:

- Set (1): Ramp profile data, where the input current is a ramp input.
- Set (2): Step profile data, where the input current is a step input.

The Set (1) was the first set available and was used to perform the identification process mentioned in section (3.2). The other data set, Set (2), was used to validate the obtained model following the identification process on Set (1) and hence led to further investigations and work on the plant's identification procedure as presented in section (3.3).

The K2 actuator model to be identified in this process is considered as a black-box, as no apriori information in regards to the model structure is available (for more details, see [7]). This model has as input the current and the pressure as output.

The data sets were examined and there was found to be a dead zone for the output pressure in presence of the input current.

The sampling time for the samples collected within the data sets was not the same between two consecutive measurements, hence as part of the preliminary processes the sampling time for all data sets was unified and set to 2 ms.

In the following sections, two identification procedures are introduced:

- Section 3.2: LTI Discrete time Model Identification with a static input nonlinearity in the form of a Hammerstein system.
- Section 3.3: Recursive Model Estimation.

3.2 Hammerstein System

3.2.1 Overview

In this section, we address the identification of a single-input single-output (SISO) linear time invariant (LTI) discrete-time system with a static input nonlinearity in the

form of a Hammerstein Model.

The key points for the identification process which were obtained on the basis of data analysis and preliminary processing are:

- The identification problem is in the form of an Output Error structure.
- The linear dynamic model $G_p(q^{-1})$ is a first order transfer function with one pole α and gain β.
- The static input nonlinearity $N(\cdot)$ is a dead zone and is modeled with a finite and known order polynomial and is considered.
- This model structure has been selected after several trails, where it has proven to be the most suitable structure for this identification problem.
- The noise η_t entering the problem and affecting the output is assumed to be bounded by Δ_η.
- The sampling time T_s is equal to 2 ms.

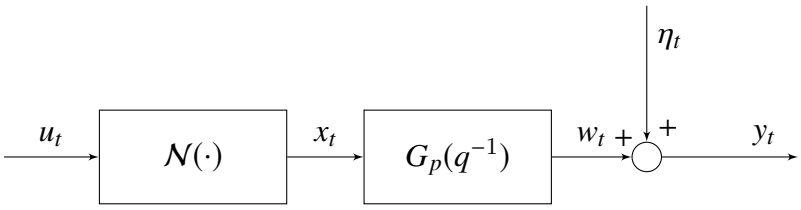


Figure 3.1: SISO Hammerstein Model

3.2.2 Problem Formulation

In reference to the Hammerstein system shown in figure (3.1) and the procedure mentioned in 2.2.1, where the linear dynamic part is modeled by a discrete-time system which transforms x_t into the noise-free output w_t according to:

$$w_t = G_p(q^{-1})x_t = \frac{B(q^{-1})}{A(q^{-1})}x_t \tag{3.1}$$

where x_t is the unmeasurable inner signal and $A(\cdot)$ and $B(\cdot)$ are polynomials in the backward shift operator q^{-1},

$$A(q^{-1}) = 1 + \alpha q^{-1} \tag{3.2}$$

$$B(q^{-1}) = \beta \tag{3.3}$$

The Hammerstein system input u_t is scaled to v_t according to:

$$v_t = \mathcal{K}u_t \tag{3.4}$$

where $\mathcal{K} = 0.01$ is the scaling gain.
This scaling is necessary in order to obtain a feasible solution for the optimization problem.
The nonlinear block transforms the scaled input v_t into x_t according to:

$$x_t = \gamma_1 v_t^5 + \gamma_2 v_t^4 + \gamma_3 v_t^3 \tag{3.5}$$

The selection for the structure of x_t as a polynomial with the terms shown in (3.5) was the best structure found to suitably model the dead zone.
Let y_t be the noise-corrupted measurements of w_t,

$$y_t = w_t + \eta_t \tag{3.6}$$

where η_t is the noise entering the problem,

$$\eta^T = [\eta_1 \quad \eta_2 \quad \dots \quad \eta_N] \tag{3.7}$$

Measurement uncertainty is known to be bounded by Δ_η,

$$|\eta_t| \leq \Delta_\eta \tag{3.8}$$

Unknown parameter vectors $\gamma \in \mathbb{R}^3$ and $\theta \in \mathbb{R}^2$ are defined respectively as,

$$\gamma^T = [\gamma_1 \quad \gamma_2 \quad \gamma_3] \tag{3.9}$$

$$\theta^T = [\alpha \quad \beta] \tag{3.10}$$

For reasons of simplicity, we will combine the parameters presented in equations (3.9) and (3.10) into one vector,

$$\Theta^T = [\gamma_1 \quad \gamma_2 \quad \gamma_3 \quad \alpha \quad \beta] \tag{3.11}$$

In order to assure a unique parametrization for the Hammerstein system shown in figure (3.1), we must set the steady-state gain $\mathcal{K}_{DC}$ of linear dynamic part $G_p(q^{-1})$ equal to 1,

$$\mathcal{K}_{DC} = \frac{\beta}{1+\alpha} = 1 \tag{3.12}$$

3.2.2.1 Feasible Parameter Set

We can define the Feasible Parameter Set (FPS) for the Hammerstein system as follows:

$$\begin{aligned} \mathcal{D}_\Theta = \{\Theta \in \mathbb{R}^5 : w_t = -\alpha w_{t-1} + \beta x_{t-1}; w_t = y_t + \eta_t; x_t = \gamma_1 v_t^5 + \gamma_2 v_t^4 + \gamma_3 v_t^3; \\ v_t = \mathcal{K} u_t; |\eta_t| \leq \Delta_\eta; \forall t = 2, ..., N\} \end{aligned} \tag{3.13}$$

where N is the number of measurements.

3.2.2.2 Extended Feasible Parameter Set

We need to extend the FPS represented in (3.13) because we have to consider the noise η_t entering the problem, hence we will have new parameters to be added, represented in the noise entering the problem. These parameters will not be estimated.
Now the extended Feasible Parameter Set (FPS) is as follows:

$$\mathcal{D}_{\Theta,\eta} = \{\Theta \in \mathbb{R}^5, \eta \in \mathbb{R}^N : y_t - \eta_t = -\alpha y_{t-1} + \alpha\eta_{t-1} + \beta x_{t-1}; x_t = \gamma_1 v_t^5 + \gamma_2 v_t^4 + \gamma_3 v_t^3; \\ v_t = \mathcal{K}u_t; |\eta_t| \leq \Delta_\eta; \forall t = 2, ..., N\} \tag{3.14}$$

3.2.2.3 Parameter Uncertainty Intervals

The Parameter Uncertainty Intervals (PUI) can be described by considering the upper and lower bounds for the parameters to be estimated,

$$\Theta_i^{min} = \min_{\mathcal{D}_\Theta} \Theta_i \quad , \quad i = 1, ..., 5 \tag{3.15}$$

$$\Theta_i^{max} = \max_{\mathcal{D}_\Theta} \Theta_i \quad , \quad i = 1, ..., 5 \tag{3.16}$$

$$PUI_{\Theta_i} = [\Theta_i^{min}, \Theta_i^{max}] \tag{3.17}$$

The Central Estimate for the parameters,

$$\Theta_i^C = \frac{\Theta_i^{min} + \Theta_i^{max}}{2} \tag{3.18}$$

3.2.2.4 Stability

To assure the the stability of the linear model represented in equation (3.1), we need to enforce a constraint on the optimization problem to be solved [8]. Since the structure of the linear model is a first order discrete time transfer function with gain β and pole α, to assure that the system is BIBO stable (Bounded-Input, Bounded-Output), the pole $alpha$ must lay within the bounds of the pole-zero map shown in figure (3.2). The condition to be enforced is,

$$|\alpha| < 1 \tag{3.19}$$

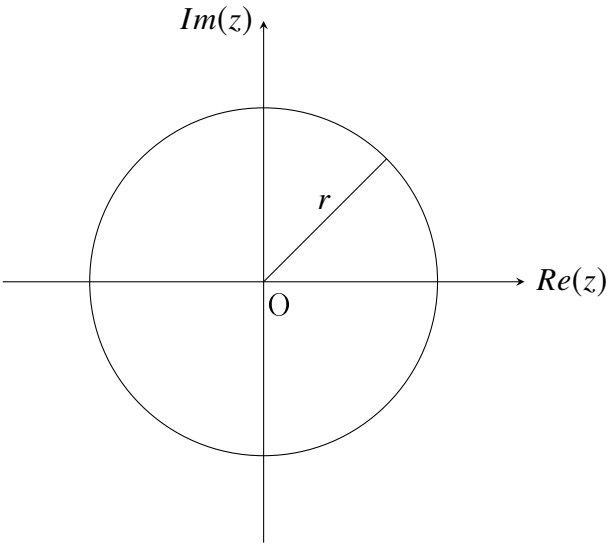


Figure 3.2: Pole-Zero Map in the z Domain (r=1)

3.2.2.5 Optimization Problem Formulation

For the optimization problem (3.15) the formulation considering all the constraints is as follows,

$$\begin{aligned}
\Theta_i^{min} &= \min_{\mathcal{D}_\Theta} \Theta_i \\
&\textit{subject to} \\
&y_t + \alpha y_{t-1} - \beta x_{t-1} - \eta_t - \alpha \eta_{t-1} = 0, \\
&x_t = \gamma_1 v_t^5 + \gamma_2 v_t^4 + \gamma_3 v_t^3, \\
&v_t = \mathcal{K} u_t, \\
&|\eta(t)| \leq \Delta_\eta, \\
&|\alpha| < 1, \\
&\forall t = 2, ..., N
\end{aligned} \tag{3.20}$$

For the optimization problem (3.16) the formulation considering all the constraints is as follows,

$$\begin{aligned}
\Theta_i^{max} &= \max_{\mathcal{D}_\Theta} \Theta_i \\
&\textit{subject to} \\
&y_t + \alpha y_{t-1} - \beta x_{t-1} - \eta_t - \alpha \eta_{t-1} = 0, \\
&x_t = \gamma_1 v_t^5 + \gamma_2 v_t^4 + \gamma_3 v_t^3, \\
&v_t = \mathcal{K} u_t, \\
&|\eta(t)| \leq \Delta_\eta, \\
&|\alpha| < 1, \\
&\forall t = 2, ..., N
\end{aligned} \tag{3.21}$$

3.2.2.6 Optimization Problem Characteristics

In regards to the optimization problem to be solved we can point out the following:

- The problem will be solved under the Set-Membership framework (introdued in chapter 2).
- This problem is a Polynomial Optimization Problem (POP).
- Since this optimization problem is in the class of an Output-Error structure, as the noise entering the process is affecting the measured output, we can classify this problem as a nonconvex problem.
- In order to be able to solve this optimization problem, we have to apply convex relaxation techniques in order to guarantee the convergence of the parameters represented in (3.11) to the global minima.

3.2.3 Results

Following the procedure presented in subsection 3.2.2, the optimization problem has been solved using SparsePOP package under the MATLAB environment (for more details about SparsePOP see [9]).
The optimization problem is solved in 3 phases as described in the following points:

- Estimate the bound for the error entering the optimization problem.
- Estimate the lower and upper bounds for the parameters shown in (3.11).
- Validate the computed model using data sets different from the one used to perform the model identification.

The error bound for the noise affecting the output measurement was found to be:

$$\Delta_\eta = 1.41$$

As for the estimation of the parameters in (3.11), the minimum relaxation order δ was found to be equal to 1,

$$\delta \geq \frac{\rho}{2}$$

where ρ is the highest degree of the polynomial described in equation (3.1).
The Parameter Uncertainty Intervals and the Central Estimates for the parameters in (3.11) with a relaxation order δ equal to 2:

$$PUI_{\Theta} = \begin{array}{r} \\ \alpha\in \\ \beta\in \\ \gamma_1\in \\ \gamma_2\in \\ \gamma_3\in \end{array} \begin{array}{c} \overbrace{\quad\quad}^{\Theta^{min}} \quad \overbrace{\quad\quad}^{\Theta^{max}} \\ \begin{bmatrix} -0.651574 & -0.651257 \\ +0.348427 & +0.348777 \\ -0.002722 & -0.0027232 \\ +0.039016 & +0.039050 \\ -0.081878 & -0.081552 \end{bmatrix} \end{array}$$

$$\Theta^{C} = \begin{array}{r} \\ \alpha= \\ \beta= \\ \gamma_1= \\ \gamma_2= \\ \gamma_3= \end{array} \begin{bmatrix} -0.65142 \\ +0.34860 \\ -0.00272 \\ +0.03903 \\ -0.08172 \end{bmatrix}$$

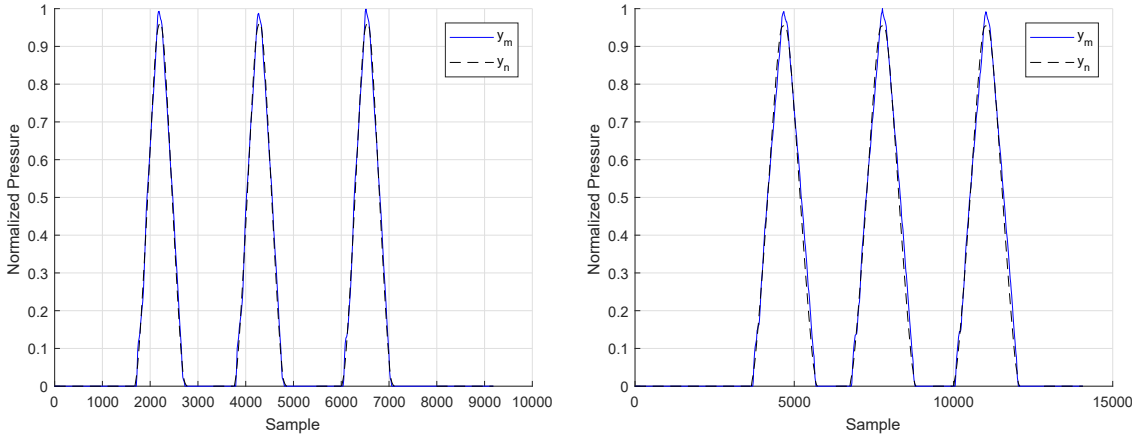
The Hammerstein model in figure (3.1) can now be described with the following equations:

$$G_p(q^{-1}) = \frac{0.3706738}{1 - 0.62932623q^{-1}} \iff G_p(z) = \frac{0.3706738}{z - 0.62932623}$$

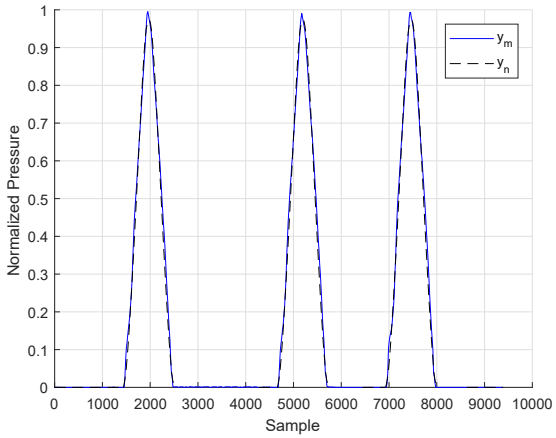
$$x_t = -0.0026051v_t^5 + 0.0375240v_t^4 - 0.0771024v_t^3$$

From the obtained results we can see that the stability condition mentioned in 3.2.2.4 has been satisfied.

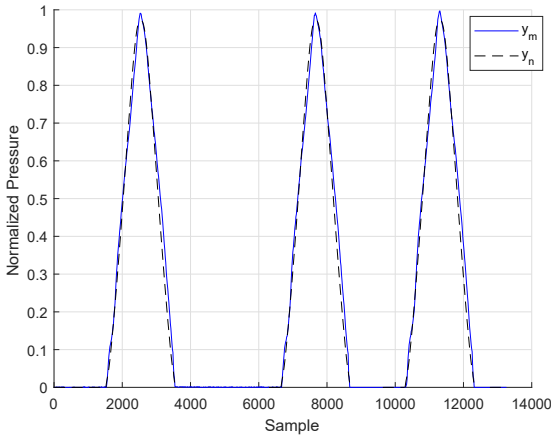
The identification was performed using only part of the measurements from data set [20°C Ramp_a]. The next figure (3.3) shows validation for the identified model for four data sets from Set (1). y_m is the measured pressure and y_n is the simulated pressure.



(a): 20°C Ramp_a (b): 20°C Ramp_b



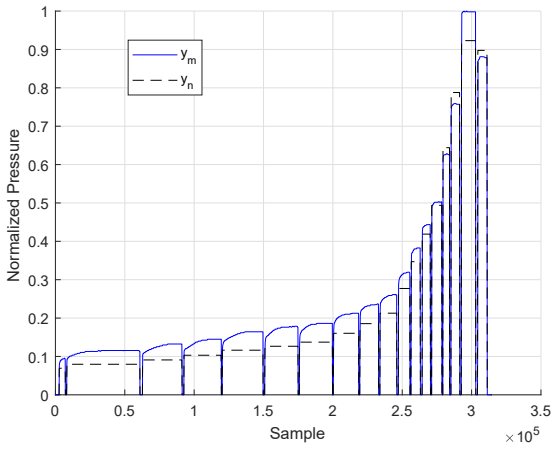
(c): 60°C Ramp_a



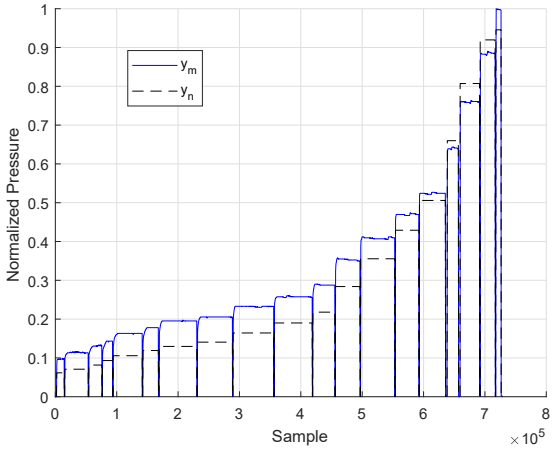
(d): 60°C Ramp_b

Figure 3.3: Hammerstein Model Validation for 4 Different Data Sets from Set (1)

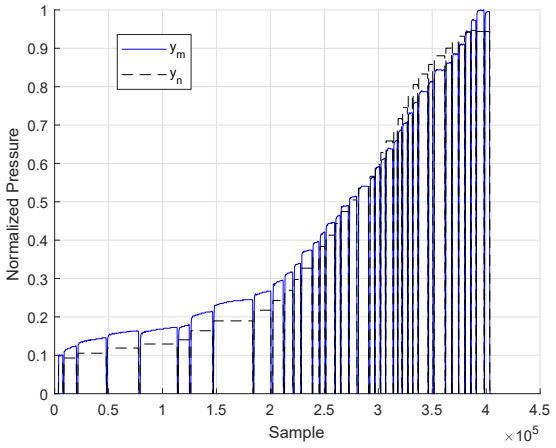
As the Identification process was performed and validated so far in regards to Set (1) (Ramp Data Sets), the next step was to validate the obtained model using Set (2) (Step Data Sets), the results can be seen in figure (3.4),



(a): 20°C Step_a



(b): 60°C Step_a



(c): -20°C Step_a

Figure 3.4: Hammerstein Model Validation for 3 Different Data Sets from Set (2)

3.2.3.1 System Identification - Step Data Sets

HS20 Model: Hammerstein Model identified using the MATLAB System Identification Toolbox

We now deploy the data set [20°C Step_a] to model the Hammerstein System using the MATLAB System Identification Toolbox. Several structures were tried and the most suitable model can be seen in equations (3.22), (3.23). The structure is the same as presented in figure (3.1).

$$w_t = G_p(q^{-1})x_t = \frac{1}{1 - 0.9562}x_t \tag{3.22}$$

$$x_t = 7.31 \times 10^{-15}u_t^5 - 2.28 \times 10^{-11}u_t^4 + 2.36 \times 10^{-8}u_t^3 - 6.9 \times 10^{-6}u_t^2 + 5.77 \times 10^{-4}u_t \tag{3.23}$$

From figure (3.5) below, it can be seen that the simulated output pressure y_n is not able to track the measured output pressure y_m. Figure (3.6) shows that the *HS*20 model is not able to reproduce the measured output pressure y_m for the data set [60°C Step_a], which is a different data set than the one used to model the plant.

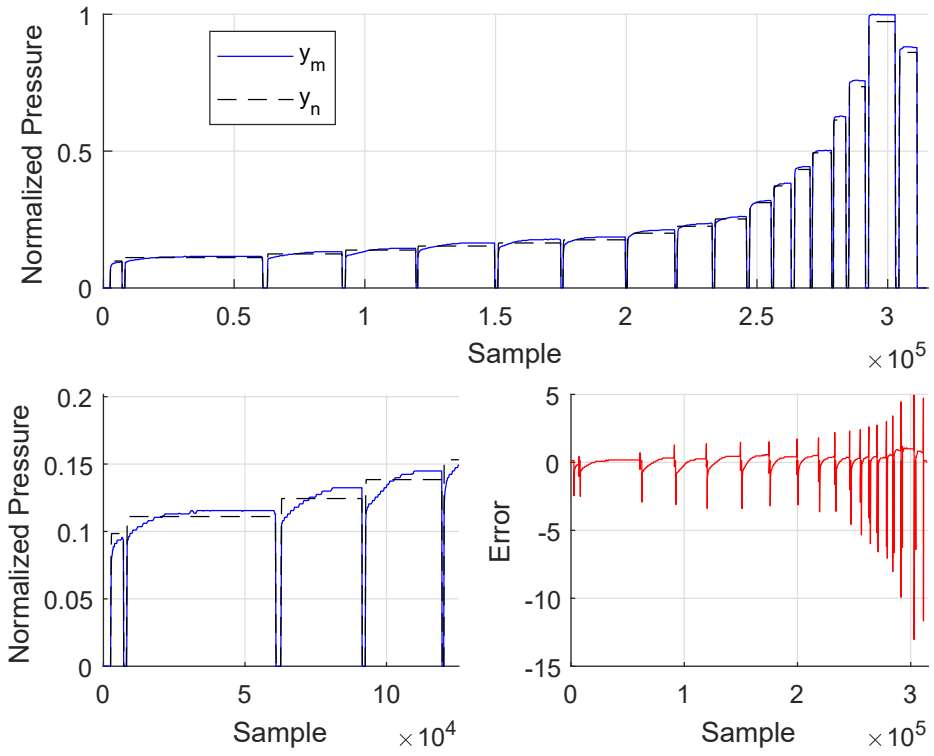


Figure 3.5: HS20 Model Validation for Data Set 20°C Step_a using MATLAB System Identification Toolbox

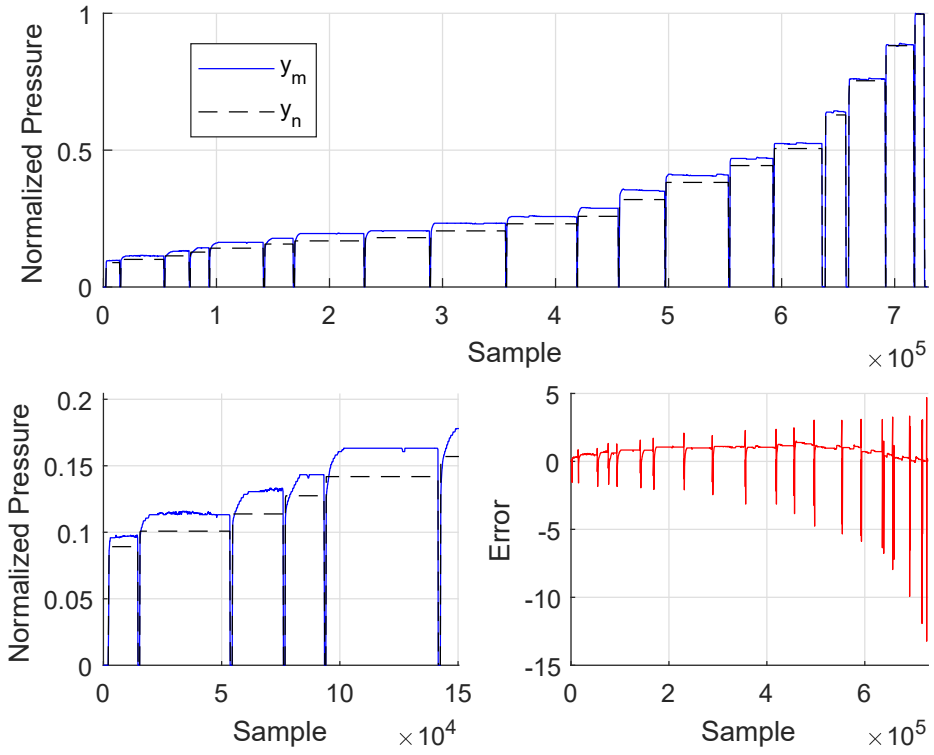


Figure 3.6: HS20 Model Validation for Data Set 60°C Step_a using MATLAB System Identification Toolbox

3.2.4 Conclusions

From the results introduced in subsection 3.2.3, we can conclude this section by stating the following points:

- The system identification process for the Hammerstein system provided relatively good results for the Ramp data sets. However, for the Step data sets, the Hammerstein model was not able to cope up with the physical system's dynamics.
- The physical system's dynamics are varying in correspondence to the input current in a nonlinear manner.
- For every input range corresponds a different model for the plant $G_p(q^{-1})$.
- Several models have to be identified in order to cover up the entire range of input/output data, assuming that the operating points are the ones available in the data sets.
- The above mentioned points motivated the need to use recursive estimation for this problem that will be introduced in the next section 3.3.

3.3 Recursive Model Estimation

3.3.1 Overview

In section 3.2, we concluded with the fact that we need to use recursive estimation in order to obtain a model for the plant. In this section we will introduce the recursive model estimation identification technique that allows to estimate a model for the plant based on input and output data collected at each sampling instant.

Recursive estimation is based on the selection of a cost function such that the resultant recursive algorithm is in the form of a least squares problem. It also implies minimizing the one step-ahead prediction error which represents only that part of y_t that cannot be predicted from the past data [10].

The starting points for this identification process can be summarized in:

- The system to be identified $G_p(q^{-1})$ is a single-input single-output system having one pole α_t and a gain β_t (most suitable model).
- The sampling time T_s is equal to 2 ms, where at each sampling instant, a model for the plant is estimated recursively.

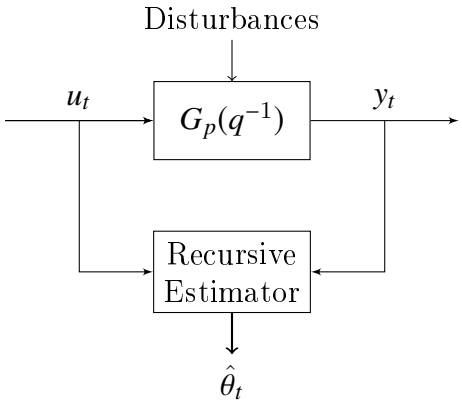


Figure 3.7: Recursive Estimation Model Layout

3.3.2 Problem Formulation

In reference to the points stated in subsection 3.3.1, we can now introduce the main structure for the recursive estimation algorithm. In this subsection we are going to introduce 4 algorithms for recursive estimation [3]:

(a) Forgetting Factor

(b) Kalman Filter

(c) Unnormalized Gradient

(d) Normalized Gradient

Now we can introduce the general form for the recursive estimation algorithm, taking into account that the main difference between all four algorithms mentioned above is the gain matrix K_t.
The recursive estimation model can be described as,

$$y_t = \frac{\beta_t}{1 + \alpha_t q^{-1}} u_t \tag{3.24}$$

The parameter vector $\hat{\theta}_t \in \mathbb{R}^2$ is defined according to the following equation,

$$\hat{\theta}_t = \hat{\theta}_{t-1} + K_t e_t \tag{3.25}$$

$$\hat{\theta}_t^T = [\alpha_t \quad \beta_t] \tag{3.26}$$

where e_t is the prediction error and is given by,

$$e_t = y_t - \varphi_t^T \hat{\theta}_{t-1} \tag{3.27}$$

The regression vector φ_t which is computed on the basis of previous input u_t and output y_t measurements,

$$\varphi_t = [-y_{t-1} \quad u_{t-1}] \tag{3.28}$$

The gain matrix K_t is as follows,

$$K_t = Q_t \varphi_t \tag{3.29}$$

Q_t is computed as follows depending on the estimation algorithm:

Forgetting Factor

$$Q_t = \frac{P_{t-1}}{\lambda + \varphi_t^T P_{t-1} \varphi_t} \tag{3.30}$$

λ is the forgetting factor which exponentially, determines the weight of older samples to the estimation at time t.

$if \lambda = 1 \rightarrow$ *Time-Invariant Parameters Estimation*

$if \lambda < 1 \rightarrow$ *Time-Varying Parameters Estimation*

λ is usually selected between 0.98 and 0.995.
P_t is the prediction error variance and is given by,

$$P_t = \frac{1}{\lambda} \left(P_{t-1} - \frac{P_{t-1} \varphi_t \varphi_t^T P_{t-1}}{\lambda + \varphi_t^T P_{t-1} \varphi_t} \right) \tag{3.31}$$

Kalman Filter

$$Q_t = \frac{P_{t-1}}{R_2 + \varphi_t^T P_{t-1} \varphi_t} \tag{3.32}$$

R_2 is the variance of the process noise e_t.
The prediction error variance P_t is given by,

$$P_t = P_{t-1} + R_1 - \frac{P_{t-1}\varphi_t \varphi_t^T P_{t-1}}{R_2 + \varphi_t^T P_{t-1} \varphi_t} \tag{3.33}$$

R_1 is the covariance matrix (drift matrix).
R_1 and $P_{t=0}$ are scaled such that R_2 is equal to zero.

Normalized & Unnormalized Gradient

For the Unnormalized Gradient algorithm,

$$Q_t = \gamma \tag{3.34}$$

where γ is the adaption gain.
As for the Normalized Gradient algorithm,

$$Q_t = \frac{\gamma}{\|\varphi_t\|^2 + Bias} \tag{3.35}$$

The adaptation gain γ is scaled at each time instant by $\|\varphi_t\|^2$. If φ_t approaches zero, it may cause huge variations in the estimated parameters and lead to instability. To prevent this from happening, the *Bias* term is added.

3.3.2.1 Stability

To assure the the stability of the recursive model represented in equation (3.24), the estimation algorithm verifies whether the absolute value for estimated pole α_t is less than one. In case the absolute value of the pole is greater than or equal to one, the algorithm assigns the pole a value of $|\alpha_t| = 0.99$ depending on it's estimated value's sign. After enforcing the pole α_t, the algorithm recomputes the value for the gain β_t on the basis of having α_t as apriori information.

3.3.3 Results

The following figures represent a compassion between the four algorithms mentioned in subsection 3.3.2 for the data set [20°C Step_a]. The ***Bias*** for all the results presented in this subsection is equal to 2.2204×10^{-16}. The measured output pressure is represented by y_m and the simulated output pressure is y_n.

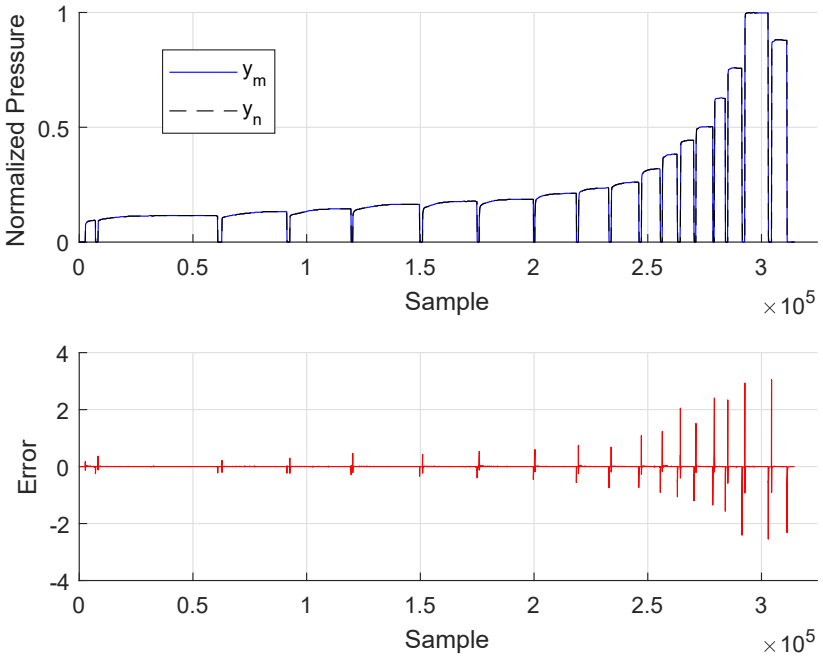


Figure 3.8: Forgetting Factor Algorithm with $\lambda = 0.995$

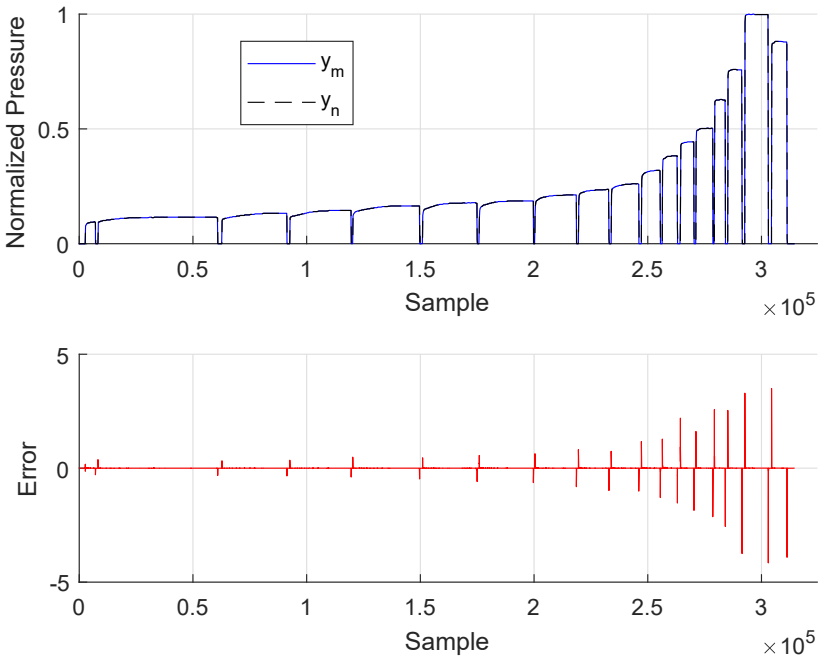


Figure 3.9: Kalman Filter Algorithm

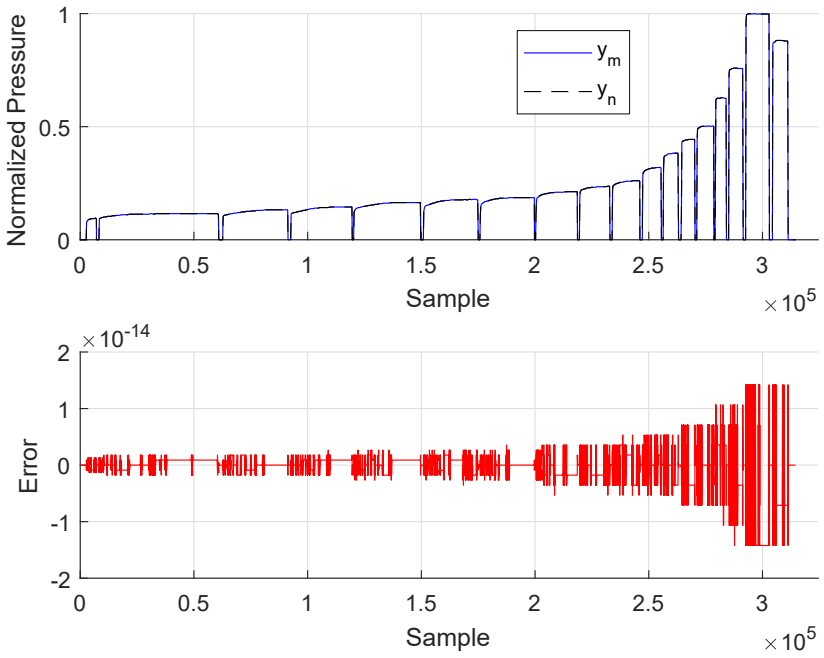


Figure 3.10: Normalized Gradient Algorithm with $\gamma = 1$

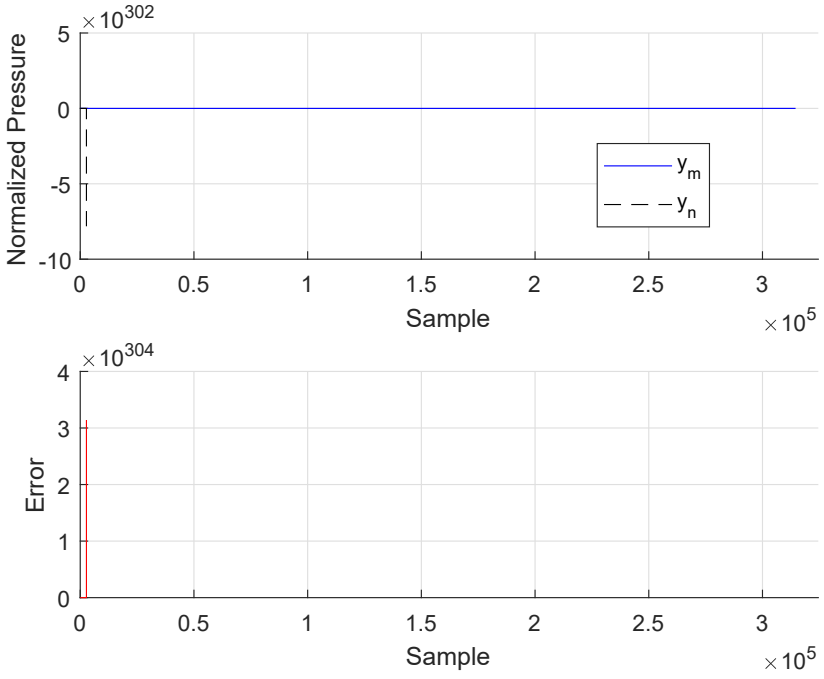


Figure 3.11: Unnormalized Gradient Algorithm with $\gamma = 1$

As it can be seen from figures (3.8), (3.9), (3.10), (3.11), the Normalized Gradient algorithm proves to be the most suitable algorithm. As for the Unnormalized algorithm, it can be clearly seen that it can not cope up with the rapidly changing dynamics of the system for all possible values for the adaption gain γ.

It is necessary to note that the adaptation gain γ was selected equal to 1, which provides the best results as it can be seen in figure (3.12) which shows the error for several models with different adaption gain γ values (this was performed on the data set [20°C Ramp_a]).

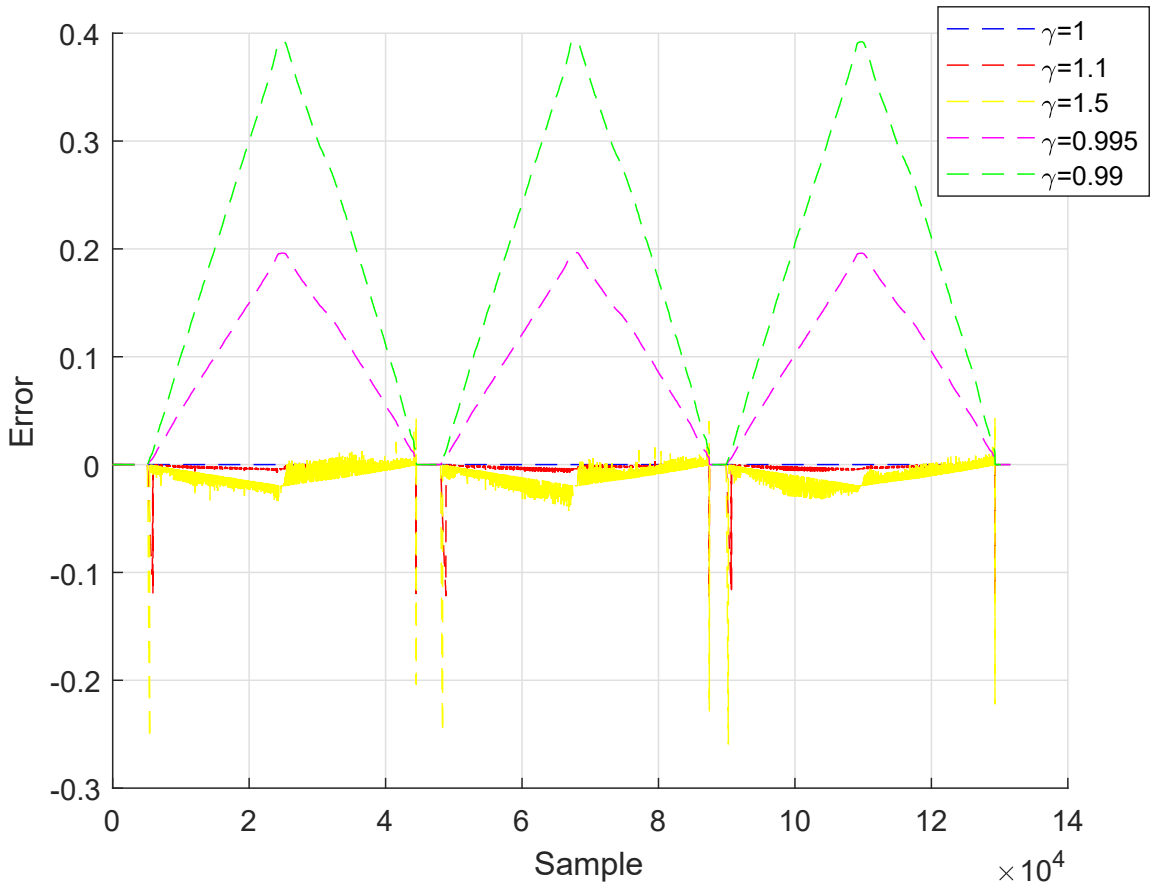


Figure 3.12: Adaption Gain γ Effect

The following figures represent the results obtained using the Normalized Gradient algorithm described in subsection 3.3.2 for various data sets. The measured output pressure is represented by y_m and the simulated output pressure is y_n.
Figures (3.18), (3.19) and (3.20), show the results for data sets with **Hysteresis**.

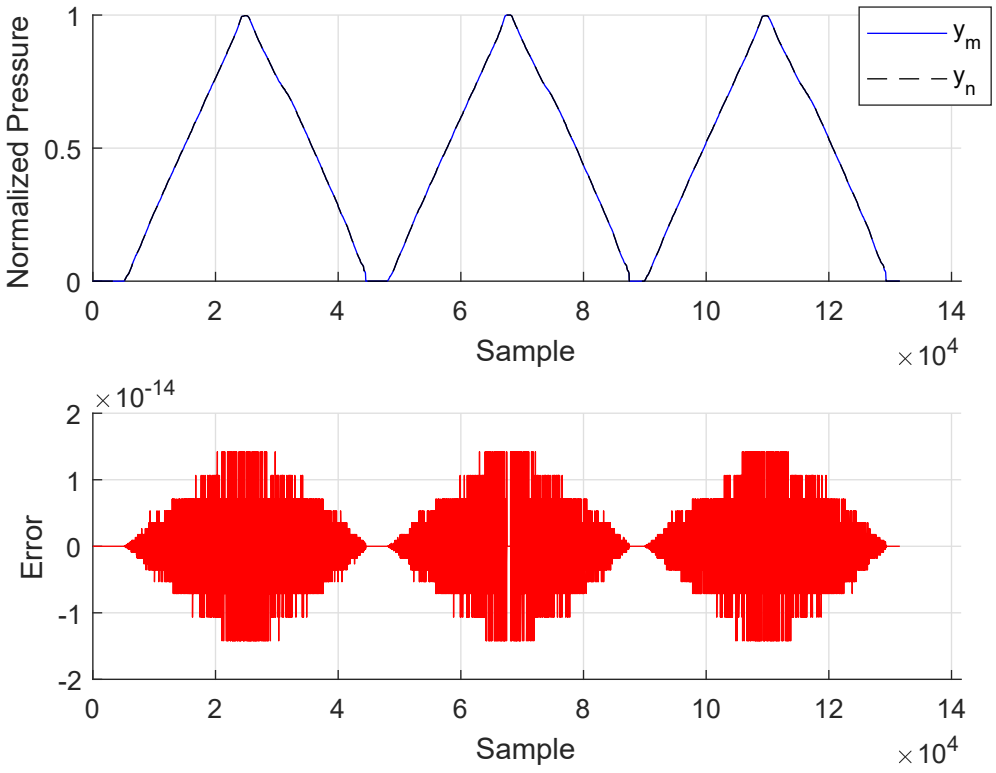


Figure 3.13: Recursive Estimation Validation for Data Set 20°C Ramp_a

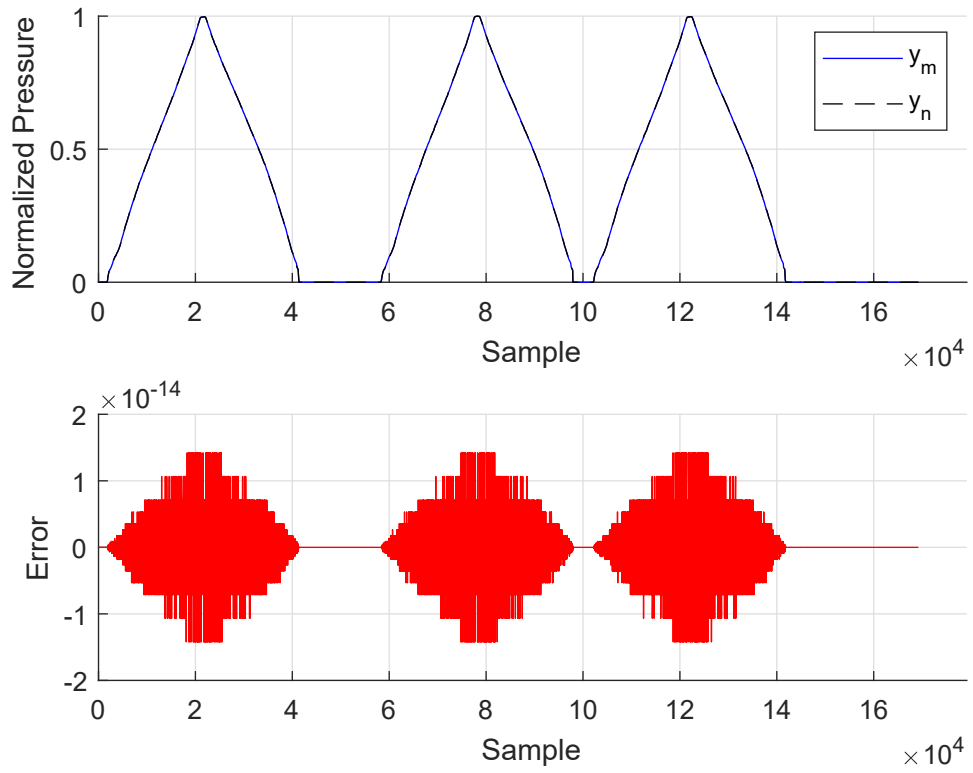


Figure 3.14: Recursive Estimation Validation for Data Set 60°C Ramp_a

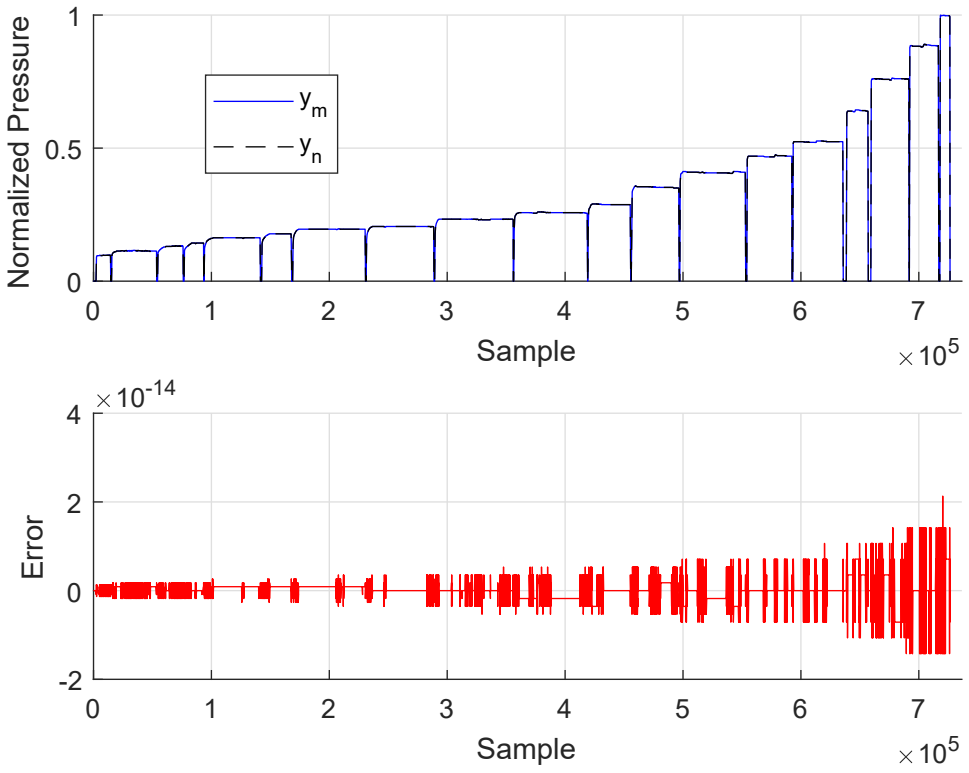


Figure 3.15: Recursive Estimation Validation for Data Set 60°C Step_a

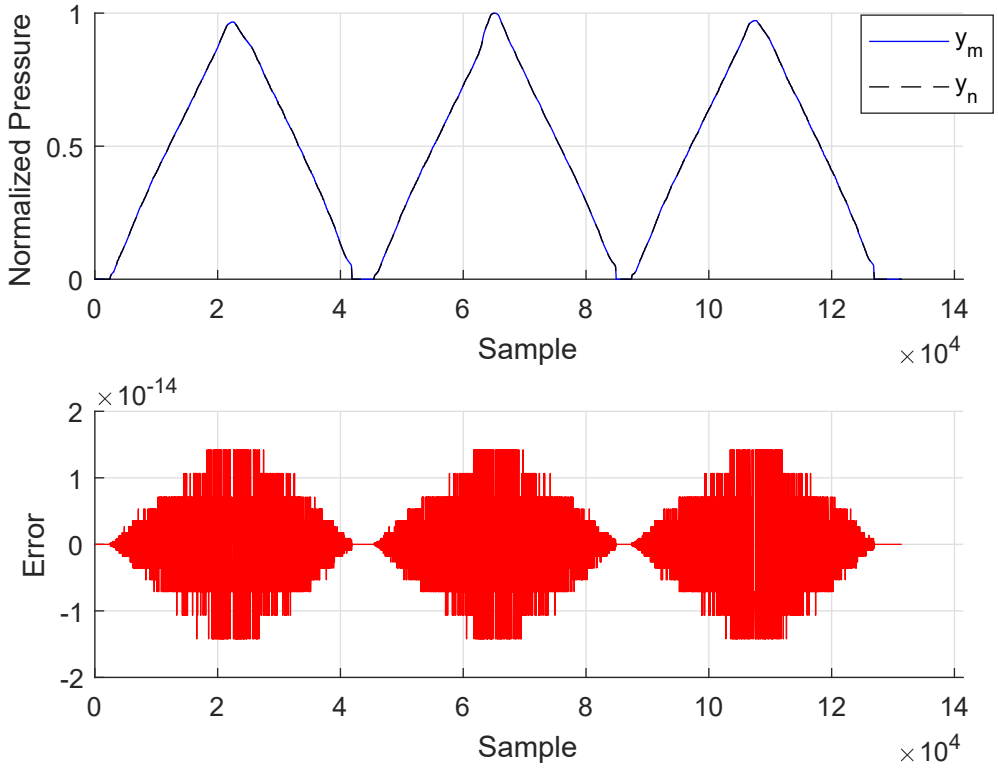


Figure 3.16: Recursive Estimation Validation for Data Set -20°C Ramp_a

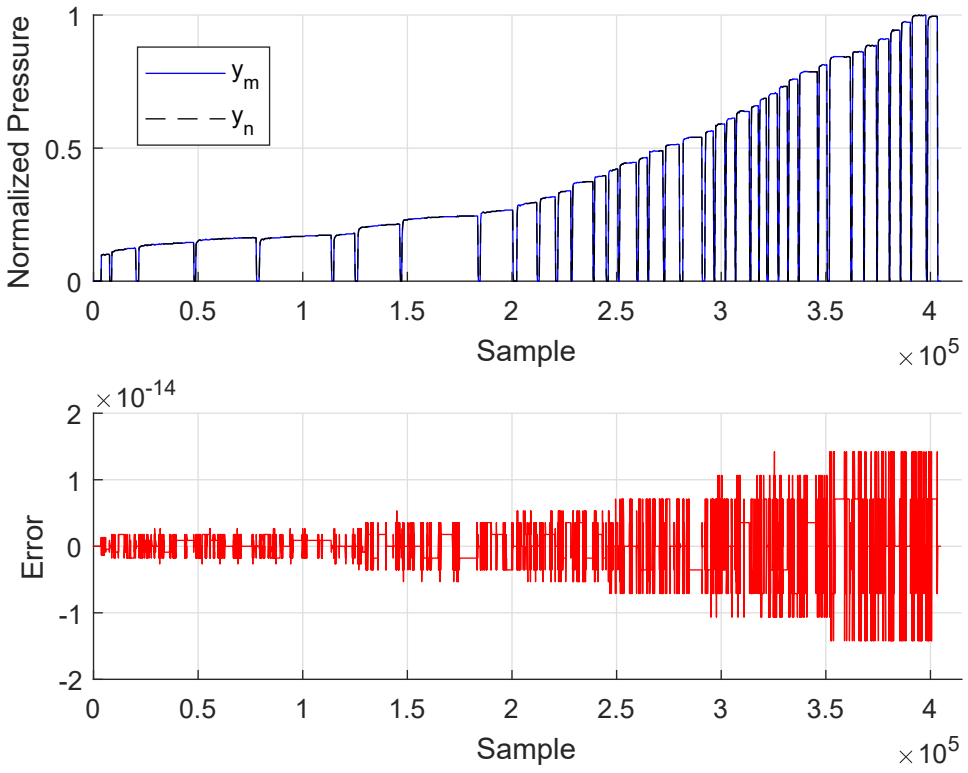


Figure 3.17: Recursive Estimation Validation for Data Set -20°C Step_a

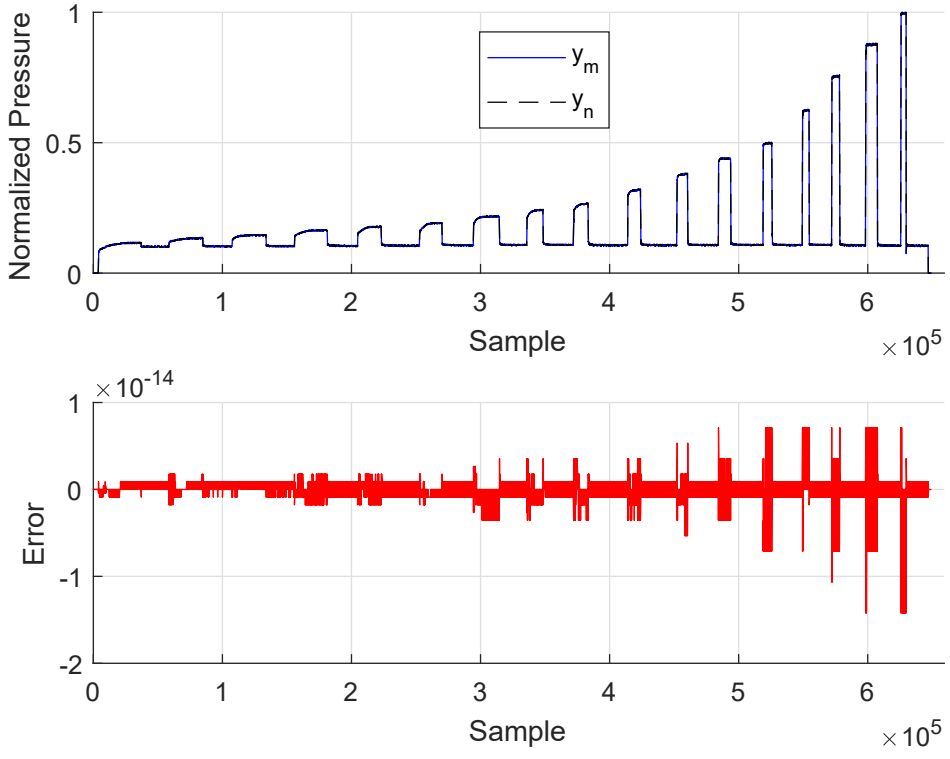


Figure 3.18: Recursive Estimation Validation for Data Set 20°C Step_b

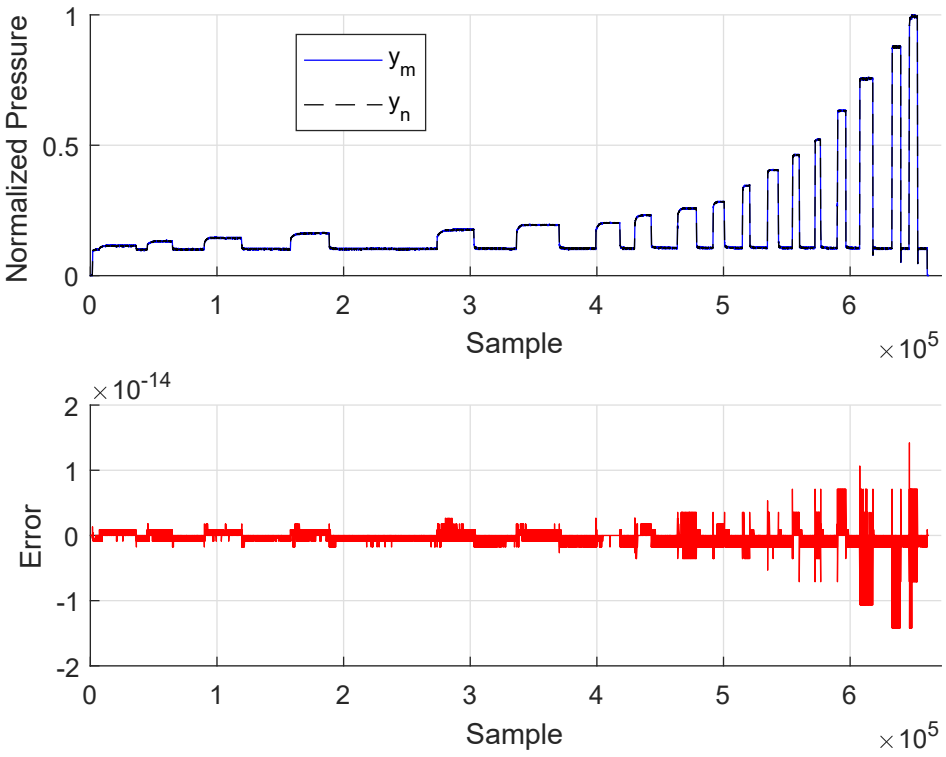


Figure 3.19: Recursive Estimation Validation for Data Set 60°C Step_c

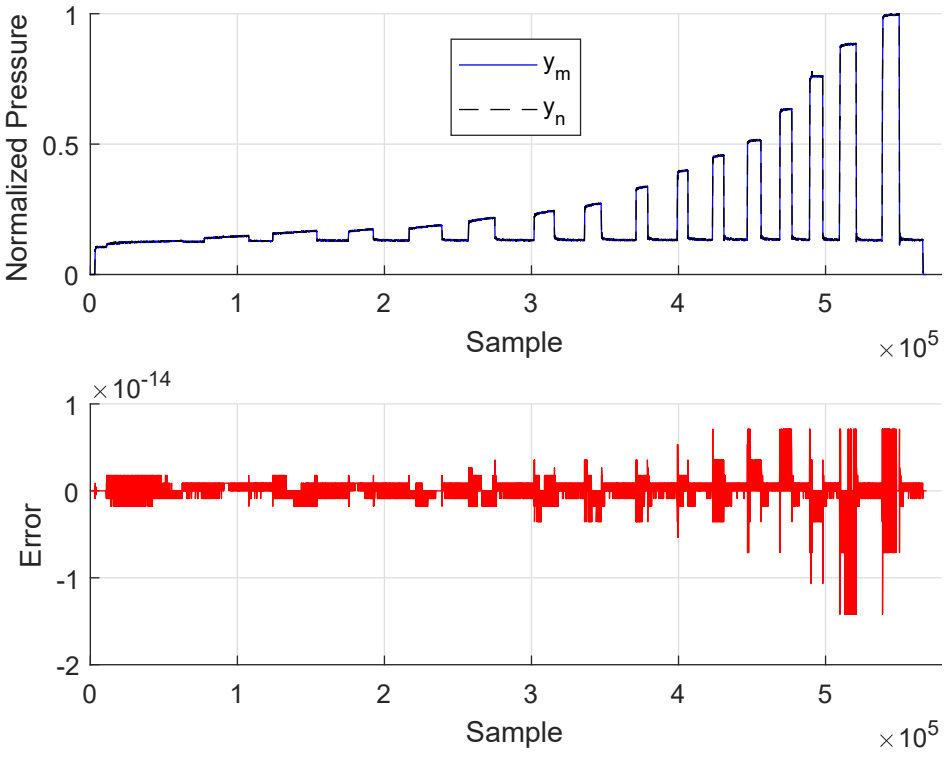


Figure 3.20: Recursive Estimation Validation for Data Set -20°C Step_c

3.3.4 Conclusions

From the results obtained in subsection 3.3.3, we can conclude with the following:

- The recursive estimation algorithm proves to be suitable for this problem as it can be seen from the obtained results. The only exception is the Unnormalized Gradient algorithm as seen in figure (3.11).
- The absolute error between the measured output pressure y_m and the simulated pressure y_n is relatively very small and is bounded by $|2 \times 10^{-14}|$.
- The Hammerstein system provided relatively good results for Set (1) but wasn't able to cope with the system's dynamics for Set (2). On the other hand the recursive estimation algorithm was able capture all the dynamics in both sets.

Chapter 4

Neural Networks

4.1 What is a Neural Network?

Definition: An artificial neural network (ANN), commonly referred to as *neural network*, is a computational model which is vaguely inspired by the human brain as it consists of an interconnected network of simple processing units (neurons) that learns from experience by modifying its connections [11].

A Neural network can be seen as a machine that is designed to model how the brain performs specific tasks. Two important aspects that can be seen to resemble the brain are [12]:

- A learning process is used to acquire information/knowledge from the network's environment.
- Acquired information/knowledge is stored using interneuron connection strengths (also known as synaptic weights).

The first initiative for neural networks was to model animal brains. They are now used as machine learning tools in various application fields such as, but not limited to:

(a) System Identification and Control

(b) Medical Diagnosis

(c) Pattern and Sequence Recognition

(d) Finance

(e) Machine Translation

(f) Data Mining

4.2 Properties and Capabilities

A neural network attains its computation capabilities through the massively distrusted structure and it's ability to both learn and generalize [12]. The generalization capability is a direct reflection of the network's ability to produce outputs for inputs that were not

part of the network's training (learning) available information (see section 4.5 for more details on learning processes).

Some of the main properties of a neural network can be summarized in the following points:

- Input-Output Mapping: The network has the capability to provide a reasonable output, given data that was not used to train the network.
- Adaptivity: A network trained to operate in specific operating conditions can be retrained in order to adapt with changes in these conditions.
- Nonlinearity: This property makes neural networks suitable for many kind of applications.
- Evidential Response: This property is linked to the confidence interval of decisions made by the network.
- Uniformity of Analysis and Design: Neural networks are considered as universal information processors.

4.3 General Structure of a Neural Network

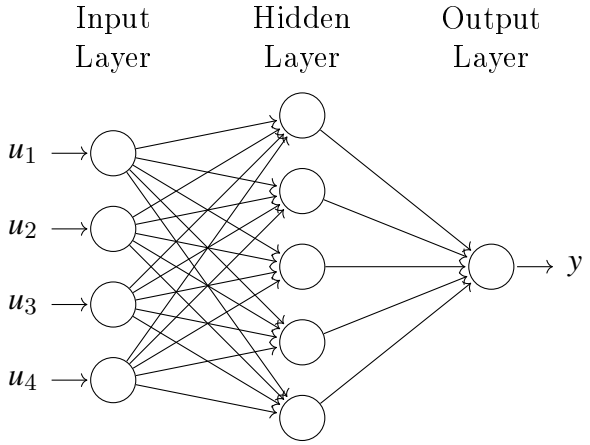


Figure 4.1: Neural Network General Structure

As it can be seen from figure (4.1), the nueral network consists of three layers:

- Input Layer: Input data acquisition (passive node/s)
- Hidden Layer: Information processing unit, made of 1 or more neurons (active node/s). The neuron will be described in subsection 4.3.1
- Output Layer: Neuron/s output data processing and computations to provide the network's output (active node/s).

4.3.1 Neuron Model

A neuron is the core operation element for the neural network as it is the unit responsible for information processing.

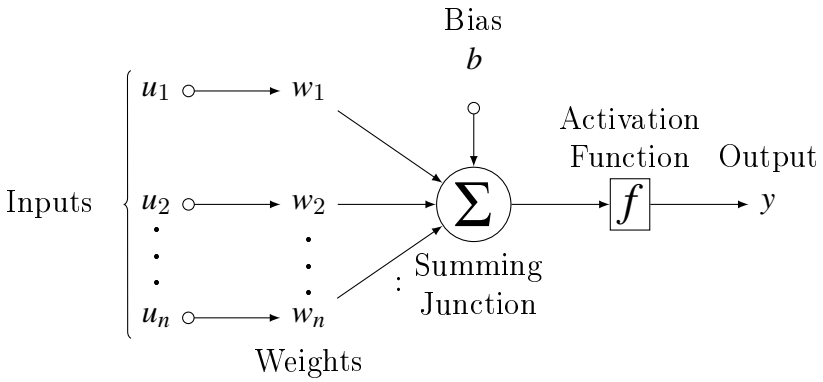


Figure 4.2: Model of a Neuron

The general structure of a neuron can be seen in figure (4.2), each neuron consists of:

- Weights $w_1, ..., w_n$, which determine the influence of each input signal $u_1 ... u_n$ on the output y.
- Adder Σ, to sum weighted inputs $u_1 w_1, ..., u_n w_n$.
- Bias b, which is added to the resultant summation of the weighted inputs. It increases/decreases the summed weighted inputs to the activation function.
- Activation Function f, to limit the output of the neuron to a finite value as described in subsection 4.3.2.

The output y of the neuron,

$$y = f(\overbrace{\sum_{i=1}^{n} u_i w_i + b}^{\varphi}) \tag{4.1}$$

The notaion φ will be used to describe the terms $\boxed{\sum_{i=1}^{n} u_i w_i + b}$

4.3.2 Activation Function

The activation function f computes the output of the neuron from φ as it can be seen in equation (4.1). The main purpose of the activation function is to introduce nonlinearity into the network.

Some of the most common activation functions used in neural networks are:

- Bipolar Function
- Step Function
- Linear Function
- Sigmoid Function
- Tanh Function
- ReLU Function

4.3.2.1 Bipolar Function

We can define the threshold function for the activation function f of the neuron as follows,

$$f(\varphi) = \begin{cases} 1 & \text{if } \varphi > 0 \\ 0 & \text{if } \varphi = 0 \\ -1 & \text{if } \varphi < 0 \end{cases} \tag{4.2}$$

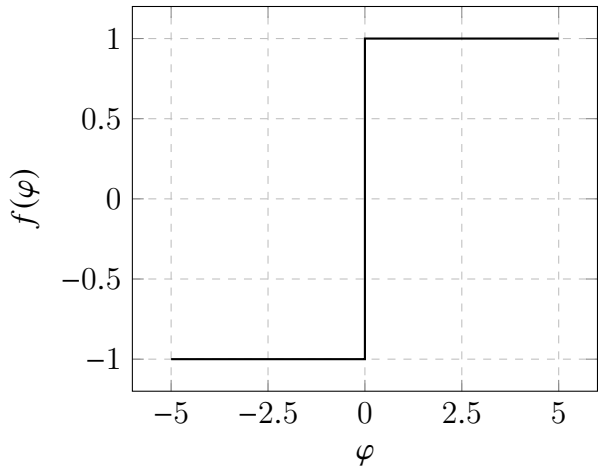


Figure 4.3: Bipolar Activation Function

4.3.2.2 Step Function

We can define the threshold function for the activation function f of the neuron as follows,

$$f(\varphi) = \begin{cases} 1 & \text{if } \varphi \geqslant 0 \\ 0 & \text{if } \varphi < 0 \end{cases} \tag{4.3}$$

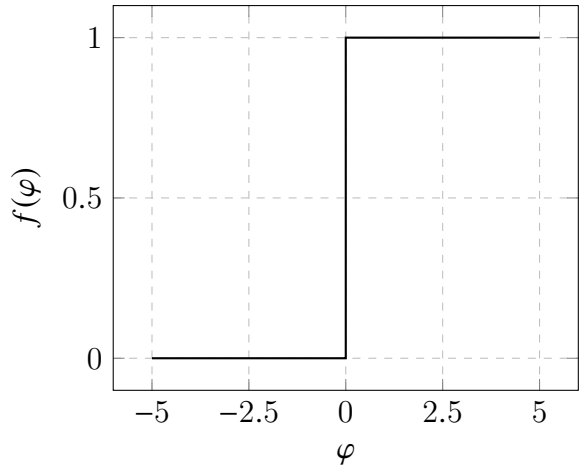


Figure 4.4: Step Activation Function

4.3.2.3 Linear Function

We can define the equation for the activation function f of the neuron as follows,

$$f(\varphi) = \varphi \tag{4.4}$$

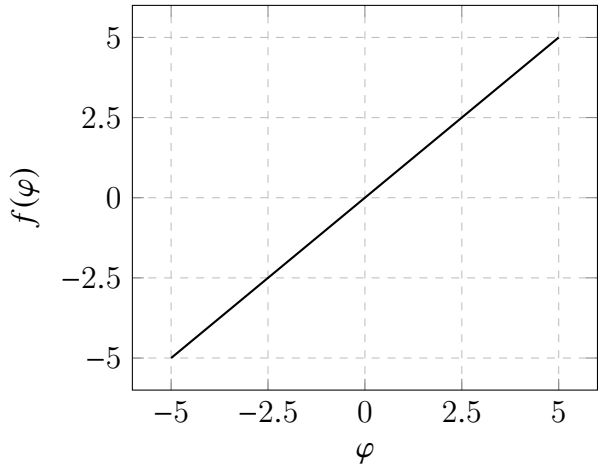


Figure 4.5: Linear Activation Function

4.3.2.4 Sigmoid Function

This is the most commonly used activation function. We can describe the the activation function f of the neuron with the following,

$$f(\varphi) = \frac{1}{1 + e^{-\varphi}} \tag{4.5}$$

The threshold function for the sigmoid activation function is the same as for bipolar function described in (4.2).

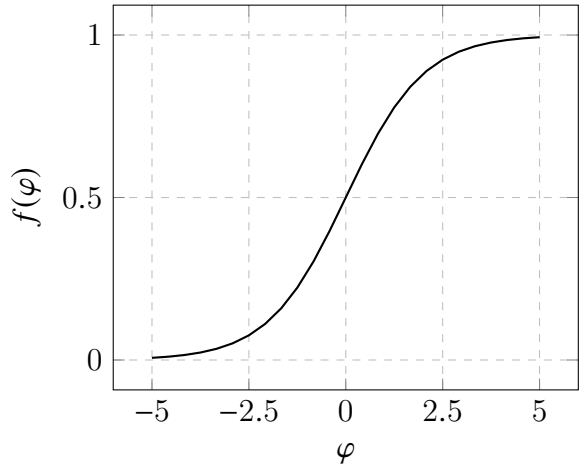


Figure 4.6: Sigmoid Activation Function

4.3.2.5 Tanh Function

Hyperbolic Tangent Function (Tanh) is similar to the sigmoid function but adds one extra advantage, the threshold function in (4.2) is not necessary anymore due to the fact that this function is bounded within the range (-1, 1). The Tanh function is also widely used in various applications.
The Tanh activation function is described as follows,

$$f(\varphi) = \tanh(\varphi) = \frac{2}{1 + e^{-2\varphi}} - 1 \tag{4.6}$$

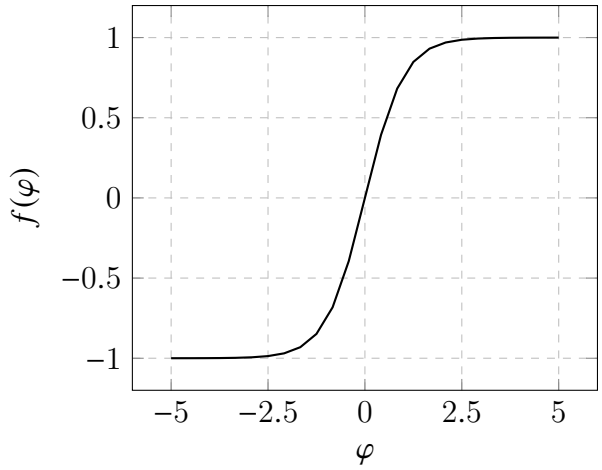


Figure 4.7: Tanh Activation Function

4.3.2.6 ReLU Function

Rectified Linear Units (ReLU) are widely used in hierarchical learning (deep learning) networks in the fields of computer vision and speech recognition.
The activation function f of the neuron is formulated as follows,

$$f(\varphi) = \begin{cases} \varphi & \text{if } \varphi \geqslant 0 \\ 0 & \text{if } \varphi < 0 \end{cases} \tag{4.7}$$

A smooth approximation for the activation function in equation (4.7) is given by,

$$f(\varphi) = \log(1 + e^{\varphi}) \tag{4.8}$$

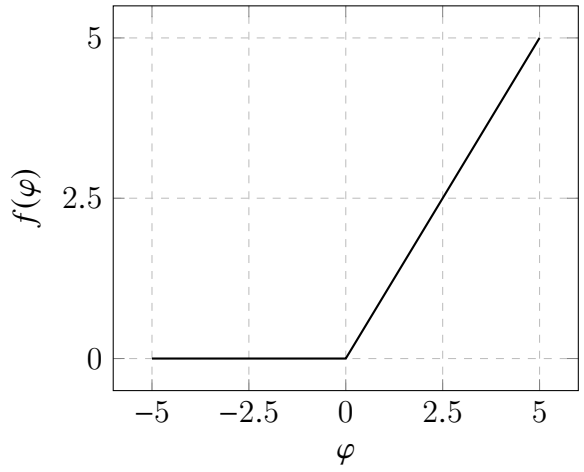


Figure 4.8: ReLU Activation Function

4.4 Neural Network Architectures

A neural network is a data processing system that has a number of interconnected neurons. In general, there are three main architectures for neural networks:

- Single Layer Feedforward Networks
- Multilayer Feedforward Networks
- Recurrent Networks

4.4.1 Single Layer Feedforward Networks

In this architecture, the neural network consists of two layers, the input layer and the output layer (which is also the hidden layer).

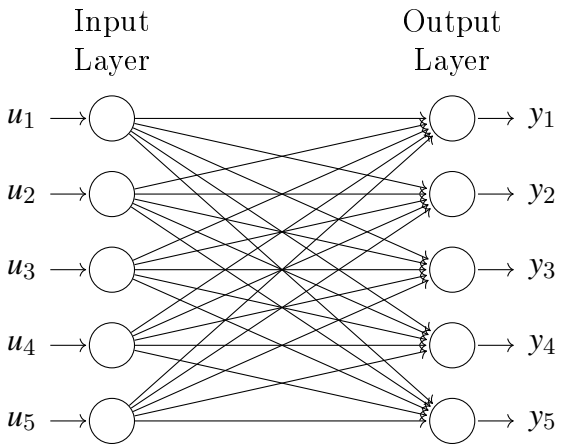


Figure 4.9: Sinlge Layer Feedforward Network

Single layer feedforward networks are commonly used in pattern classification and linear filtering problems [13].

4.4.2 Multilayer Feedforward Networks

The neural network in this architecture is formed by three layers, the input layer, the output layer and one or more hidden layers.

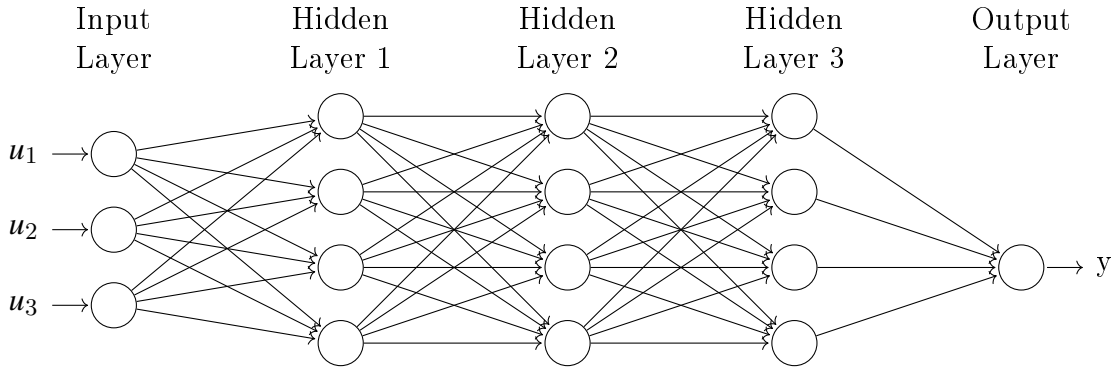


Figure 4.10: Multilayer Feedforward Network with 3 Hidden Layers

This architecture is the most common and is used to solve many problems such as system identification, control, optimization, function approximation and robotics.

4.4.3 Recurrent Networks

The recurrent neural network has the same architecture as a feedforward one, but it distinguishes from it by having at least one feedback loop. The output of a neurons serves as an input for other neurons.

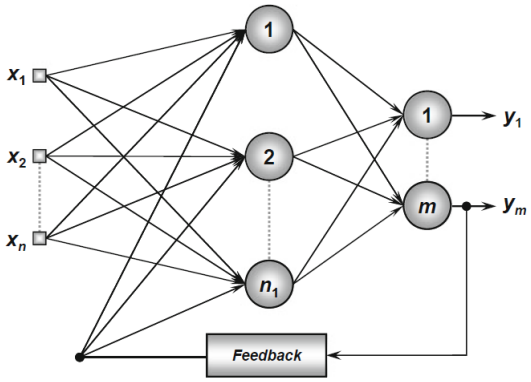


Figure 4.11: Recurrent Network General Structure [13]

This architecture has the feature of dynamic processing meaning that it can be deployed to time-variant systems (time-series prediction, system identification and control).

4.5 Neural Network Learning

Learning process (or rule) is the process of improving the neural network's performance by updating the weights $w_1 ... w_n$ and bias b when the network is applied to a specific set of training data [12]. The learning process can be seen as a measure for the network's performance improvement on future tasks.
Learning processes can be be classified into three categories:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

4.5.1 Supervised Learning

In this learning process, the network has knowledge/information of the environment which is a sequence of input-output data for the system to be modeled.
Given a sequence of N samples input-output data:

$$u_1 y_1, u_2 y_2, ..., u_N y_N$$

We want to find a function f that maps the inputs $u_1 ... u_N$ to the outputs $y_1, ..., y_N$,

$$y = f(u) \tag{4.9}$$

The most common example for supervised learning process is linear regression [14].

4.5.2 Unsupervised Learning

In this learning process, the network has only the input data and no knowledge/information about the output data. The main aim of this learning process is to learn patterns in the input to be able to provide an output by minimizing a given cost function.

Unsupervised learning is majorly used in clustering and association problems [15].

4.5.3 Reinforcement Learning

This learning process is similar to the supervised learning process, but here the learning of an input-output mapping is continuously performed through interaction with the environment in order to maximize a scalar performance index known as the reinforcement signal.

Reinforcement learning is widely used in control problems [16] and game theory [17].

Chapter 5

Virtual Sensor

5.1 Introduction

In this chapter, we introduce the Virtual Sensor modelling procedure followed in order to obtain a model for the K2 clutch position sensor. The main motivation for this modelling process is that the position measurement is not present in real-time operation and hence it has to be estimated by means of a Virtual Sensor.

The K2 clutch position sensor to be modeled is considered as a black-box model, as no apriori information in regards to the model structure is available. As the Virtual Sesnor is modelled in two phases, as it will be introduced in this chapter. The first phase (***CPX*** model, section 5.2) has as inputs the current and the pressure and the output is the clutch position. For the second phase (***PX*** and ***CFPX*** models, section 5.3), the input is the pressure and the output is the position.

As in chapter (3), from preliminarily processing on the available data sets, a dead zone was found to be present. There are three dead zones involved in this modelling problem,

- Dead zone between the current and the pressure.
- Dead zone between the current and the position.
- Dead zone between the pressure and the position.

The Virtual Sensors ,***CPX*** and ***PX***, are in the form of *Moving-Average* models (FIR) and will be modelled using Multilayer Feedforward Neural Networks (for more details, see 4.4.2). This is presented in sections 5.2 and 5.3. In subsection 5.3.4 we model the Virtual Sensor using Curve Fitting techniques motivated by the curve shape describing the input pressure and the output position.

The Median Filter is also introduced in this chapter to eliminate the glitch behaviour that appears on the simulated output of the Virtual Sensor, this can be seen in 5.2.4 for the CPX Model and in 5.3.4 for the CFPX model.

5.2 Virtual Sensor Modelling - CPX Model

5.2.1 Overview

> ***CPX Model***: Neural Network model that has the current, the pressure as inputs and the position as output

In this section we will introduce the modelling of the CPX model. The structure for this model can be seen in figure (5.1) and equation (5.1),

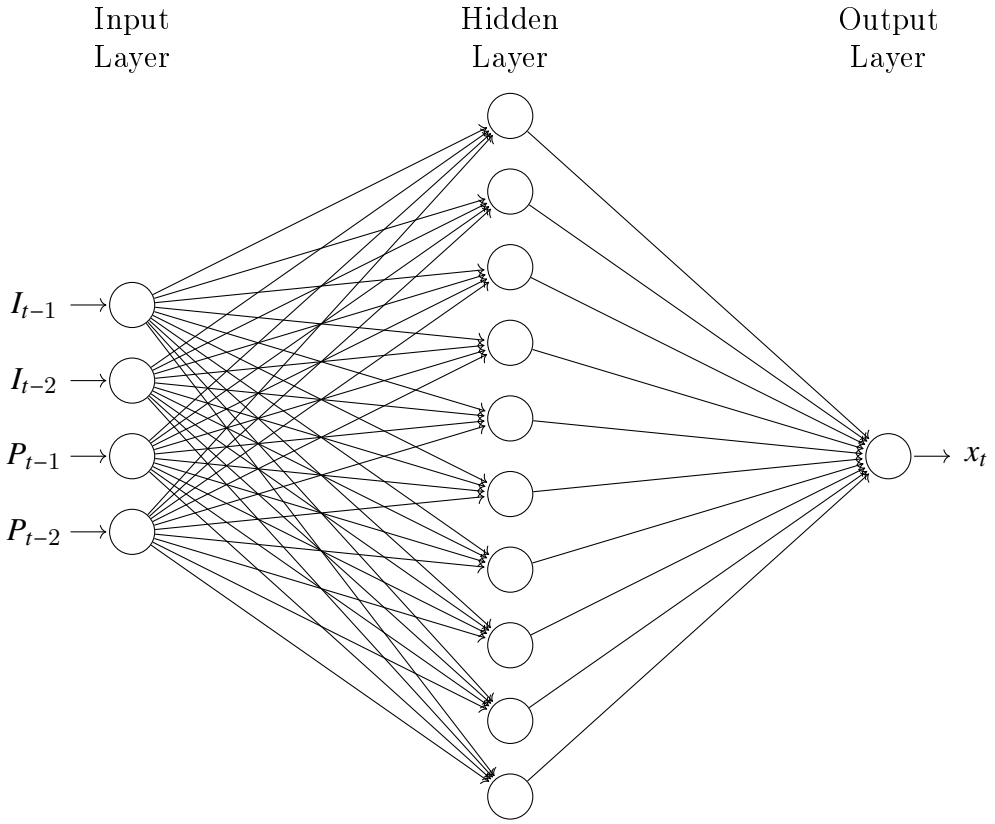


Figure 5.1: CPX Model Structure

$$x_t = f(I_{t-1}, I_{t-2}, P_{t-1}, P_{t-2}) \tag{5.1}$$

From figure (5.1), the description for the three layers is as follows,

- Input Layer: Four inputs, current at time $t-1$ (I_{t-1}), current at time $t-2$ (I_{t-2}), pressure at time $t-1$ (P_{t-1}) and pressure at time $t-2$ (P_{t-2}).
- Hidden Layer: 10 neurons (see 4.3.1 for more details).

- Output Layer: One output, position at time t (x_t).

The CPX model structure, shown in equation (5.1) and figure (5.1), has been selected on the basis of performance and complexity analysis. In table (5.1), we compare different structures for the data set [-20°C Step_a] on the basis of the mean squared error (MSE). It is necessary to point out that sampling time T_s is equal to 2 ms.

Number of Neurons			**Performance MSE**
2	-	-	0.098
5	-	-	0.029
10	-	-	0.016
20	-	-	0.016
100	-	-	0.022
10	5	-	0.012
10	10	-	0.012
10	10	10	0.011
15	5	15	0.011

Table 5.1: *CPX* - Performance Comparison for Different Neural Network Structures

As it can be seen from table (5.1), the networks with 2 and 3 hidden layers provide a slightly better performance in terms of the Mean Squared Error (MSE), but these models add more complexity.

5.2.2 Problem Formulation

In reference to the model structure presented in subsection 5.2.1, we can now define the terms involved in this model. To do this we look inside each of the three layers described in figure (5.1),

Input Layer: Acquires the four inputs I_{t-1}, I_{t-2}, P_{t-1} and $Pt-2$ at each sampling instant T_s. Output of this layer is the 4 acquired inputs.

Hidden Layer: In each $neuron_j$ ($\forall j = 1, 2, ..., 10$), the inputs are adjusted by the weights $w_{1j}, ..., w_{4j}$, then their summation (weighted input) is added to the bias b_{Hj} which increases/decreases the input φ_j to the activation function f. The activation function is a Hyperbolic Tangent Function (see 4.3.2.4), which was found to be the most suitable one for this modelling problem. Output of this layer is the vector y that can be seen in the following formulation,

$$\varphi_j = \begin{bmatrix} w_{1j} & w_{2j} & w_{3j} & w_{4j} \end{bmatrix} \begin{bmatrix} I_{t-1} \\ I_{t-2} \\ P_{t-1} \\ P_{t-2} \end{bmatrix} + b_{Hj} \tag{5.2}$$

$$y_j = f(\varphi_j) = \frac{2}{1 + e^{-2\varphi_j}} - 1 \tag{5.3}$$

$$y^T = \begin{bmatrix} y_1 & \dots & y_{10} \end{bmatrix} \tag{5.4}$$

Output Layer: Provides the Virtual Sensor's position measurement x_t by adjusting the output vector of the hidden layer y by weights $w_{x1}...w_{x10}$ and bias b_x. The resultant output x_t is then given by,

$$x_t = \begin{bmatrix} w_{x1} & \dots & w_{x10} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_{10} \end{bmatrix} + b_x \tag{5.5}$$

5.2.2.1 Modelling Problem Characteristics

Below are some notes in regards to the Virtual Sensor modelling procedure,

- The problem will be modelled using MATLAB'S Neural Network toolbox.
- The data for training the neural network are divided into three sets, 70% for training, 15% for testing and 15% for validation.
- The learning process that will be used is the Supervised Learning process (see 4.5.1 for more details).
- The Supervised Learning training function that will be implemented is the Levenberg-Marquardt optimization algorithm (for more details, see [18]).

5.2.3 Results

*CPX*20: Virtual Sensor modelled using data set 20°C Step_a
*CPX*60: Virtual Sensor modelled using data set 60°C Step_a
*CPXm*20: Virtual Sensor modelled using data set -20°C Step_a
*CPX*20*H*: Virtual Sensor modelled using data set 20°C Step_b

Following the modelling procedure mentioned in subsection 5.2.2, the following figures represent the results obtained by simulating the Virtual Sensor *CPX*20 using three different data sets. The measured output position is represented by y_m and the simulated output position is y_n.

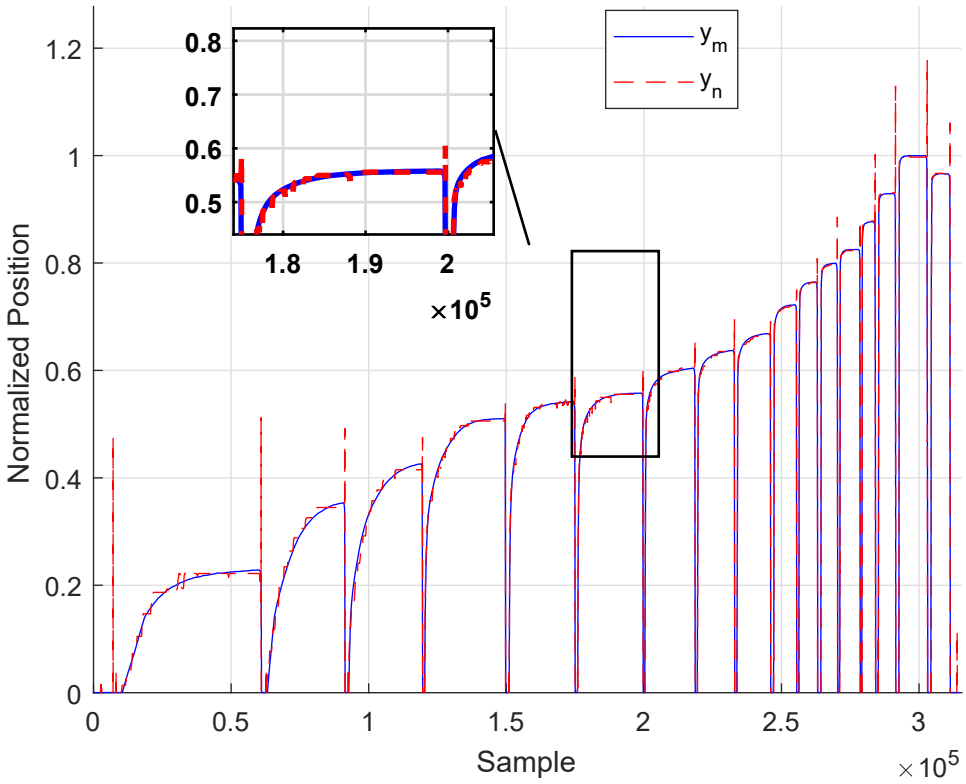


Figure 5.2: *CPX*20 Virtual Sensor Validation for Data Set 20°C Step_a

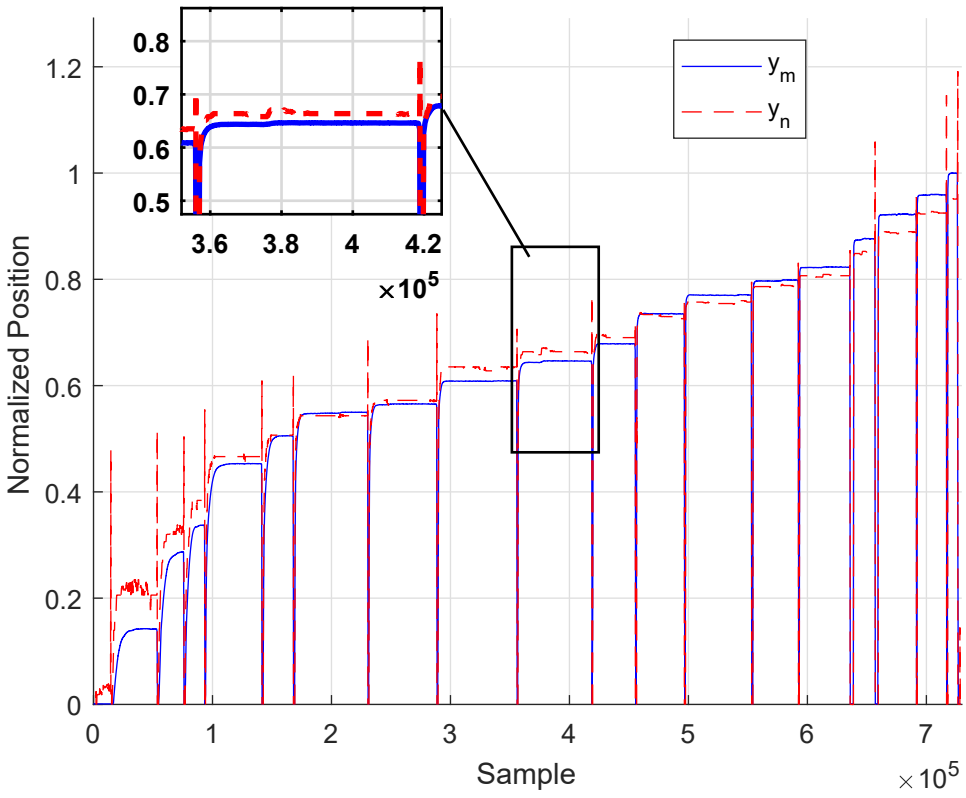


Figure 5.3: *CPX*20 Virtual Sensor Validation for Data Set 60°C Step_a

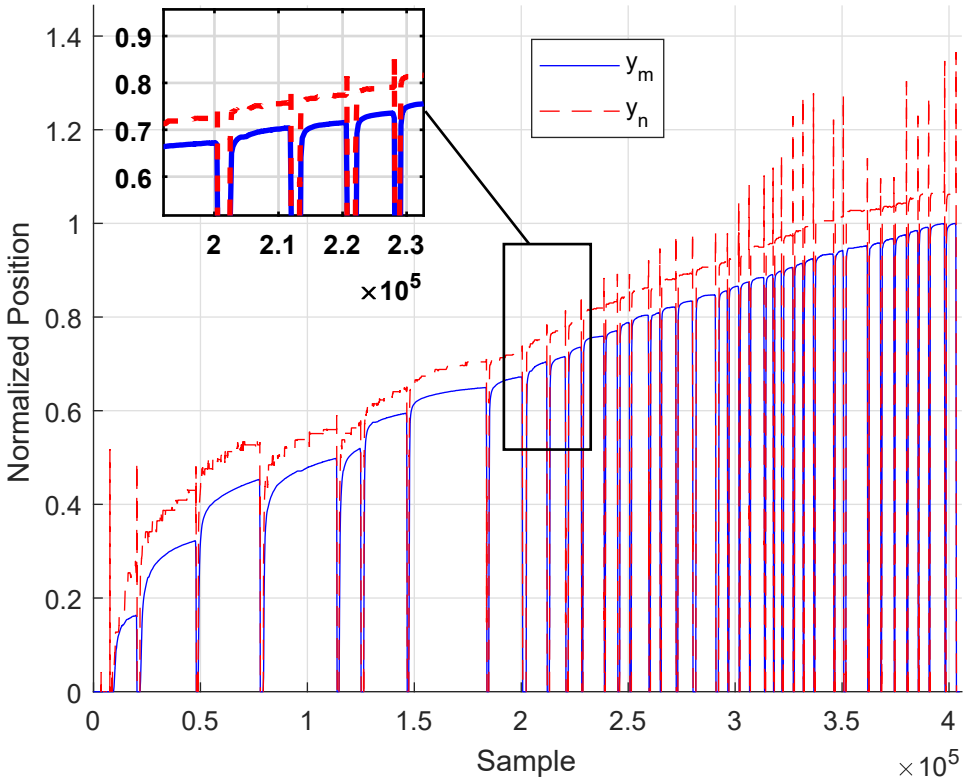


Figure 5.4: *CPX*20 Virtual Sensor Validation for Data Set -20°C Step_a

As it can be seen from figure (5.2), the *CPX*20 Virtual Sensor is able to provide relatively good tracking for the output when simulating the network using the same data set used to train the network (only 70% of this data set was used for training). It can be seen the presence of glitches in the simulated position on the falling edge of the input.

On the other hand, figures (5.3), (5.4) show that the Virtual Sensor *CPX*20 is not able to reproduce the position measurement for the data sets [60°C Step_a] and [-20°C Step_a], respectively. Several Virtual Sensors were modelled, with different hidden layer sizes, number of neurons and architectures, yet no single model was found that can model all the data sets collected at different temperatures. This motivated the fact that we need to design a Virtual Sensor for each operating temperature. The following figures (5.5), (5.6) show the results for each individual Virtual Sensor *CPX*60 and *CPXm*20 as well as the glitch behaviour mentioned above.

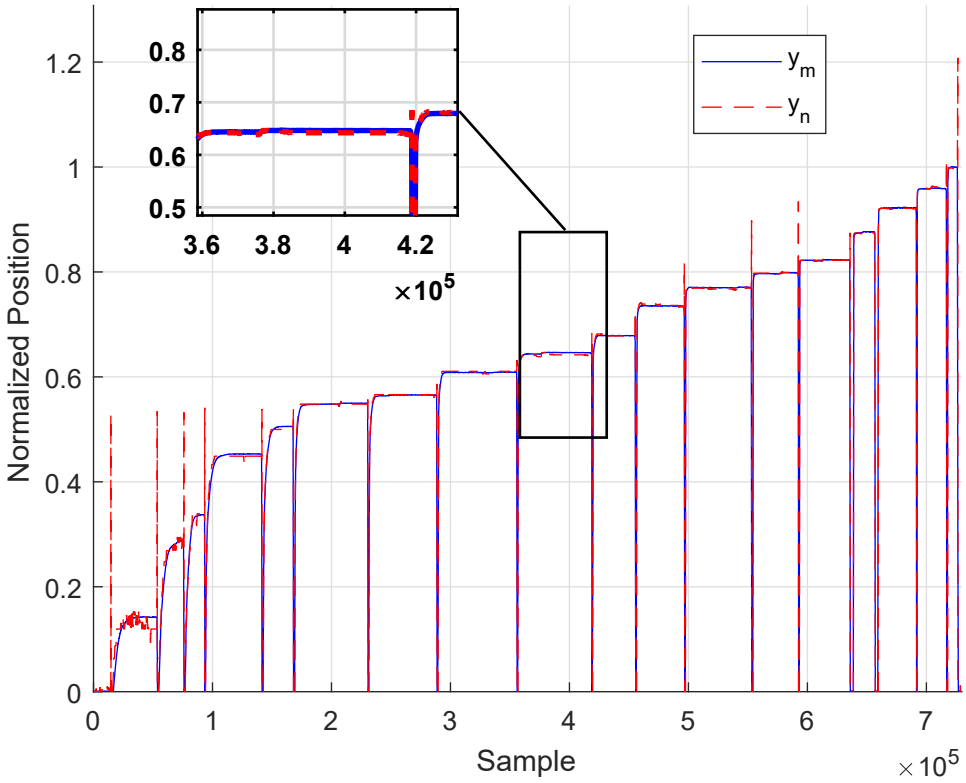


Figure 5.5: *CPX*60 Virtual Sensor Validation for Data Set 60°C Step_a

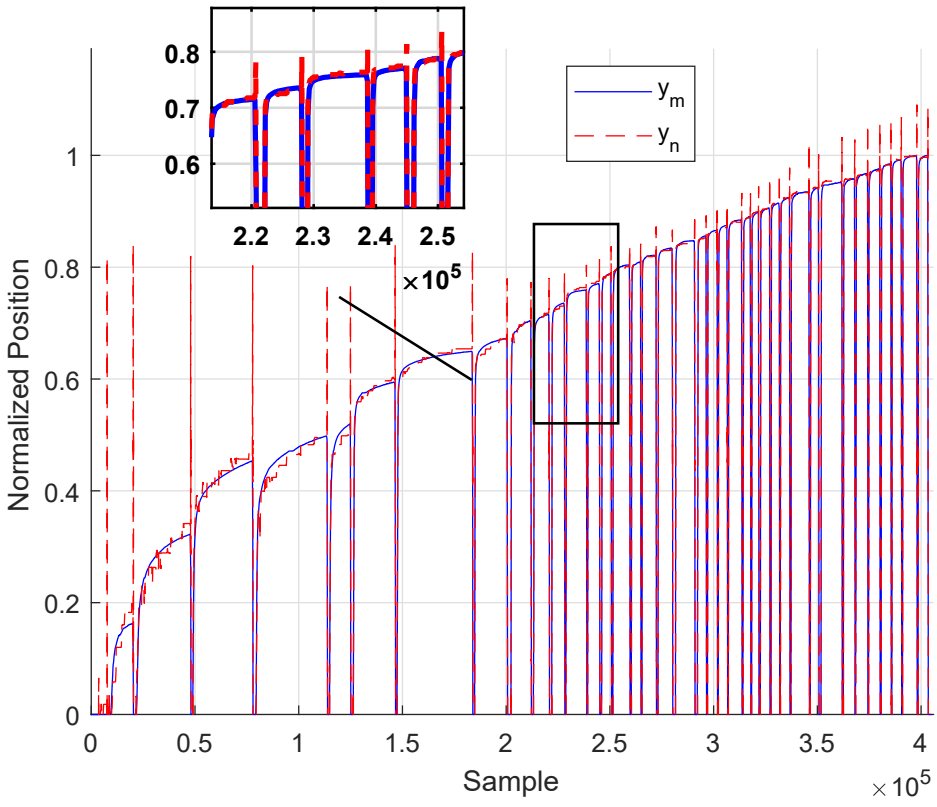


Figure 5.6: *CPXm*20 Virtual Sensor Validation for Data Set -20°C Step_a

For the data sets with **Hysteresis**, the Virtual Sensors were not able to cope up the hysteresis phenomena in regards to each data set separately (figure (5.7) below shows the validation using ***CPX***20 on data set [20°C Step_b]). In figure (5.8), the validation for the Virtual Sensor ***CPX***20***H*** can be seen on data set [20°C Step_b].

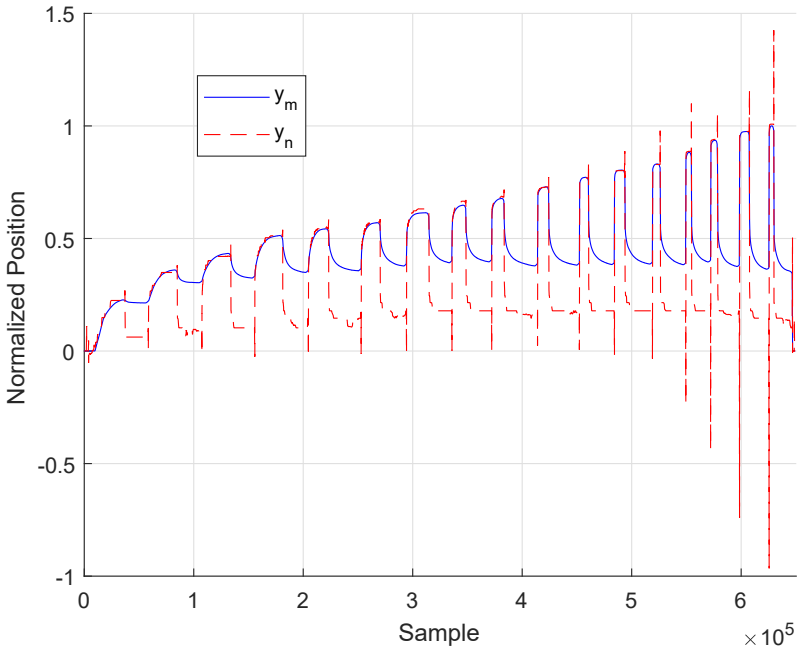


Figure 5.7: ***CPX***20 Virtual Sensor Validation for Data Set 20°C Step_b

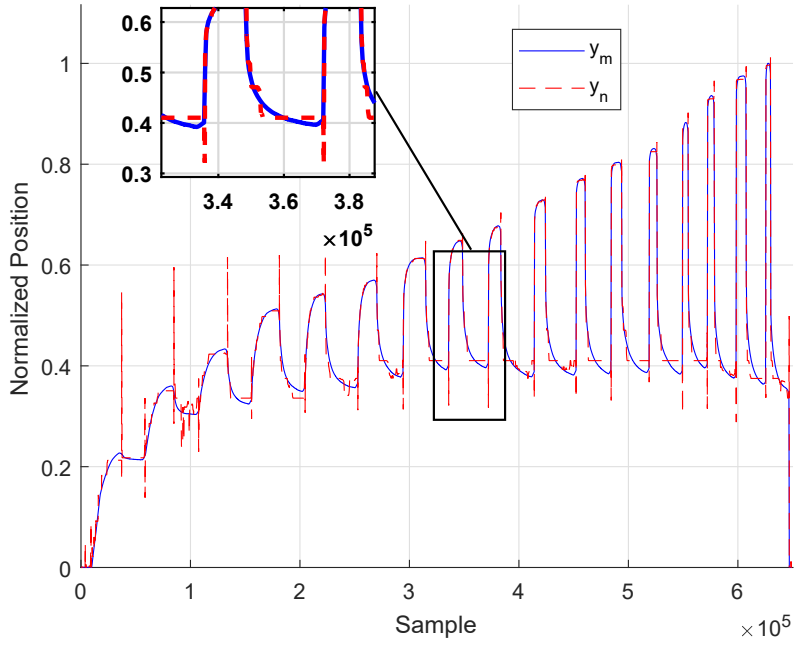


Figure 5.8: ***CPX***20***H*** Virtual Sensor Validation for Data Set 20°C Step_b

For the Virtual Sensor ***CPX***20, the sensor was simulated with partial measurements

from the data set with the input current as a Ramp profile for validation in addition to the data set 20°C Step_a. The results are seen in figure (5.9),

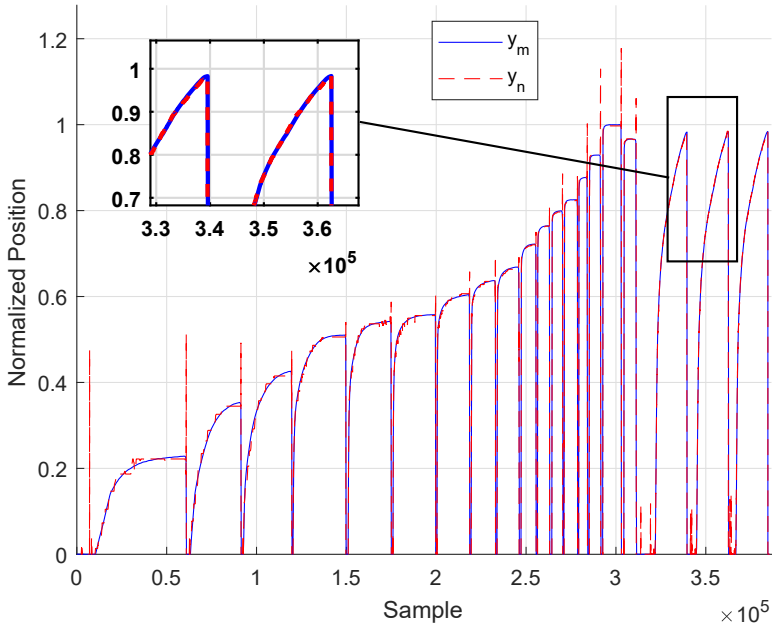


Figure 5.9: *CPX*20 Virtual Sensor Validation for Combined Data Sets at 20°C

5.2.4 Glitch Elimination

Following the results presented in subsection (5.2.3), we will now apply a *One-Dimensional Median Filter* (see [19, 20], for more details) to process the output signal of the ***CPX*** Virtual Sensor (section 5.2), improving the overall performance. In the following figure (5.10), we can see the structure for the Virtual Sensor followed by the Median Filter,

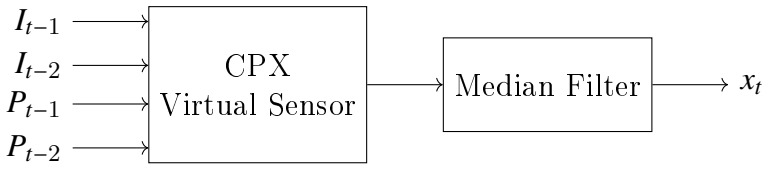


Figure 5.10: CPX Virtual Sensor with Median Filter

The One-Dimensional Median Filter works by replacing every point of a signal with its median. The median is calculated on the basis of the filter's order ρ, where depending on this order the filter uses ρ neighboring points to compute the median of the measurement. Because there are glitches (spikes) in the simulated output position on the falling edge of the input, we need to calculate a suitable order for the filter. On the basis of several

trails, the order of the Median Filter has been selected as,

$$\rho_f = 50$$

In the following figures (5.11), (5.12), (5.13), we can see the results for the ***CPX*** Virtual Sensors modelled in section 5.2 after applying the filter to the neural network's output and enforcing the input deadzone.

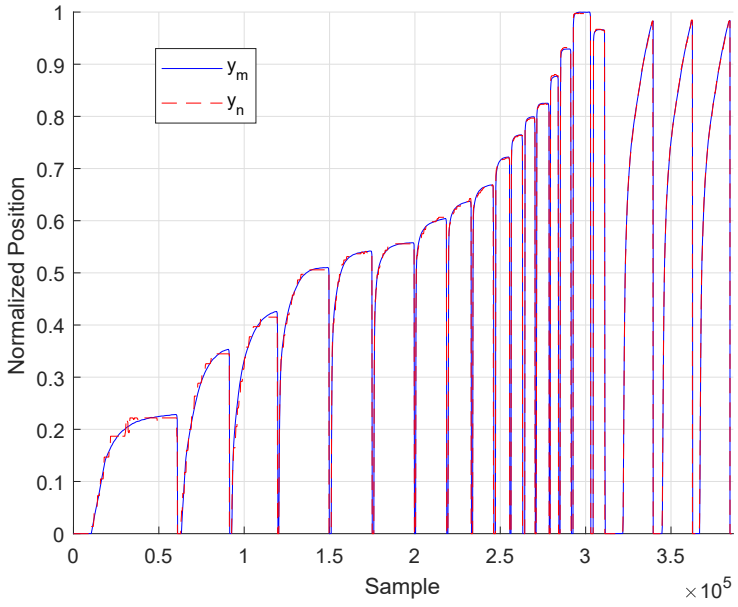


Figure 5.11: ***CPX***20 with Median Filter Validation for Combined Data Sets at 20°C

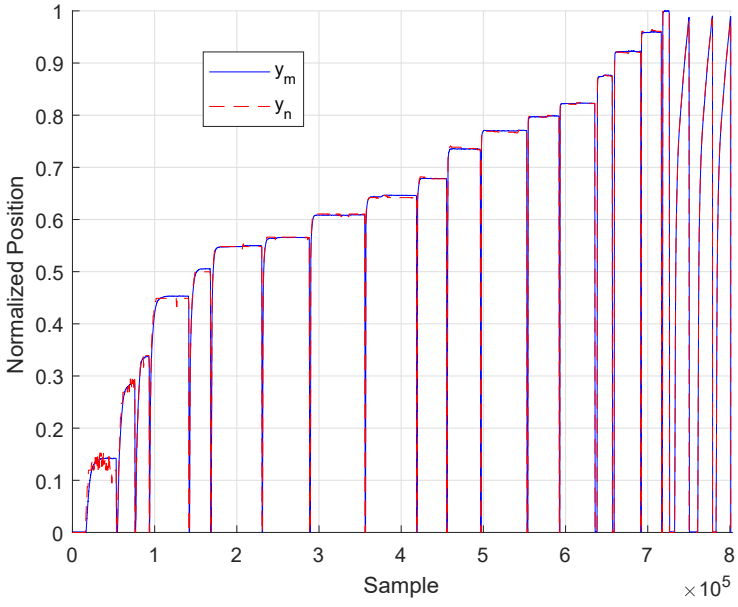


Figure 5.12: ***CPX***60 with Median Filter Validation for Combined Data Sets at 60°C

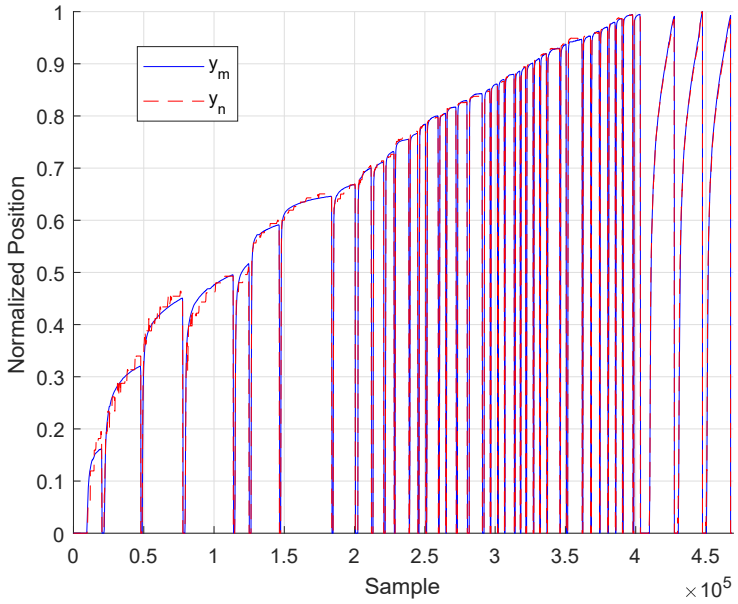


Figure 5.13: *CPXm*20 with Median Filter Validation for Combined Data Sets at -20°C

For the data sets with *Hysteresis*, there was no improvement in the performance, hence leading to the conclusion that the currently used structures are not suitable to model this behaviour.

5.2.5 Conclusions

We can conclude this section by stating the following points:

- Virtual Sensors designed at a specific operating temperature are not able to reproduce the position measurements at other temperatures, hence a Virtual Sensor is needed for each operating temperature.
- In regards to the simulated position, a glitch was present in the falling edge of the input. This motivated the need to introduce a Median Filter on the output of the Virtual Sensor which was able to eliminate the glitch behaviour.
- The *CPX*20*H* Virtual Sensor is not able to reproduce the measured position x_t.

5.3 Virtual Sensor Modelling - PX Model

5.3.1 Overview

PX Model: Neural Network model that has the pressure as input and the position as output

In this section we now look at another model structure for the Virtual Sensor, motivated by the curve shape, as seen in figure (5.14), and with the aim of reducing the model complexity we now introduce the PX model.

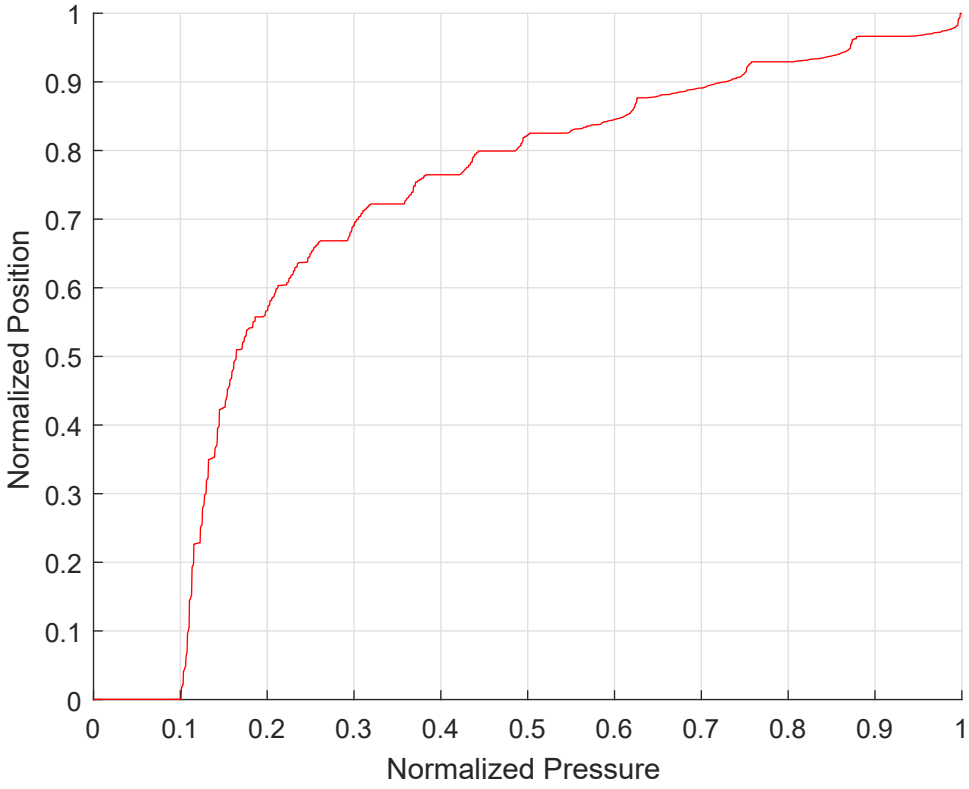


Figure 5.14: Pressure-Position Curve Shape for Data Set 20°C Step_a

The structure for this model can be seen in figure (5.15) and equation (5.6). The three layers are described as follows,

- Input Layer: One input, pressure at time t (P_t).
- Hidden Layer: 10 neurons (same structure as described in subsection 5.2.1).
- Output Layer: One output, position at time t (x_t).

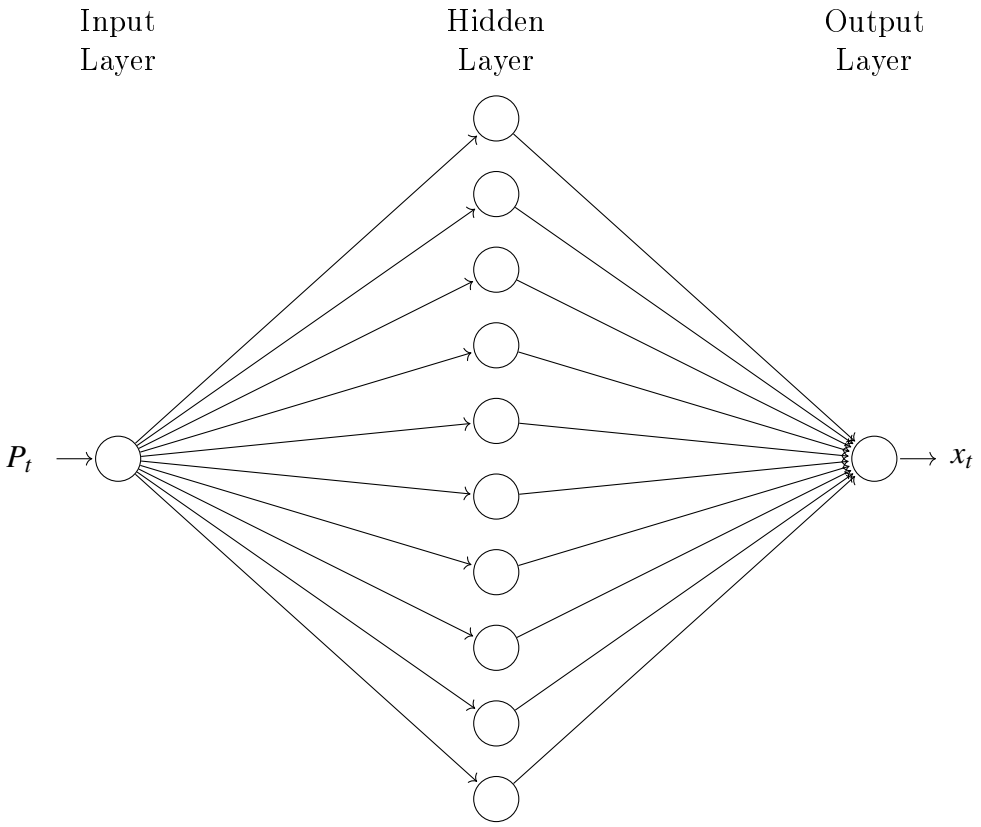


Figure 5.15: PX Model Structure

$$x_t = f(P_t) \tag{5.6}$$

From figure (5.15), the three layers are decribed as follows,

- Input Layer: One input, pressure at time t (P_t).
- Hidden Layer: 10 neurons (same structure as described in subsection 5.2.1).
- Output Layer: One output, position at time t (x_t).

The PX model structure, described in equation (5.6) and figure (5.15), has been selected on the basis of performance analysis. In table (5.2), we compare different structures for the data set [20°C Step_a] on the basis of the acheived training performance. It is necessary to point out that sampling time T_s is equal to 2 ms.

Number of Neurons			Performance MSE
2	-	-	0.06
5	-	-	0.04
10	-	-	0.039
20	-	-	0.039
100	-	-	0.038
10	5	-	0.038
10	10	-	0.038
10	10	10	0.037
15	10	15	0.037

Table 5.2: *PX* - Performance Comparison for Different Neural Network Structures

As it can be seen from table (5.2), the networks with 3 hidden layers provide a slightly better performance in terms of the Mean Squared Error (MSE), but these models add more complexity.

5.3.2 Problem Formulation

In reference to the model structure presented in subsection 5.3.1, we can now define the terms involved in this model. To do this we look inside each of the three layers described in figure (5.15),

Input Layer: Acquires the input P_t at each sampling instant T_s. Output of this layer is the 1 acquired input.

Hidden Layer: In each $neuron_j$ $(\forall j = 1, 2, ..., 10)$, the input is adjusted by the weight w_j, then the weighted input is added to the bias b_j which increases/decreases the input φ_j to the activation function f. The activation function is a Hyperbolic Tangent Function (see 4.3.2.4), which was found to be the most suitable one for this modelling problem. Output of this layer is the vector y that can be seen in the following formulation,

$$\varphi_j = w_j P_t + b_j \tag{5.7}$$

$$y_j = f(\varphi_j) = \frac{2}{1 + e^{-2\varphi_j}} - 1 \tag{5.8}$$

$$y^T = \begin{bmatrix} y_1 & \dots & y_{10} \end{bmatrix} \tag{5.9}$$

Output Layer: Provides the Virtual Sensor's position measurement x_t by adjusting the output vector of the hidden layer y by weights $w_{x1}...w_{x10}$ and bias b_x. The

resultant output x_t is then given by,

$$x_t = \begin{bmatrix} w_{x1} & \dots & w_{x10} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_{10} \end{bmatrix} + b_x \tag{5.10}$$

Modelling problem characteristics for the design of the ***PX*** Virtual Sensor are the same as for the CPX model (for more details, see 5.2.2.1).

5.3.3 Results

PX20: Virtual Sensor modelled using data set 20°C Step_a
PX60: Virtual Sensor modelled using data set 60°C Step_a
PXm20: Virtual Sensor modelled using data set -20°C Step_a

We introduce in this subsection the results for the three Virtual Sensors modelled at different temperatures following the modelling procedure introduced in subsection (5.3.2). The measured output position is represented by y_m and the simulated output position is y_n.

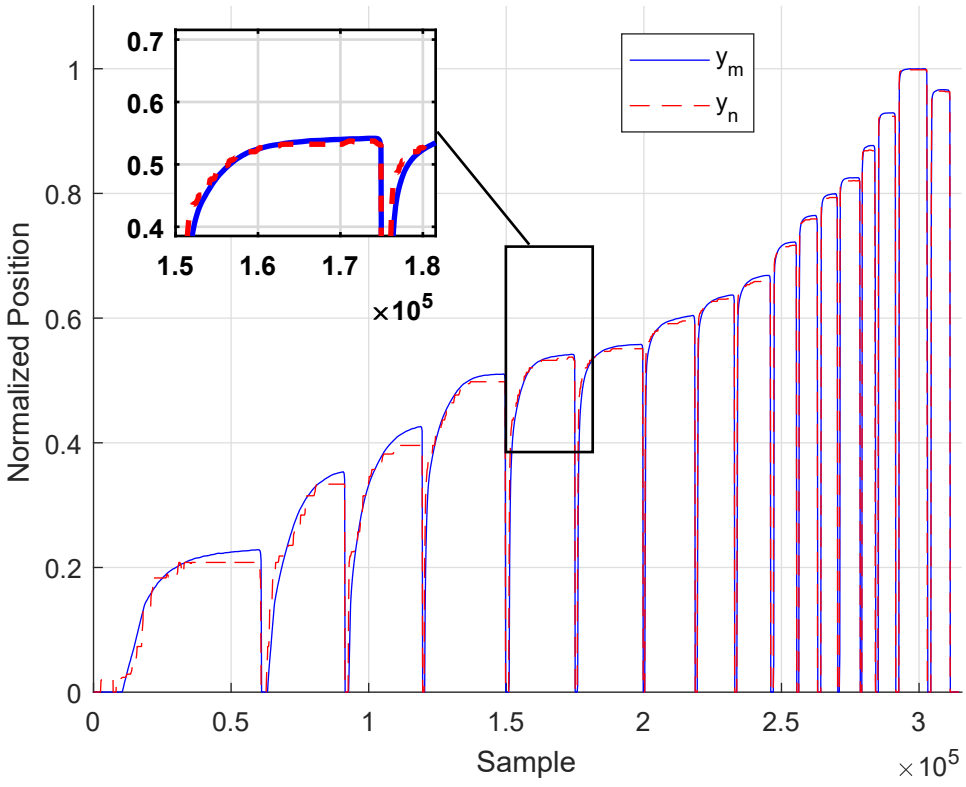


Figure 5.16: ***PX***20 Virtual Sensor Validation for Data Set 20°C Step_a

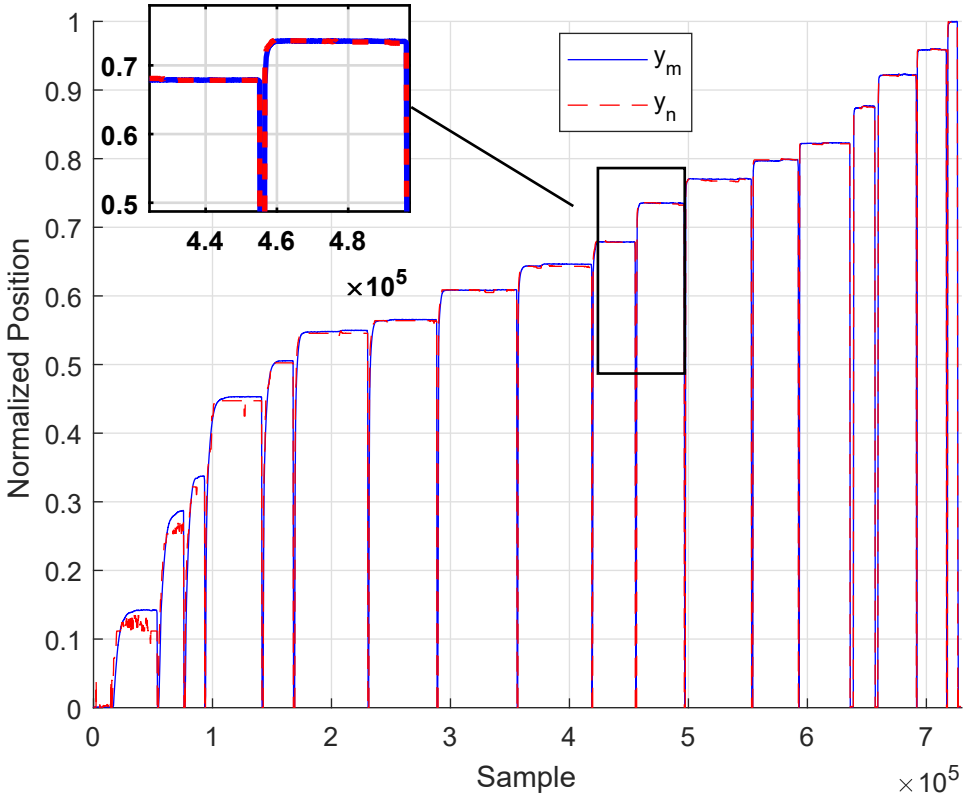


Figure 5.17: *PX*60 Virtual Sensor Validation for Data Set 60°C Step_a

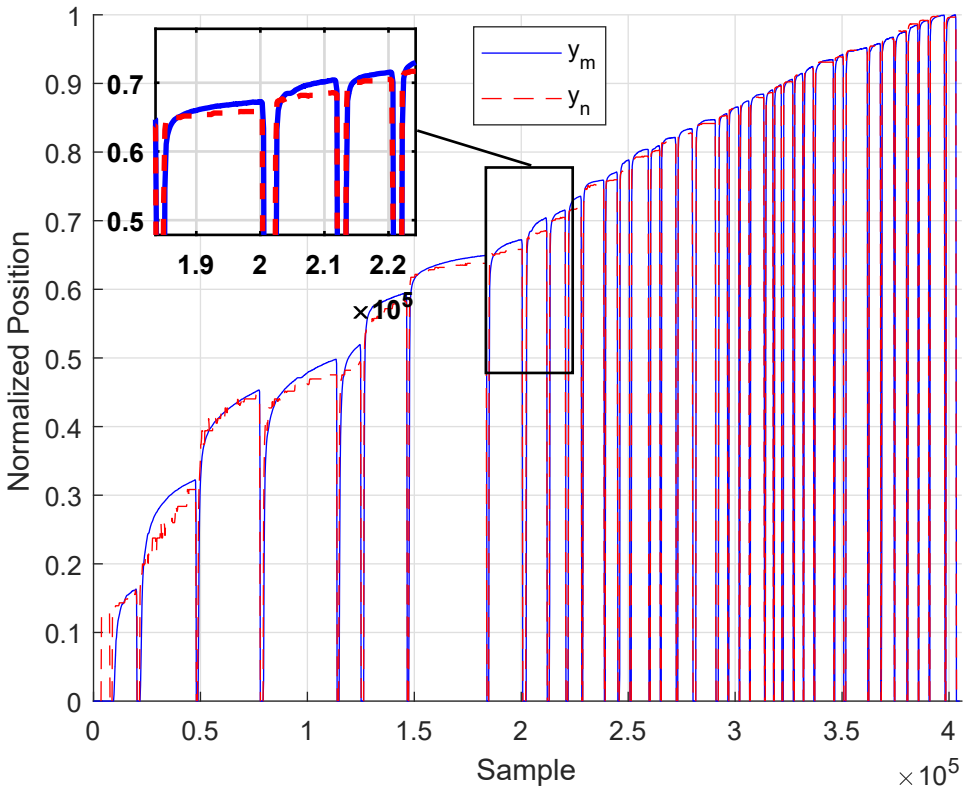


Figure 5.18: *PXm*20 Virtual Sensor Validation for Data Set -20°C Step_a

As it can be seen from figures (5.16), (5.17), (5.18), the Virtual Sensors are able to track the measured output position x_t with some deviations in the low position measurements and the input dead zone region which is not properly modelled. We model the dead zone by including an enforcement term in the Virtual Sensor output. The results can be seen in figures (5.19), (5.20) and (5.21).

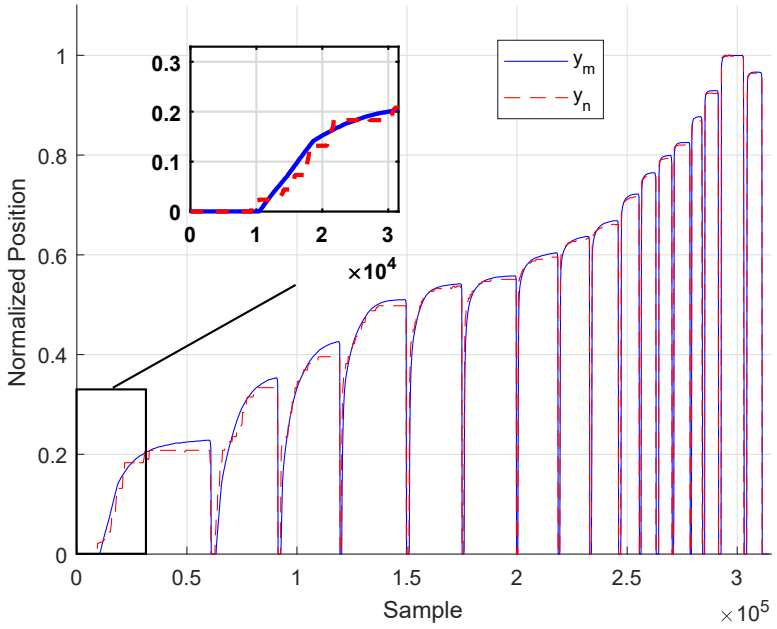


Figure 5.19: *PX*20 Virtual Sensor Validation for Data Set 20°C Step_a with enforced Input Dead Zone

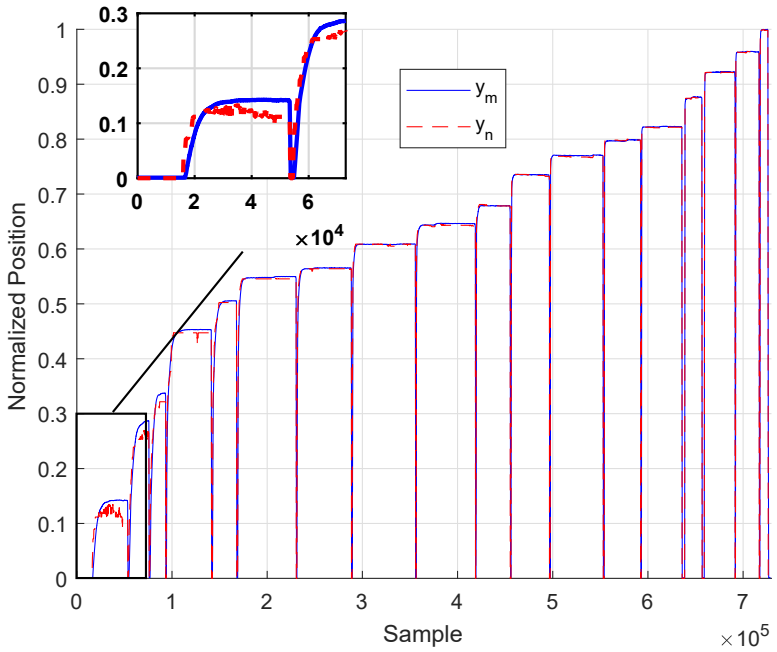


Figure 5.20: *PX*60 Virtual Sensor Validation for Data Set 60°C Step_a with enforced Input Dead Zone

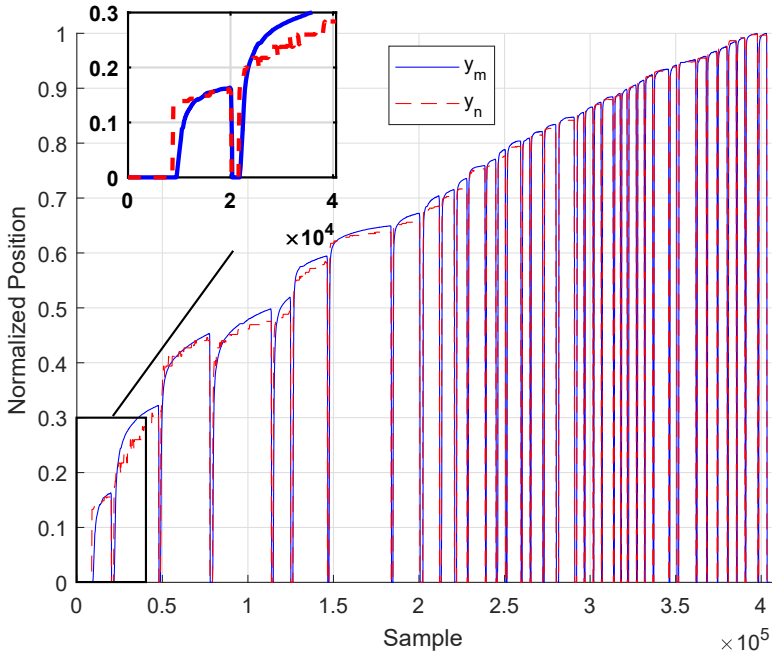


Figure 5.21: *PXm*20 Virtual Sensor Validation for Data Set -20°C Step_a with enforced Input Dead Zone

Next we include partial measurements from the data sets with the input current as a Ramp profile for validation. The three Virtual Sensors provide good tracking performance for the rising edge of the ramp signal as it can be seen in figures (5.22), (5.23), (5.24), but it is not able to follow the falling edge of the signal as in figure (5.25).

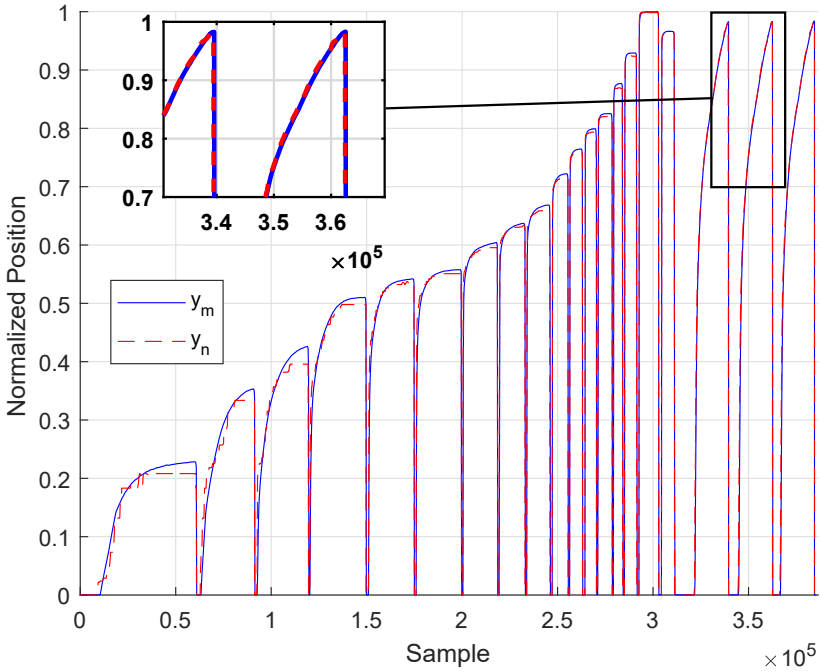


Figure 5.22: *PX*20 Virtual Sensor Validation for Combined Data Sets at 20°C with enforced Input Dead Zone

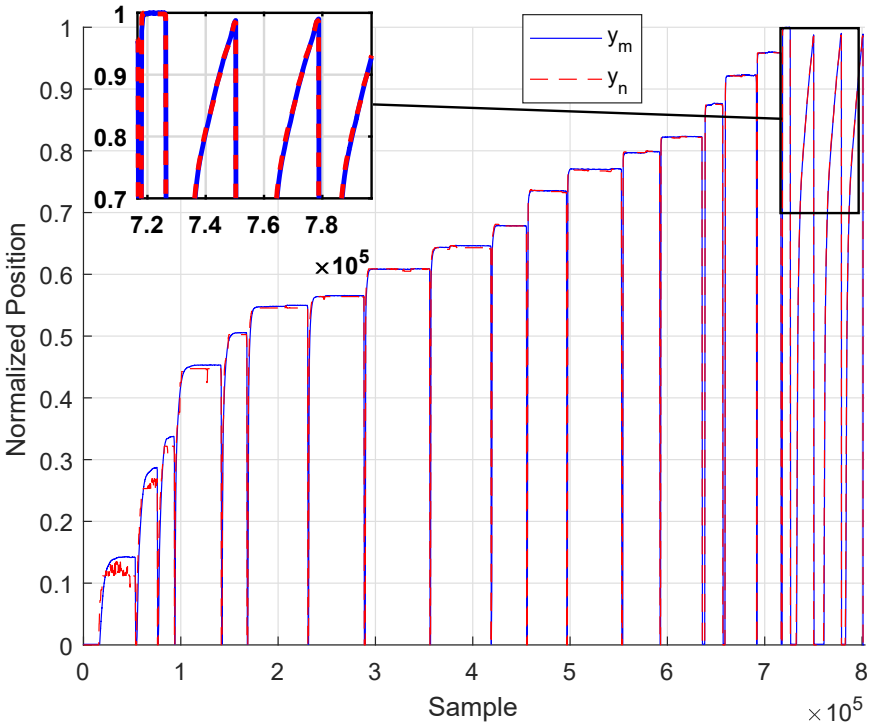


Figure 5.23: *PX*60 Virtual Sensor Validation for Combined Data Sets at 60°C with enforced Input Dead Zone

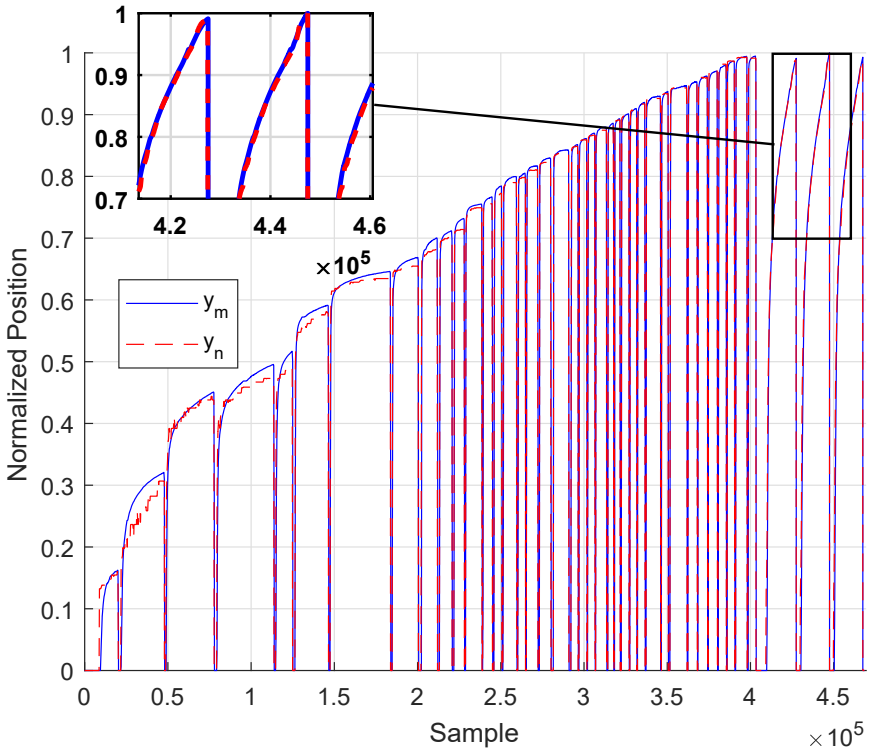


Figure 5.24: *PXm*20 Virtual Sensor Validation for Combined Data Sets at -20°C with enforced Input Dead Zone

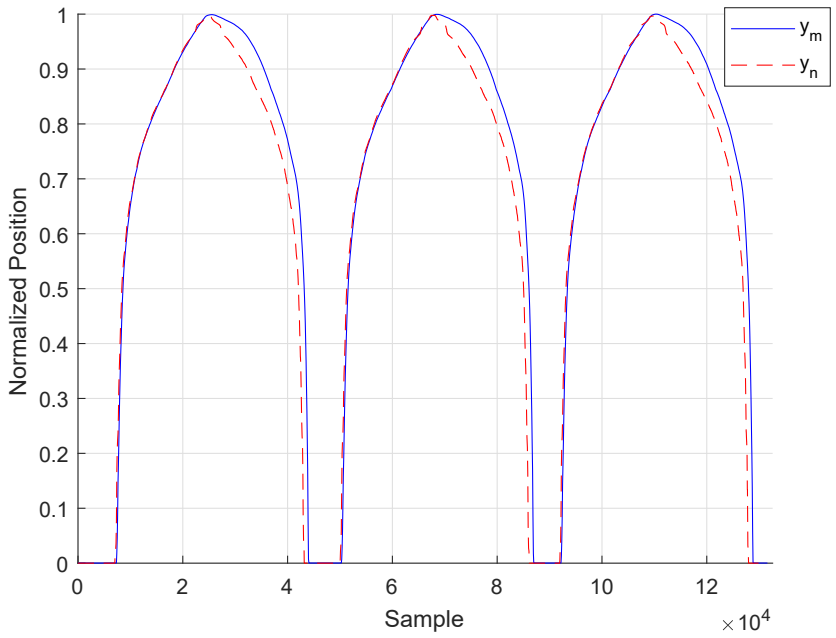


Figure 5.25: *PX*20 Virtual Sensor Validation for Ramp Profile Data Set at 20°C with enforced Input Dead Zone

The performance of the CPX20H network, figure (5.8), was better than the PX20H network shown in figure (5.26) below, which implies that the hysteresis behaviour needs to be modelled with a high order dynamic model.

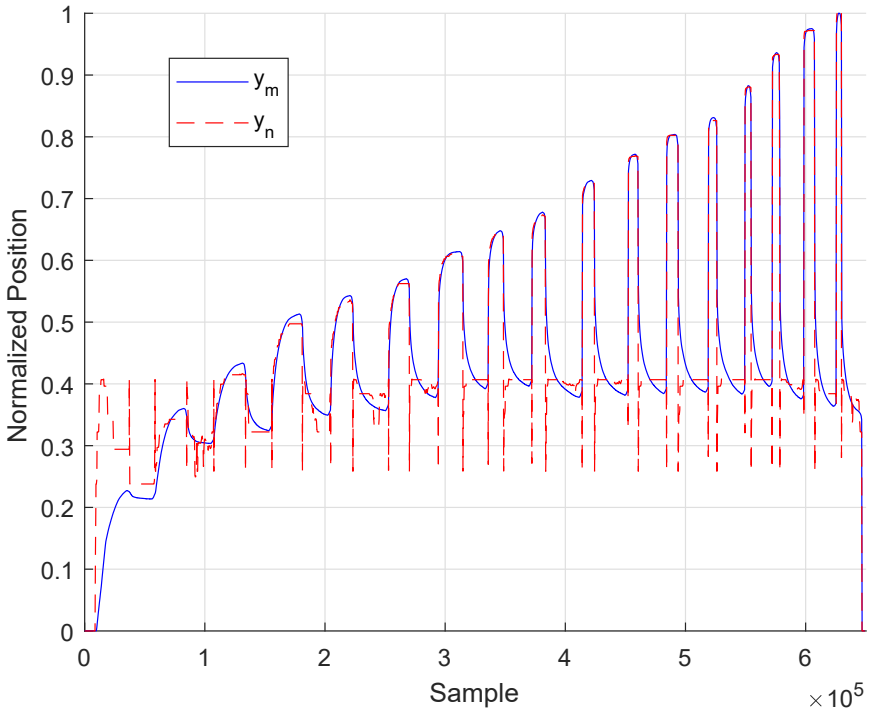


Figure 5.26: *PX*20*H* Virtual Sensor Validation for Data Set 20°C Step_b

5.3.4 Neural Networks vs Curve Fitting

CFPX Model: Curve-Fitting model that has the pressure as input and the position as output
CFPX20: Virtual Sensor modelled using data set 20°C Step_a and polynomial order $\rho = 20$

As it can be seen from subsection 5.3.2, the *PX* Virtual Sensor is a static model, hence this motivates the approach of using Curve Fitting techniques to model the *PX* Virtual Sensor. The MATLAB Curve Fitting Toolbox will be used to design a finite and known order polynomial to fit the input-output data.

A comparison on the basis of the Mean Squared Error (MSE) for different polynomial structures is presented in table (5.3) for the data set [20°C Step_a] using the MATLAB Curve Fitting Toolbox. It is necessary to state that the *PX*20 Virtual Sensor modelled with the same data set achieved a performance of ***0.039*** as shown in table (5.2).

Polynomial Order (ρ)	**Performance MSE**
5	0.2458
10	0.0821
20	0.0431
50	0.0420
100	0.0419

Table 5.3: *CFPX* - Performance Comparison for Different Polynomial Structures

The polynomial f that models the Virtual Sensor using Curve Fitting techniques is as follows,

$$x_t = f(P) = \sum_{j=1}^{m} \gamma_j P_t^{m-j+1} \tag{5.11}$$

where $\gamma_j \in \mathbb{R}^m$ is the j^{th} polynomial f coefficient.
In figure (5.27), we can see a graphical comparison between the *PX*20 model and three polynomials with orders 5, 10 and 20 applied on data set [20°C Step_a]. It is necessary to point out that the input dead zone has been enforced on all four models. The measured output position is represented by y_m and the simulated output position is y_n.

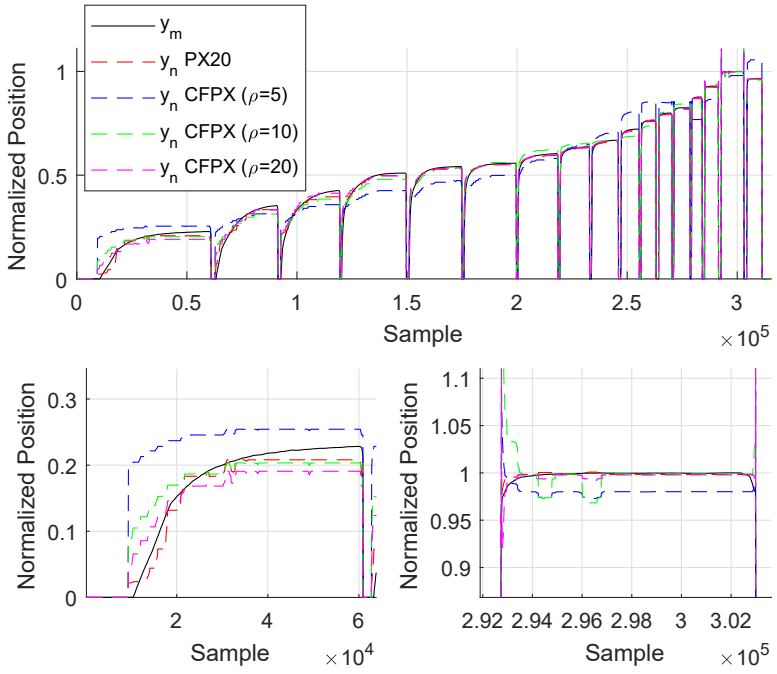


Figure 5.27: $PX20$ vs $CFPX$ Models for Data Set 20°C Step_a

From figure (5.27), we can see that the performance for the $CFPX$ model, with $\rho = 20$, almost matches the $PX20$, but it introduces glitches on both sides of the signal (rising and falling) at high values for the position measurement x_t. As it can be seen in figure (5.28), the glitch has been removed by applying a One Dimensional Median Filter with order $\rho_f = 250$ (see 5.2.4 for more details).

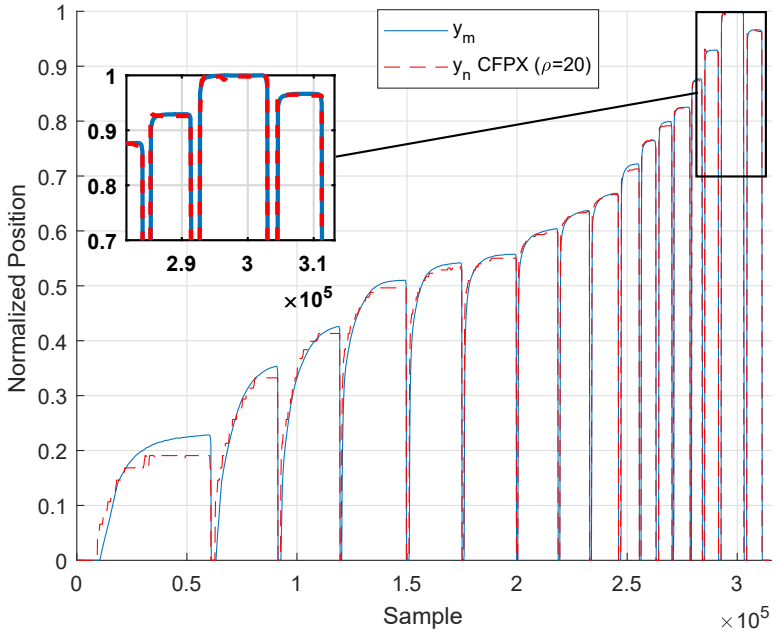


Figure 5.28: $CFPX20$ with Median Filter Validation for Data Set 20°C Step_a

In figure (5.29), we can see a comparison between the $CPX20$, $PX20$ and $CFPX20$

Virtual Sensors applied on data set [20°C Step a]. It can be stated that the three structures provide relatively similar results, except for the low position measurements as the *CPX*20 model has an overall better performance.

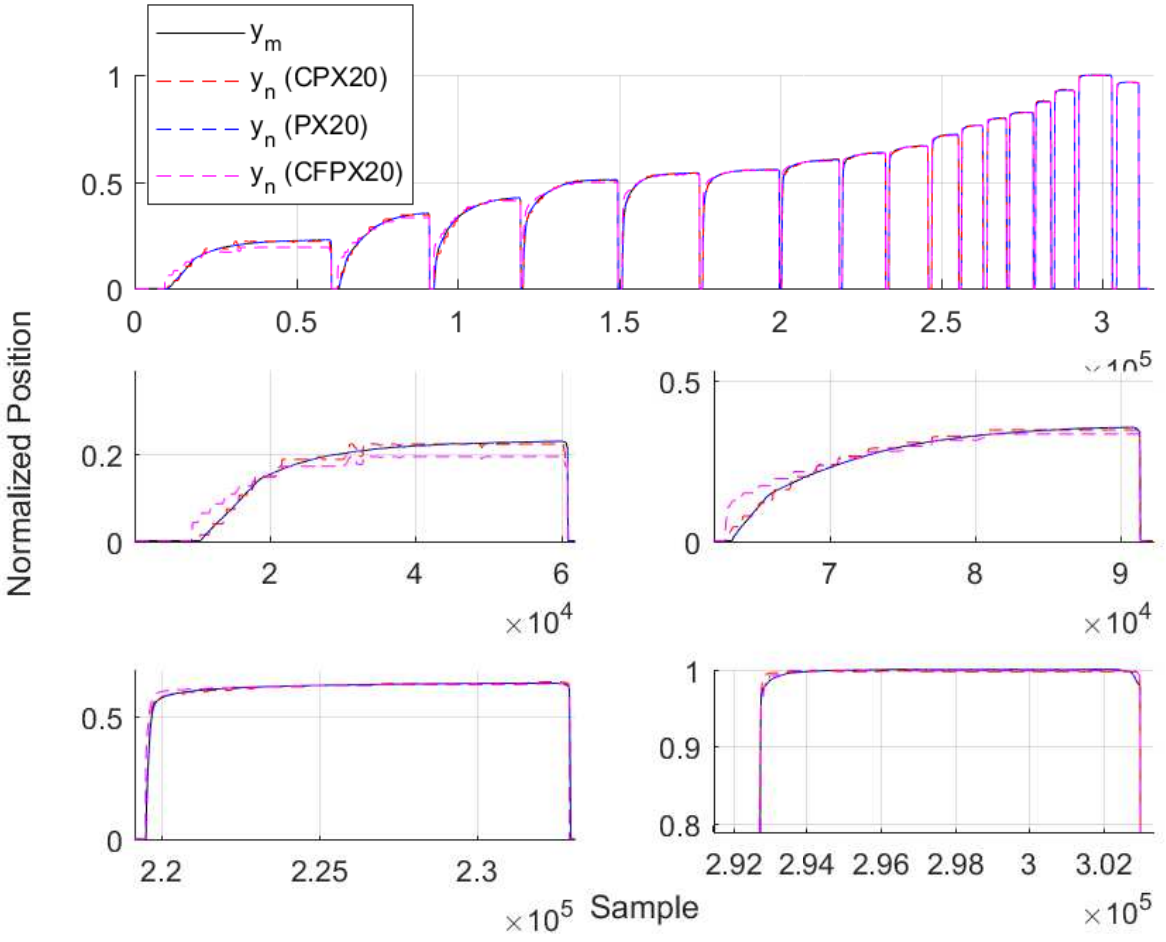


Figure 5.29: *CPX*20, *PX*20, *CFPX*20 Output Comparison for Data Set 20°C Step_a

In table (5.4), we present a comparison for the three models in figure (5.29) in terms of complexity and performance. The performance is on the basis of the Mean Squared Error (MSE) following the glitch filtering and the input dead zone enforcement.

Virtual Sensor Model	**Number of Parameters**	**Performance MSE**
*CPX*20	61	0.0156
*PX*20	31	0.0387
*CFPX*20	21	0.0491

Table 5.4: Complexity and Performance Comparison for *CPX*20, *PX*20, *CFPX*20 Virtual Sensors

5.3.5 Conclusions

We can conclude this section by stating the following points:

- Virtual Sensors designed at a specific operating temperature are able to reproduce the position measurements at the same temperature. An exception for this is the *Hysteresis* behaviour, as it can not be modeled with all the presented structures in this chapter.
- The One-Dimensional Median Filter has proven it's capability to eliminate the glitches present in the transient phase of the signal.
- The three Virtual Sensor models, CPX, PX, $CFPX$, provide similar results in terms of reproducing the position measurement x_t, with a slight better performance for the CPX models in the low position range.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this section, we present a summary for the work completed throught his thesis and highlight the obtained results.

In chapter (3), the system identification process for the ***Hammerstein*** system under the *Set-Membership* framework was able to model the $K2$ actuator plant when applying the data sets with the ramp input current, however, when applying the step input, the system was not able to reproduce the output pressure. We then moved to the *Recursive-Estimation* framework which proved to be suitable for this identification problem as we managed to estimate a model for the plant at each sampling instant. The relative error between the measured and simulated pressures is bounded by $|2 \times 10^{-14}|$.

In Chapter (5), we explored several architectures to model a *Virtual Sensor* to estimate the clutch position, as these measurements are not available in the real application. We used two structures of *Multilayer Feedforward Neural Networks* and a *Curve Fitting* finite, known order polynomial to model the virtual sensor. The first *Neural Network* structure is in the form of a dynamic model with two input delays, and this model provided relatively good results for the output position after glitch elimination using a *Median Filter*.

Following that, a static *Virtual Sensor*, with the pressure at time t as the input and the position at time t as the output, was designed using two different model structures. The *Neural Network* structure and the *Curve Fitting*, after glitch elimination through a *Median Filter*, gave almost the same results.

Comparing the three structures for the *Virtual Sensor*, the first model (***CPX*** model, as referred to in chapter (5), section 5.2), had a better performance in terms of the *Mean Squared Error* (MSE) and measured position tracking.

6.2 Future Work

Starting from the results achieved in this thesis, we need to exploit the possibilty of introducing a recursive form for the *Set-Membership* framework.

Furthermore, a generalized structure to model the *Virtual Sensor* is an area of concern, as we need to obtain a single model that is capable of taking into account all the operating temperatures and to reproduce the hysteresis behaviour.

Bibliography

[1] Cimmino Vafidis, Constantinos and Francesco. "Fpt's high torque density dual dry clutch transmission (htd-ddct)", Fiat Powertrain Research and Technology. 1.1

[2] Maurizio Zoppi, Claudio Cervone, Gerardo Tiso, and Francesco Vasca. Software in the loop model and decoupling control for dual clutch automotive transmissions. In Systems and Control (ICSC), 2013 3rd International Conference on, pages 349–354. IEEE, 2013. 1.1.2

[3] Ljung, L. (1999). System identification theory for the user (2nd ed., Prentice Hall information and system sciences series). Upper Saddle River: Prentice Hall PTR. 2.1, 3.3.2

[4] Söderström, T., & Stoica, P. (1989). System identification (Prentice- Hall International series in systems and control engineering). New York: Prentice- Hall. 2.1

[5] Norton, J. (1985). Identification and Application of Bounded-Parameter Models. IFAC Proceedings Volumes, 18(5), 1197-1202. 2.1

[6] Cerone, Piga, & Regruto. (2012). Set-Membership Error-in-Variables Identification Through Convex Relaxation Techniques. Automatic Control, IEEE Transactions on, 57(2), 517-522. 2.2.1.3

[7] Ljung, L. (2001). Black-box Models from Input-output Measurements. 3.1

[8] Cerone V, Piga D, & Regruto D. (2011). Enforcing stability constraints in Set-membership identification of linear dynamic systems. In: AUTOMATICA, In: AUTOMATICA. - ISSN: 0005-1098. 3.2.2.4

[9] Waki, H., Kim, Kojima, Muramatsu, & Sugimoto. (2008). Algorithm 883: SparsePOP—A Sparse Semidefinite Programming Relaxation of Polynomial Optimization Problems. ACM Transactions on Mathematical Software (TOMS), 35(2), 1-13. 3.2.3

[10] Young, P. (1984). Recursive estimation and time- series analysis an introduction (Communications and Control Engineering Series). Berlin: Springer. 3.3.1

[11] Van Gerven, M. (2017). Computational Foundations of Natural Intelligence. Frontiers in Computational Neuroscience, 11, 112. 4.1

[12] Haykin, S. (1999). Neural networks a comprehensive foundation (2nd ed.). Upper Saddle River: Prentice Hall. 4.1, 4.2, 4.5

[13] da Silva I.N., Hernane Spatti D., Andrade Flauzino R., Liboni L.H.B., dos Reis Alves S.F. (2017) Artificial Neural Network Architectures and Training Processes. In: Artificial Neural Networks. Springer, Cham. (document), 4.4.1, 4.11

[14] Murphy, K. (2012). Machine learning a probabilistic perspective (Adaptve computation and machine learning series). Cambridge (Mass.): MIT Press. 4.5.1

[15] R. Rojas: Neural Networks, Springer-Verlag, Berlin, 1996. 4.5.2

[16] Vichugov, V.N., Tsapko, G.P., & Tsapko, S.G. (2005). Application of reinforcement learning in control system development. Science and Technology, 2005. KORUS 2005. Proceedings. The 9th Russian-Korean International Symposium on, 732-733. 4.5.3

[17] Nowé A., Vrancx P., De Hauwere YM. (2012) Game Theory and Multi-agent Reinforcement Learning. In: Wiering M., van Otterlo M. (eds) Reinforcement Learning. Adaptation, Learning, and Optimization, vol 12. Springer, Berlin, Heidelberg. 4.5.3

[18] Gill, P. R.; Murray, W.; and Wright, M. H. "The Levenberg-Marquardt Method." §4.7.3 in Practical Optimization. London: Academic Press, pp. 136-137, 1981. 5.2.2.1

[19] Pratt, William K. Digital Image Processing. 4th Ed. Hoboken, NJ: John Wiley & Sons, 2007. 5.2.4

[20] Kendall, M., & Stuart, M. (1977). The advanced theory of statistics (4th ed.). London: Griffin. 5.2.4